# Namespace DotNetBrowser

## Interfaces

IAutoDisposable

Represents the object which can be disposed by itself (without the explicit Dispose method call).

IAutoDisposable<TArgs>

Represents the object which can be disposed by itself (without the explicit Dispose method call).

# Interface IAutoDisposable

Namespace: DotNetBrowser

Assembly: DotNetBrowser.dll

Represents the object which can be disposed by itself (without the explicit Dispose method call).

```
public interface IAutoDisposable
```

## Properties

IsDisposed

Indicates if the object is already disposed.

## Events

Disposed

Occurs when the object has been disposed.

# Property IsDisposed

Namespace: DotNetBrowser

Assembly: DotNetBrowser.dll

## IsDisposed

Indicates if the object is already disposed.

```
bool IsDisposed { get; }
```

## Property Value

bool

`true` if the object is already disposed and no longer access to the members allowed. The access attempt may cause ObjectDisposedException or an unexpected behaviour.

# Event Disposed

Namespace: DotNetBrowser

Assembly: DotNetBrowser.dll

Occurs when the object has been disposed.

```
event EventHandler Disposed
```

## Returns

EventHandler

Occurs when the object has been disposed.

# Interface IAutoDisposable<TArgs>

Namespace: DotNetBrowser

Assembly: DotNetBrowser.dll

Represents the object which can be disposed by itself (without the explicit Dispose method call).

```
public interface IAutoDisposable<TArgs> : IAutoDisposable
```

## Type Parameters

`TArgs`

The type of the event arguments for the Disposed event.

**Inherited Members**

IAutoDisposable.IsDisposed

## Events

Disposed

Occurs when the object has been disposed.

# Event Disposed

Namespace: DotNetBrowser

Assembly: DotNetBrowser.dll

Occurs when the object has been disposed.

```
event EventHandler<TArgs> Disposed
```

## Returns

EventHandler<TArgs>

Occurs when the object has been disposed.

# Namespace DotNetBrowser.AvaloniaUi

## Classes

BitmapExtensions

Extensions for Bitmap class.

BrowserView

The Avalonia-based implementation of a browser view.

# Class BitmapExtensions

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

Extensions for Bitmap class.

```
public static class BitmapExtensions
```

**Inheritance**

object ← BitmapExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

ToSkBitmap(Bitmap)

Converts a Bitmap to the SkiaSharp SKBitmap.

ToSkImage(Bitmap)

Converts a Bitmap to the SkiaSharp SKImage.

ToUiBitmap(Bitmap)

Converts a Bitmap to the Avalonia UI Avalonia.Media.Imaging.IBitmap.

# Method ToSkBitmap

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## ToSkBitmap(Bitmap)

Converts a Bitmap to the SkiaSharp SKBitmap.

```
public static SKBitmap ToSkBitmap(this Bitmap bitmap)
```

## Parameters

`bitmap` Bitmap

The bitmap to convert.

## Returns

SKBitmap

The converted SKBitmap instance.

# Method ToSkImage

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## ToSkImage(Bitmap)

Converts a Bitmap to the SkiaSharp SKImage.

```
public static SKImage ToSkImage(this Bitmap bitmap)
```

## Parameters

`bitmap` Bitmap

The bitmap to convert.

## Returns

SKImage

The converted SKImage instance.

# Method ToUiBitmap

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## ToUiBitmap(Bitmap)

Converts a Bitmap to the Avalonia UI Avalonia.Media.Imaging.IBitmap.

```
public static Bitmap ToUiBitmap(this Bitmap bitmap)
```

## Parameters

`bitmap` Bitmap

The bitmap to convert.

## Returns

Bitmap

The converted Avalonia.Media.Imaging.IBitmap instance.

# Class BrowserView

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

The Avalonia-based implementation of a browser view.

```
public class BrowserView : UserControl, INotifyPropertyChanged, IDataContextProvider,
ILogical, IThemeVariantHost, IResourceHost, IResourceNode, IStyleHost, ISetLogicalParent,
ISetInheritanceParent, ISupportInitialize, IStyleable, INamed, IInputElement,
IDataTemplateHost, ISetterValue, IBrowserView
```

**Inheritance**

object ← AvaloniaObject ← Animatable ← StyledElement ← Visual ← Layoutable ← Interactive ← InputElement ← Control ← TemplatedControl ← ContentControl ← UserControl ← BrowserView

**Implements**

INotifyPropertyChanged, IDataContextProvider, ILogical, IThemeVariantHost, IResourceHost, IResourceNode, IStyleHost, ISetLogicalParent, ISetInheritanceParent, ISupportInitialize, IStyleable, INamed, IInputElement, IDataTemplateHost, ISetterValue, IBrowserView

**Inherited Members**

ContentControl.ContentProperty , ContentControl.ContentTemplateProperty , ContentControl.HorizontalContentAlignmentProperty , ContentControl.VerticalContentAlignmentProperty , ContentControl.RegisterContentPresenter(ContentPresenter) , ContentControl.Content , ContentControl.ContentTemplate , ContentControl.Presenter , ContentControl.HorizontalContentAlignment , ContentControl.VerticalContentAlignment , TemplatedControl.BackgroundProperty , TemplatedControl.BackgroundSizingProperty , TemplatedControl.BorderBrushProperty , TemplatedControl.BorderThicknessProperty , TemplatedControl.CornerRadiusProperty , TemplatedControl.FontFamilyProperty , TemplatedControl.FontFeaturesProperty , TemplatedControl.FontSizeProperty , TemplatedControl.FontStyleProperty , TemplatedControl.FontWeightProperty , TemplatedControl.FontStretchProperty , TemplatedControl.ForegroundProperty , TemplatedControl.PaddingProperty , TemplatedControl.TemplateProperty , TemplatedControl.IsTemplateFocusTargetProperty , TemplatedControl.TemplateAppliedEvent , TemplatedControl.GetIsTemplateFocusTarget(Control) , TemplatedControl.SetIsTemplateFocusTarget(Control, bool) , TemplatedControl.ApplyTemplate() , TemplatedControl.GetTemplateFocusTarget() ,

TemplatedControl.OnAttachedToLogicalTree(LogicalTreeAttachmentEventArgs) ,
TemplatedControl.OnDetachedFromLogicalTree(LogicalTreeAttachmentEventArgs) ,
TemplatedControl.OnApplyTemplate(TemplateAppliedEventArgs) ,
TemplatedControl.OnTemplateChanged(AvaloniaPropertyChangedEventArgs) ,
TemplatedControl.Background , TemplatedControl.BackgroundSizing , TemplatedControl.BorderBrush ,
TemplatedControl.BorderThickness , TemplatedControl.CornerRadius , TemplatedControl.FontFamily ,
TemplatedControl.FontFeatures , TemplatedControl.FontSize , TemplatedControl.FontStyle ,
TemplatedControl.FontWeight , TemplatedControl.FontStretch , TemplatedControl.Foreground ,
TemplatedControl.Padding , TemplatedControl.Template , TemplatedControl.TemplateApplied ,
Control.FocusAdornerProperty , Control.TagProperty , Control.ContextMenuProperty ,
Control.ContextFlyoutProperty , Control.RequestBringIntoViewEvent , Control.ContextRequestedEvent ,
Control.LoadedEvent , Control.UnloadedEvent , Control.SizeChangedEvent ,
Control.OnLoaded(RoutedEventArgs) , Control.OnUnloaded(RoutedEventArgs) ,
Control.OnSizeChanged(SizeChangedEventArgs) ,
Control.OnAttachedToVisualTreeCore(VisualTreeAttachmentEventArgs) ,
Control.OnDetachedFromVisualTreeCore(VisualTreeAttachmentEventArgs) ,
Control.OnLostFocus(RoutedEventArgs) , Control.OnCreateAutomationPeer() ,
Control.OnPointerReleased(PointerReleasedEventArgs) , Control.OnKeyUp(KeyEventArgs) ,
Control.FocusAdorner , Control.DataTemplates , Control.ContextMenu , Control.ContextFlyout ,
Control.IsLoaded , Control.Tag , Control.ContextRequested , Control.Loaded , Control.Unloaded ,
Control.SizeChanged , InputElement.FocusableProperty , InputElement.IsEnabledProperty ,
InputElement.IsEffectivelyEnabledProperty , InputElement.CursorProperty ,
InputElement.IsKeyboardFocusWithinProperty , InputElement.IsFocusedProperty ,
InputElement.IsHitTestVisibleProperty , InputElement.IsPointerOverProperty ,
InputElement.IsTabStopProperty , InputElement.GotFocusEvent , InputElement.LostFocusEvent ,
InputElement.KeyDownEvent , InputElement.KeyUpEvent , InputElement.TabIndexProperty ,
InputElement.TextInputEvent , InputElement.TextInputMethodClientRequestedEvent ,
InputElement.PointerEnteredEvent , InputElement.PointerExitedEvent ,
InputElement.PointerMovedEvent , InputElement.PointerPressedEvent ,
InputElement.PointerReleasedEvent , InputElement.PointerCaptureLostEvent ,
InputElement.PointerWheelChangedEvent , InputElement.TappedEvent , InputElement.HoldingEvent ,
InputElement.DoubleTappedEvent , InputElement.Focus(NavigationMethod, KeyModifiers) ,
InputElement.OnKeyDown(KeyEventArgs) , InputElement.OnTextInput(TextInputEventArgs) ,
InputElement.OnPointerEntered(PointerEventArgs) , InputElement.OnPointerExited(PointerEventArgs) ,
InputElement.OnPointerMoved(PointerEventArgs) ,
InputElement.OnPointerPressed(PointerPressedEventArgs) ,
InputElement.OnPointerCaptureLost(PointerCaptureLostEventArgs) ,
InputElement.OnPointerWheelChanged(PointerWheelEventArgs) ,
InputElement.UpdateIsEffectivelyEnabled() , InputElement.Focusable , InputElement.IsEnabled ,
InputElement.Cursor , InputElement.IsKeyboardFocusWithin , InputElement.IsFocused ,

InputElement.IsHitTestVisible , InputElement.IsPointerOver , InputElement.IsTabStop , InputElement.IsEffectivelyEnabled , InputElement.TabIndex , InputElement.KeyBindings , InputElement.IsEnabledCore , InputElement.GestureRecognizers , InputElement.GotFocus , InputElement.LostFocus , InputElement.KeyDown , InputElement.KeyUp , InputElement.TextInput , InputElement.TextInputMethodClientRequested , InputElement.PointerEntered , InputElement.PointerExited , InputElement.PointerMoved , InputElement.PointerPressed , InputElement.PointerReleased , InputElement.PointerCaptureLost , InputElement.PointerWheelChanged , InputElement.Tapped , InputElement.Holding , InputElement.DoubleTapped , Interactive.AddHandler(RoutedEvent, Delegate, RoutingStrategies, bool) , Interactive.AddHandler<TEventArgs>(RoutedEvent<TEventArgs>, EventHandler<TEventArgs>, RoutingStrategies, bool) , Interactive.RemoveHandler(RoutedEvent, Delegate) , Interactive.RemoveHandler<TEventArgs>(RoutedEvent<TEventArgs>, EventHandler<TEventArgs>) , Interactive.RaiseEvent(RoutedEventArgs) , Interactive.BuildEventRoute(RoutedEvent) , Layoutable.DesiredSizeProperty , Layoutable.WidthProperty , Layoutable.HeightProperty , Layoutable.MinWidthProperty , Layoutable.MaxWidthProperty , Layoutable.MinHeightProperty , Layoutable.MaxHeightProperty , Layoutable.MarginProperty , Layoutable.HorizontalAlignmentProperty , Layoutable.VerticalAlignmentProperty , Layoutable.UseLayoutRoundingProperty , Layoutable.UpdateLayout() , Layoutable.Measure(Size) , Layoutable.Arrange(Rect) , Layoutable.InvalidateMeasure() , Layoutable.InvalidateArrange() , Layoutable.AffectsMeasure<T>(params AvaloniaProperty[]) , Layoutable.AffectsArrange<T>(params AvaloniaProperty[]) , Layoutable.MeasureCore(Size) , Layoutable.MeasureOverride(Size) , Layoutable.ArrangeCore(Rect) , Layoutable.ArrangeOverride(Size) , Layoutable.OnMeasureInvalidated() , Layoutable.OnVisualParentChanged(Visual, Visual) , Layoutable.Width , Layoutable.Height , Layoutable.MinWidth , Layoutable.MaxWidth , Layoutable.MinHeight , Layoutable.MaxHeight , Layoutable.Margin , Layoutable.HorizontalAlignment , Layoutable.VerticalAlignment , Layoutable.DesiredSize , Layoutable.IsMeasureValid , Layoutable.IsArrangeValid , Layoutable.UseLayoutRounding , Layoutable.EffectiveViewportChanged , Layoutable.LayoutUpdated , Visual.BoundsProperty , Visual.ClipToBoundsProperty , Visual.ClipProperty , Visual.IsVisibleProperty , Visual.OpacityProperty , Visual.OpacityMaskProperty , Visual.EffectProperty , Visual.HasMirrorTransformProperty , Visual.RenderTransformProperty , Visual.RenderTransformOriginProperty , Visual.FlowDirectionProperty , Visual.VisualParentProperty , Visual.ZIndexProperty , Visual.GetFlowDirection(Visual) , Visual.SetFlowDirection(Visual, FlowDirection) , Visual.InvalidateVisual() , Visual.Render(DrawingContext) , Visual.AffectsRender<T>(params AvaloniaProperty[]) , Visual.LogicalChildrenCollectionChanged(object, NotifyCollectionChangedEventArgs) , Visual.OnAttachedToVisualTree(VisualTreeAttachmentEventArgs) , Visual.OnDetachedFromVisualTree(VisualTreeAttachmentEventArgs) , Visual.InvalidateMirrorTransform() , Visual.Bounds , Visual.ClipToBounds , Visual.Clip , Visual.IsEffectivelyVisible , Visual.IsVisible , Visual.Opacity , Visual.OpacityMask , Visual.Effect ,

Visual.HasMirrorTransform , Visual.RenderTransform , Visual.RenderTransformOrigin , Visual.FlowDirection , Visual.ZIndex , Visual.VisualChildren , Visual.VisualRoot , Visual.BypassFlowDirectionPolicies , Visual.AttachedToVisualTree , Visual.DetachedFromVisualTree , StyledElement.DataContextProperty , StyledElement.NameProperty , StyledElement.ParentProperty , StyledElement.TemplatedParentProperty , StyledElement.ThemeProperty , StyledElement.BeginInit() , StyledElement.EndInit() , StyledElement.ApplyStyling() , StyledElement.InitializeIfNeeded() , StyledElement.TryGetResource(object, ThemeVariant, out object) , StyledElement.OnDataContextChanged(EventArgs) , StyledElement.OnDataContextBeginUpdate() , StyledElement.OnDataContextEndUpdate() , StyledElement.OnInitialized() , StyledElement.Name , StyledElement.Classes , StyledElement.DataContext , StyledElement.IsInitialized , StyledElement.Styles , StyledElement.StyleKey , StyledElement.Resources , StyledElement.TemplatedParent , StyledElement.Theme , StyledElement.LogicalChildren , StyledElement.PseudoClasses , StyledElement.StyleKeyOverride , StyledElement.Parent , StyledElement.ActualThemeVariant , StyledElement.AttachedToLogicalTree , StyledElement.DetachedFromLogicalTree , StyledElement.DataContextChanged , StyledElement.Initialized , StyledElement.ResourcesChanged , StyledElement.ActualThemeVariantChanged , Animatable.TransitionsProperty , Animatable.OnPropertyChangedCore(AvaloniaPropertyChangedEventArgs) , Animatable.Transitions , AvaloniaObject.CheckAccess() , AvaloniaObject.VerifyAccess() , AvaloniaObject.ClearValue(AvaloniaProperty) , AvaloniaObject.ClearValue<T>(AvaloniaProperty<T>) , AvaloniaObject.ClearValue<T>(StyledProperty<T>) , AvaloniaObject.ClearValue<T>(DirectPropertyBase<T>) , AvaloniaObject.Equals(object) , AvaloniaObject.GetHashCode() , AvaloniaObject.GetValue(AvaloniaProperty) , AvaloniaObject.GetValue<T>(StyledProperty<T>) , AvaloniaObject.GetValue<T>(DirectPropertyBase<T>) , AvaloniaObject.GetBaseValue<T>(StyledProperty<T>) , AvaloniaObject.IsAnimating(AvaloniaProperty) , AvaloniaObject.IsSet(AvaloniaProperty) , AvaloniaObject.SetValue(AvaloniaProperty, object, BindingPriority) , AvaloniaObject.SetValue<T>(StyledProperty<T>, T, BindingPriority) , AvaloniaObject.SetValue<T>(DirectPropertyBase<T>, T) , AvaloniaObject.SetCurrentValue(AvaloniaProperty, object) , AvaloniaObject.SetCurrentValue<T>(StyledProperty<T>, T) , AvaloniaObject.Bind(AvaloniaProperty, IBinding) , AvaloniaObject.Bind(AvaloniaProperty, IObservable<object>, BindingPriority) , AvaloniaObject.Bind<T>(StyledProperty<T>, IObservable<object>, BindingPriority) , AvaloniaObject.Bind<T>(StyledProperty<T>, IObservable<T>, BindingPriority) , AvaloniaObject.Bind<T>(StyledProperty<T>, IObservable<BindingValue<T>>, BindingPriority) , AvaloniaObject.Bind<T>(DirectPropertyBase<T>, IObservable<object>) , AvaloniaObject.Bind<T>(DirectPropertyBase<T>, IObservable<T>) , AvaloniaObject.Bind<T>(DirectPropertyBase<T>, IObservable<BindingValue<T>>) , AvaloniaObject.CoerceValue(AvaloniaProperty) , AvaloniaObject.UpdateDataValidation(AvaloniaProperty, BindingValueType, Exception) ,

AvaloniaObject.RaisePropertyChanged<T>(DirectPropertyBase<T>, T, T) , AvaloniaObject.SetAndRaise<T>(DirectPropertyBase<T>, ref T, T) , AvaloniaObject.InheritanceParent , AvaloniaObject.this[AvaloniaProperty] , AvaloniaObject.this[IndexerDescriptor] , AvaloniaObject.PropertyChanged , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

**Extension Methods**
BrowserViewExtensions.InitializeFrom(IBrowserView, IBrowser)

## Remarks

To bind an instance of this view to a particular browser, use its InitializeFrom(IBrowserView, IBrowser) method:

```
browserView.InitializeFrom(browser);
```

This method should be called from the UI thread.

## Constructors

BrowserView()

## Properties

CreatePopupHandler
: This handler will be used when browser decides whether a new popup instance can be created or not, if a different handler was not configured for the browser.

MenuItems
: The collection of the current context menu items.

OpenExtensionActionPopupHandler
: This handler is used for handling the popups that are requested to be shown by the extension action, if a different handler was not configured for the browser.

OpenPopupHandler
: This handler is used when a new popup browser instance should be opened, if a different handler was not configured for the browser.

SaveCreditCardHandler

This handler is used when the user is prompted to save the credit cards in the credit card store, if a different handler was not configured for the browser.

SavePasswordHandler
This handler is used when the user is prompted to save the credentials in the password store, if a different handler was not configured for the browser.

SaveUserDataProfileHandler
This handler is used when the user is prompted to save the user data profiles in the user data profile store, if a different handler was not configured for the browser.

SelectCertificateHandler
This handler is used when the client SSL certificate should be selected, if a different handler was not configured for the browser.

ShowContextMenuHandler
The default context menu handler implementation for this browser view.

StartDownloadHandler
This handler is used when the browser is about to start downloading the file, if a different handler was not configured for the browser.

UpdatePasswordHandler
This handler is used when the user is prompted to update the password in the password store, if a different handler was not configured for the browser.

UpdateUserDataProfileHandler
This handler is used when the user is prompted to update the user data profile in the user data profile store, if a different handler was not configured for the browser.

## Methods

InitializeComponent(bool)
Wires up the controls and optionally loads XAML markup and attaches dev tools (if Avalonia.Diagnostics package is referenced).

OnFocusRequested(object, EventArgs)
Event handler for FocusRequested event.

OnGotFocus(GotFocusEventArgs)
Called before the Avalonia.Input.InputElement.GotFocus event occurs.

OnPropertyChanged(AvaloniaPropertyChangedEventArgs)

Called when a avalonia property changes on the object.

# Constructor BrowserView

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## BrowserView()

```
public BrowserView()
```

# Property CreatePopupHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## CreatePopupHandler

This handler will be used when browser decides whether a new popup instance can be created or not, if a different handler was not configured for the browser.

```
public IHandler<CreatePopupParameters, CreatePopupResponse> CreatePopupHandler { get; }
```

## Property Value

IHandler<CreatePopupParameters, CreatePopupResponse>

# Property MenuItems

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## MenuItems

The collection of the current context menu items.

```
public ObservableCollection<MenuItemViewModel> MenuItems { get; }
```

## Property Value

ObservableCollection<MenuItemViewModel>

# Property OpenExtensionActionPopupHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## OpenExtensionActionPopupHandler

This handler is used for handling the popups that are requested to be shown by the extension action, if a different handler was not configured for the browser.

```
public IHandler<OpenExtensionActionPopupParameters> OpenExtensionActionPopupHandler {
get; }
```

## Property Value

IHandler<OpenExtensionActionPopupParameters>

# Property OpenPopupHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## OpenPopupHandler

This handler is used when a new popup browser instance should be opened, if a different handler was not configured for the browser.

```
public IHandler<OpenPopupParameters> OpenPopupHandler { get; }
```

## Property Value

IHandler<OpenPopupParameters>

## Remarks

**Important:** the engine will be blocked until you return control from the callback.

# Property SaveCreditCardHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## SaveCreditCardHandler

This handler is used when the user is prompted to save the credit cards in the credit card store, if a different handler was not configured for the browser.

```
public IHandler<SaveCreditCardParameters, SaveCreditCardResponse> SaveCreditCardHandler {
get; }
```

## Property Value

IHandler<SaveCreditCardParameters, SaveCreditCardResponse>

# Property SavePasswordHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## SavePasswordHandler

This handler is used when the user is prompted to save the credentials in the password store, if a different handler was not configured for the browser.

```
public IHandler<SavePasswordParameters, SavePasswordResponse> SavePasswordHandler { get; }
```

## Property Value

IHandler<SavePasswordParameters, SavePasswordResponse>

# Property SaveUserDataProfileHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## SaveUserDataProfileHandler

This handler is used when the user is prompted to save the user data profiles in the user data profile store, if a different handler was not configured for the browser.

```
public IHandler<SaveUserDataProfileParameters, SaveUserDataProfileResponse>
SaveUserDataProfileHandler { get; }
```

## Property Value

IHandler<SaveUserDataProfileParameters, SaveUserDataProfileResponse>

# Property SelectCertificateHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## SelectCertificateHandler

This handler is used when the client SSL certificate should be selected, if a different handler was not configured for the browser.

```
public IHandler<SelectCertificateParameters, SelectCertificateResponse>
SelectCertificateHandler { get; }
```

## Property Value

IHandler<SelectCertificateParameters, SelectCertificateResponse>

# Property ShowContextMenuHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## ShowContextMenuHandler

The default context menu handler implementation for this browser view.

```
public IHandler<ShowContextMenuParameters, ShowContextMenuResponse> ShowContextMenuHandler
{ get; set; }
```

## Property Value

IHandler<ShowContextMenuParameters, ShowContextMenuResponse>

# Property StartDownloadHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## StartDownloadHandler

This handler is used when the browser is about to start downloading the file, if a different handler was not configured for the browser.

```
public IHandler<StartDownloadParameters, StartDownloadResponse> StartDownloadHandler {
get; }
```

## Property Value

IHandler<StartDownloadParameters, StartDownloadResponse>

# Property UpdatePasswordHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## UpdatePasswordHandler

This handler is used when the user is prompted to update the password in the password store, if a different handler was not configured for the browser.

```
public IHandler<UpdatePasswordParameters, UpdatePasswordResponse> UpdatePasswordHandler {
get; }
```

## Property Value

IHandler<UpdatePasswordParameters, UpdatePasswordResponse>

# Property UpdateUserDataProfileHandler

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## UpdateUserDataProfileHandler

This handler is used when the user is prompted to update the user data profile in the user data profile store, if a different handler was not configured for the browser.

```
public IHandler<UpdateUserDataProfileParameters, UpdateUserDataProfileResponse>
UpdateUserDataProfileHandler { get; }
```

## Property Value

IHandler<UpdateUserDataProfileParameters, UpdateUserDataProfileResponse>

# Method InitializeComponent

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## InitializeComponent(bool)

Wires up the controls and optionally loads XAML markup and attaches dev tools (if Avalonia.Diagnostics package is referenced).

```
public void InitializeComponent(bool loadXaml = true)
```

## Parameters

`loadXaml` bool

Should the XAML be loaded into the component.

# Method OnFocusRequested

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## OnFocusRequested(object, EventArgs)

Event handler for FocusRequested event.

```
public void OnFocusRequested(object sender, EventArgs e)
```

## Parameters

`sender` object

the sender of the event

`e` EventArgs

the event args

# Method OnGotFocus

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## OnGotFocus(GotFocusEventArgs)

Called before the Avalonia.Input.InputElement.GotFocus event occurs.

```
protected override void OnGotFocus(GotFocusEventArgs e)
```

## Parameters

`e` GotFocusEventArgs

The event args.

# Method OnPropertyChanged

Namespace: DotNetBrowser.AvaloniaUi

Assembly: DotNetBrowser.AvaloniaUi.dll

## OnPropertyChanged(AvaloniaPropertyChangedEventArgs)

Called when a avalonia property changes on the object.

```
protected override void OnPropertyChanged(AvaloniaPropertyChangedEventArgs change)
```

## Parameters

`change` AvaloniaPropertyChangedEventArgs

The property change details.

# Namespace DotNetBrowser.AvaloniaUi.Dialogs

## Classes

DefaultAuthenticationHandler

Default Avalonia UI authentication handler, which displays an authentication dialog.

DialogHandlerBase

The base class for all dialog handlers.

# Class DefaultAuthenticationHandler

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

Default Avalonia UI authentication handler, which displays an authentication dialog.

```
public class DefaultAuthenticationHandler : DialogHandlerBase,
IHandler<AuthenticateParameters, AuthenticateResponse>
```

**Inheritance**

object ← DialogHandlerBase ← DefaultAuthenticationHandler

**Implements**

IHandler<AuthenticateParameters, AuthenticateResponse>

**Inherited Members**

DialogHandlerBase.Parent , DialogHandlerBase.FocusParent() , DialogHandlerBase.GetStorageProvider() , DialogHandlerBase.UnfocusFocusBrowser(IBrowser) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

DefaultAuthenticationHandler(Visual)

Creates and initializes the dialog handler.

## Methods

Handle(AuthenticateParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

# Constructor DefaultAuthenticationHandler

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

## DefaultAuthenticationHandler(Visual)

Creates and initializes the dialog handler.

```
public DefaultAuthenticationHandler(Visual parent)
```

## Parameters

`parent` Visual

The parent object for the dialog displayed by this handler.

# Method Handle

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

## Handle(AuthenticateParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

```
public AuthenticateResponse Handle(AuthenticateParameters parameters)
```

### Parameters

`parameters` AuthenticateParameters

the handler parameters.

### Returns

AuthenticateResponse

an object that represents the response that should be determined based on the provided parameters.

# Class DialogHandlerBase

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

The base class for all dialog handlers.

```
public class DialogHandlerBase
```

**Inheritance**

object ← DialogHandlerBase

**Derived**

DefaultAuthenticationHandler

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

DialogHandlerBase(Visual)

Creates and initializes the dialog handler.

## Properties

Parent

The parent object for the dialog displayed by this handler. This object is used to locate the owner window for the modal dialogs.

## Methods

FocusParent()

Tells the parent that it should request focus.

GetStorageProvider()

Gets the Avalonia.Platform.Storage.IStorageProvider that provides the file picker API.

UnfocusFocusBrowser(IBrowser)

Tells the Browser instance to unfocus and focus. This method is used as a workaround because the default implementation of unfocus and focus functionality (in the WindowedView class) does not give an expected keyboard focus result.

# Constructor DialogHandlerBase

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

## DialogHandlerBase(Visual)

Creates and initializes the dialog handler.

```
public DialogHandlerBase(Visual parent)
```

## Parameters

`parent` Visual

The parent object for the dialog displayed by this handler.

# Property Parent

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

## Parent

The parent object for the dialog displayed by this handler. This object is used to locate the owner window for the modal dialogs.

```
public Visual Parent { get; }
```

## Property Value

Visual

# Method FocusParent

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

## FocusParent()

Tells the parent that it should request focus.

```
protected void FocusParent()
```

# Method GetStorageProvider

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

## GetStorageProvider()

Gets the Avalonia.Platform.Storage.IStorageProvider that provides the file picker API.

```
protected IStorageProvider GetStorageProvider()
```

## Returns

IStorageProvider

The Avalonia.Platform.Storage.IStorageProvider instance.

# Method UnfocusFocusBrowser

Namespace: DotNetBrowser.AvaloniaUi.Dialogs

Assembly: DotNetBrowser.AvaloniaUi.dll

## UnfocusFocusBrowser(IBrowser)

Tells the Browser instance to unfocus and focus. This method is used as a workaround because the default implementation of unfocus and focus functionality (in the WindowedView class) does not give an expected keyboard focus result.

```
protected void UnfocusFocusBrowser(IBrowser browser)
```

## Parameters

`browser` IBrowser

# Namespace DotNetBrowser.Avalonia Ui.Extensions

## Classes

DefaultOpenExtensionActionPopupHandler

DefaultOpenExtensionPopupHandler

Default extension popup handler, which displays an extension popup as a separate window.

# Class DefaultOpenExtensionActionPopupHandler

Namespace: DotNetBrowser.AvaloniaUi.Extensions

Assembly: DotNetBrowser.AvaloniaUi.dll

```
public class DefaultOpenExtensionActionPopupHandler :
IHandler<OpenExtensionActionPopupParameters>
```

### Inheritance

object ← DefaultOpenExtensionActionPopupHandler

### Implements

IHandler<OpenExtensionActionPopupParameters>

### Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

DefaultOpenExtensionActionPopupHandler(Visual)

## Methods

Handle(OpenExtensionActionPopupParameters)

This method is called when the Chromium callback should be handled synchronously. In a number of cases, the Chromium engine will be blocked until this method returns.

# Constructor DefaultOpenExtensionActionPopupHandler

Namespace: DotNetBrowser.AvaloniaUi.Extensions

Assembly: DotNetBrowser.AvaloniaUi.dll

## DefaultOpenExtensionActionPopupHandler(Visual)

```
public DefaultOpenExtensionActionPopupHandler(Visual parent)
```

### Parameters

`parent` Visual

# Method Handle

Namespace: DotNetBrowser.AvaloniaUi.Extensions

Assembly: DotNetBrowser.AvaloniaUi.dll

## Handle(OpenExtensionActionPopupParameters)

This method is called when the Chromium callback should be handled synchronously. In a number of cases, the Chromium engine will be blocked until this method returns.

```
public void Handle(OpenExtensionActionPopupParameters parameters)
```

## Parameters

`parameters` OpenExtensionActionPopupParameters

The handler parameters.

# Class DefaultOpenExtensionPopupHandler

Namespace: DotNetBrowser.AvaloniaUi.Extensions

Assembly: DotNetBrowser.AvaloniaUi.dll

Default extension popup handler, which displays an extension popup as a separate window.

```
public sealed class DefaultOpenExtensionPopupHandler :
IHandler<OpenExtensionPopupParameters>
```

**Inheritance**

object ← DefaultOpenExtensionPopupHandler

**Implements**

IHandler<OpenExtensionPopupParameters>

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

DefaultOpenExtensionPopupHandler(Visual)

Creates and initializes a new instance of DefaultOpenExtensionPopupHandler.

## Methods

Handle(OpenExtensionPopupParameters)

This method is called when Chromium is about to display the extension popup window.

# Constructor DefaultOpenExtensionPopupHandler

Namespace: DotNetBrowser.AvaloniaUi.Extensions

Assembly: DotNetBrowser.AvaloniaUi.dll

## DefaultOpenExtensionPopupHandler(Visual)

Creates and initializes a new instance of DefaultOpenExtensionPopupHandler.

```
public DefaultOpenExtensionPopupHandler(Visual parent)
```

## Parameters

`parent` Visual

The parent object for the window displayed by this handler.

# Method Handle

Namespace: DotNetBrowser.AvaloniaUi.Extensions

Assembly: DotNetBrowser.AvaloniaUi.dll

## Handle(OpenExtensionPopupParameters)

This method is called when Chromium is about to display the extension popup window.

```
public void Handle(OpenExtensionPopupParameters parameters)
```

## Parameters

`parameters` OpenExtensionPopupParameters

The extension popup parameters.

# Namespace DotNetBrowser.Browser

## Classes

BitmapTimeoutException
Thrown when a bitmap request times out.

BrowserDisposeOptions
The dispose options of the browser.

BrowserViewExtensions
Extension methods for IBrowserView interface.

CspParameters
The parameters describing the particular key container within a particular cryptographic service provider (CSP).

## Interfaces

IBrowser
A web browser control that allows loading a web page or a local HTML file, accessing DOM and executing JavaScript on the loaded web page, getting notifications about loading progress, dispatching keyboard and mouse events, etc.

IBrowserSettings
The settings of the browser.

IBrowserView
The interface that is implemented by UI components which are able to display the browser.

IOffScreenRenderProvider
Provides access to an image rendered by the browser that works in the off-screen rendering mode.

## Enums

PreferredColorScheme
The preferred color scheme for the web content in the browser.

SavePageType
Determines how the web page will be saved.

WebRtcIpHandlingPolicy

The media performance/privacy tradeoffs which affect how WebRTC traffic will be routed and how much local address information will be exposed.

# Class BitmapTimeoutException

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Thrown when a bitmap request times out.

```
public class BitmapTimeoutException : InvalidOperationException, ISerializable
```

**Inheritance**

object ← Exception ← SystemException ← InvalidOperationException ← BitmapTimeoutException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

BitmapTimeoutException()

Constructs a new instance of BitmapTimeoutException.

# Constructor BitmapTimeoutException

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## BitmapTimeoutException()

Constructs a new instance of BitmapTimeoutException.

```
public BitmapTimeoutException()
```

# Class BrowserDisposeOptions

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

The dispose options of the browser.

```
public sealed class BrowserDisposeOptions
```

**Inheritance**

object ← BrowserDisposeOptions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

BeforeUnloadEventHandled

Specifies whether the `beforeunload` and `unload` JavaScript events should be handled if they are present on the web page loaded into the browser instance that is about to be disposed.

# Property BeforeUnloadEventHandled

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## BeforeUnloadEventHandled

Specifies whether the `beforeunload` and `unload` JavaScript events should be handled if they are present on the web page loaded into the browser instance that is about to be disposed.

```
public bool BeforeUnloadEventHandled { get; set; }
```

## Property Value

bool

`true` if the registered on the web page `beforeunload` and `unload` JavaScript events should be handled when disposing a browser instance, `false` otherwise.

# Class BrowserViewExtensions

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Extension methods for IBrowserView interface.

```
public static class BrowserViewExtensions
```

**Inheritance**

object ← BrowserViewExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

InitializeFrom(IBrowserView, IBrowser)

Initialize a browser view from this particular IBrowser instance. If there is another browser view bound to this browser instance, that view will be de-initialized.

# Method InitializeFrom

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## InitializeFrom(IBrowserView, IBrowser)

Initialize a browser view from this particular IBrowser instance. If there is another browser view bound to this browser instance, that view will be de-initialized.

```
public static void InitializeFrom(this IBrowserView browserView, IBrowser browser)
```

### Parameters

`browserView` IBrowserView

the browser view to initialize.

`browser` IBrowser

the browser instance to initialize from.

# Class CspParameters

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

The parameters describing the particular key container within a particular cryptographic service provider (CSP).

```
public sealed class CspParameters
```

**Inheritance**

object ← CspParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

KeyContainerName

Gets or sets the key container name.

Pin

Gets or sets the PIN associated with a smart card key.

ProviderName

Gets or sets the cryptographic service provider (CSP) name.

ProviderType

Gets or sets the provider type.

# Property KeyContainerName

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## KeyContainerName

Gets or sets the key container name.

```
public string KeyContainerName { get; set; }
```

## Property Value

string

The key container name.

# Property Pin

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Pin

Gets or sets the PIN associated with a smart card key.

```
public string Pin { get; set; }
```

## Property Value

string

The PIN associated with a smart card key.

## Remarks

Use this property to supply a password for a smart card key. When you specify a password using this property, a password dialog box will not be presented to the user.

# Property ProviderName

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## ProviderName

Gets or sets the cryptographic service provider (CSP) name.

```
public string ProviderName { get; set; }
```

## Property Value

string

The cryptographic service provider (CSP) name.

# Property ProviderType

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## ProviderType

Gets or sets the provider type.

```
public int ProviderType { get; set; }
```

## Property Value

int

The provider type.There are currently a number of predefined provider types, and a custom CSP writer can define a new provider type. The defined provider types are discussed in https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptographic-provider-types

# Interface IBrowser

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

A web browser control that allows loading a web page or a local HTML file, accessing DOM and executing JavaScript on the loaded web page, getting notifications about loading progress, dispatching keyboard and mouse events, etc.

```
public interface IBrowser : IDisposable, IAutoDisposable
```

**Inherited Members**
IDisposable.Dispose() , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Remarks

The `IBrowser` instance itself is running in a separate native process that allocates memory and system resources that must be released. So, when a `IBrowser` instance is no longer needed, it must be disposed through the Dispose() method to release all the allocated memory and system resources. For example:

```
IBrowser browser = engine.CreateBrowser();
...
browser.Dispose();
```

Any attempt to use an already disposed `IBrowser` instance will lead to the ObjectDisposedException.

The `IBrowser` instance is disposed automatically when its `IEngine` is disposed or unexpectedly crashed. If the instance represents a popup window created by JavaScript via the `window.open()` function, then JavaScript can close the instance using the `window.close()` function.

To get notifications that the `IBrowser` instance has been disposed please subscribe to the following event:

```
browser.Disposed += (s, e) => {
    // The Browser instance has been disposed.
};
```

## Properties

AllFrames

Gets all the frames on the currently loaded web page.

Audio

Gets the IAudio instance that allows controlling audio on the loaded web page and receive notifications when audio has been started or stopped playing.

Capture

Gets the ICapture instance that can be used for listening and handling capture sessions.

Cast

Gets the ICast instance that can be used for casting media on receivers.

ConvertJsNameHandler

Gets or sets a handler that is used for the JavaScript name converting. Implement this handler to have control of how the names are converted when binding or executing JavaScript.

CreatePopupHandler

Gets or sets a handler that is used when the browser decides whether a new popup instance can be created or not.

CreditCards

Gets the ICreditCards instance that can be used for working with credit cards saved in the Chromium credit card store.

DevTools

Gets the IDevTools instance that allows working with Chromium Developer Tools for this browser.

Dialogs

Gets the IDialogs instance that can be used for configuring the dialogs that can be shown by browser.

DragAndDrop

Gets the IDragAndDrop instance that can be used for managing drag&drop operations for the browser.

Engine

Gets the IEngine instance associated with this browser.

Favicon

Gets the favicon of the currently loaded web page.

FocusedFrame

Gets the focused frame on the currently loaded web page.

FullScreen

Gets the IFullScreen instance that can be used for controlling the browser's fullscreen mode.

InjectCssHandler

Gets or sets a handler that is used when the document element has been created and a custom stylesheet can be injected into the document.

InjectJsHandler

Gets or sets a handler that is used when the document element has been created and a custom JavaScript can be injected into the document.

JsDialogs

Gets the IJsDialogs instance that can be used for configuring the JavaScript dialogs that can be shown by browser.

Keyboard

Gets the IKeyboard instance that can be used for listening and simulating keyboard input events.

MainFrame

The main frame on the currently loaded web page, if it exists.

Mouse

Gets the IMouse instance that can be used for listening and simulating mouse input events.

Navigation

Gets the INavigation instance that can be used for controlling navigation in the current browser instance.

OpenExtensionActionPopupHandler

Gets or sets a handler that is used for handling the popups that are requested to be shown by the extension action.

OpenPopupHandler

Gets or sets a handler that is used when a new popup browser instance should be opened.

Passwords

Gets the IPasswords instance that can be used for working with logins and passwords saved in the Chromium password store.

PrintHtmlContentHandler

Gets or sets a handler that is used when the HTML content printing is initiated.

PrintPdfContentHandler

Gets or sets a handler that is used when the PDF content printing is initiated.

Profile

Gets the IProfile instance associated with this browser.

RequestPdfDocumentPasswordHandler

Gets or sets a handler that is used when the frame requests the password for an encrypted PDF document.

RequestPrintHandler

Gets or sets a handler that is used when the printing is initiated.

SelectCertificateHandler

Gets or sets a handler that is used when the web server requires authorization via client certificate.

Settings

Gets the IBrowserSettings instance that can be used for modifying the settings of the browser.

ShowContextMenuHandler

Gets or sets a handler that is used when the browser should show a context menu.

Size

Gets or sets the browser size.

StartDownloadHandler

Gets or sets a handler that is used when the browser is about to start downloading the file.

TextFinder

Gets the ITextFinder instance that can be used for finding text on a web page loaded in the browser.

Title

Gets the title of the currently loaded web page.

Touch

Gets the ITouch instance that can be used for listening touch input events.

Url

Gets the URL of the currently loaded web page or an empty string if the browser hasn't loaded any web page yet.

UserAgent

Gets or sets the `user-agent` of the current browser instance.

UserDataProfiles

Gets the IUserDataProfiles instance that can be used for working with user data profiles saved in the Chromium user data store.

Zoom

Gets the IZoom instance that can be used for zooming content of a web page loaded in the current browser instance.

## Methods

DisposeAsync(BrowserDisposeOptions)

Asynchronously disposes the current browser instance according to the given options.

Focus()

Tells the browser that it has focus and must be activated.

ReplaceMisspelledWord(string)

Replaces misspelled word under cursor on the currently loaded web page with the given `word`. If there is no misspelled word under cursor, this method does nothing.

SaveWebPage(string, string, SavePageType)

Initiates the saving process of the currently loaded web page. The web page can be saved as a single HTML file or the file with resources. Before saving a web page make sure that it is not being loaded. It is recommended to completely loaded the web page and only then save it.

SetPrivateKeyProviderPin(CspParameters)

Sets the key exchange PIN for the particular cryptographic service provider (CSP) and the particular key container in it. This functionality can be used to set the PIN that is requested when trying to use a client certificate stored on the smart card.

TakeImage()

Creates and returns an image of the currently loaded web page.

Unfocus()

Tells the browser that it does not have focus and must be deactivated.

## Events

BrowserBecameResponsive

Occurs when the browser instance has become responsive.

BrowserBecameUnresponsive

Occurs when the browser instance has become unresponsive.

ConsoleMessageReceived
Occurs when the message was added to the console.

FaviconChanged
Occurs when the web page favicon has changed.

FocusGained
Occurs when the browser instance has gained the focus.

FocusLost
Occurs when the browser instance has lost the focus.

FocusRequested
Occurs when JavaScript sends a request to focus the Browser instance by calling the `window.focus()` method.

FrameCreated
Occurs when the IFrame has been created.

FrameDeleted
Occurs when the IFrame has been deleted.

MediaStreamCaptureStarted
Occurs when the web page has started capturing an audio or video stream.

MediaStreamCaptureStopped
Occurs when the web page has stopped capturing an audio or video stream.

PrintPreviewClosed
Occurs when print preview dialog is closed.

PrintPreviewOpened
Occurs when print preview dialog is opened.

RenderProcessTerminated
Occurs when a render process has been terminated.

SpellCheckCompleted
Occurs when spell checking on the frame has been completed.

StatusChanged
Occurs when the status text has been changed.

TitleChanged
Occurs when the web page title has been changed.

UpdateBoundsRequested

Occurs when JavaScript requests to update bounds of the Browser instance.

# Property AllFrames

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## AllFrames

Gets all the frames on the currently loaded web page.

```
IEnumerable<IFrame> AllFrames { get; }
```

## Property Value

IEnumerable<IFrame>

All the frames on the currently loaded web page. If the current browser did not load any web pages, this will be an empty collection.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Audio

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Audio

Gets the IAudio instance that allows controlling audio on the loaded web page and receive notifications when audio has been started or stopped playing.

```
IAudio Audio { get; }
```

### Property Value

IAudio

The IAudio instance associated with this IBrowser.

# Property Capture

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Capture

Gets the ICapture instance that can be used for listening and handling capture sessions.

```
ICapture Capture { get; }
```

## Property Value

ICapture

The ICapture instance associated with this IBrowser.

# Property Cast

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Cast

Gets the ICast instance that can be used for casting media on receivers.

```
ICast Cast { get; }
```

## Property Value

ICast

The ICast instance associated with this IBrowser.

# Property ConvertJsNameHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## ConvertJsNameHandler

Gets or sets a handler that is used for the JavaScript name converting. Implement this handler to have control of how the names are converted when binding or executing JavaScript.

```
IHandler<ConvertJsNameParameters, ConvertJsNameResponse> ConvertJsNameHandler { get; set; }
```

## Property Value

IHandler<ConvertJsNameParameters, ConvertJsNameResponse>

## Remarks

The handler should be configured before registering .NET objects on the JavaScript side.

The ConvertJsNameParameters object contains information about the MemberInfo of the .NET object.

Use the NoConversion method to use the member name as is.

Use the CamelCase method to convert the first letter of the name to the lower case.

Use the PascalCase method to convert the first letter of the name to the upper case.

Use the ConvertTo(string) method for customizing the name.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property CreatePopupHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## CreatePopupHandler

Gets or sets a handler that is used when the browser decides whether a new popup instance can be created or not.

```
IHandler<CreatePopupParameters, CreatePopupResponse> CreatePopupHandler { get; set; }
```

## Property Value

IHandler<CreatePopupParameters, CreatePopupResponse>

## Remarks

Use the Create() method to let the browser to create a new popup.

Use the Suppress() method to suppress popup.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Suppress() will be used.

**Important:** the engine will be blocked until you return control from the callback.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property CreditCards

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## CreditCards

Gets the ICreditCards instance that can be used for working with credit cards saved in the Chromium credit card store.

```
ICreditCards CreditCards { get; }
```

## Property Value

ICreditCards

The ICreditCards instance associated with this IBrowser.

# Property DevTools

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## DevTools

Gets the IDevTools instance that allows working with Chromium Developer Tools for this browser.

```
IDevTools DevTools { get; }
```

## Property Value

IDevTools

The IDevTools instance associated with this IBrowser.

# Property Dialogs

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Dialogs

Gets the IDialogs instance that can be used for configuring the dialogs that can be shown by browser.

```
IDialogs Dialogs { get; }
```

## Property Value

IDialogs

The IDialogs instance associated with this IBrowser.

# Property DragAndDrop

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## DragAndDrop

Gets the IDragAndDrop instance that can be used for managing drag&drop operations for the browser.

```
IDragAndDrop DragAndDrop { get; }
```

## Property Value

IDragAndDrop

The IDragAndDrop instance associated with this IBrowser.

# Property Engine

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this browser.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this IBrowser.

# Property Favicon

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Favicon

Gets the favicon of the currently loaded web page.

```
Bitmap Favicon { get; }
```

## Property Value

Bitmap

The favicon of the currently loaded web page. Will return a default Chromium favicon if the browser did not load any web pages or loaded web page URL is invalid.

## Remarks

Maximum favicon size is 16x16. If the actual favicon size is bigger, then it will be resized to fit this constraint.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property FocusedFrame

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## FocusedFrame

Gets the focused frame on the currently loaded web page.

```
IFrame FocusedFrame { get; }
```

## Property Value

IFrame

The focused frame on the currently loaded web page. Can be null if there is no focused frame.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property FullScreen

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## FullScreen

Gets the IFullScreen instance that can be used for controlling the browser's fullscreen mode.

```
IFullScreen FullScreen { get; }
```

## Property Value

IFullScreen

The IFullScreen instance associated with this IBrowser.

# Property InjectCssHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## InjectCssHandler

Gets or sets a handler that is used when the document element has been created and a custom stylesheet can be injected into the document.

```
IHandler<InjectCssParameters, InjectCssResponse> InjectCssHandler { get; set; }
```

## Property Value

IHandler<InjectCssParameters, InjectCssResponse>

## Remarks

Use the Inject(string) method to inject a custom stylesheet into the document.

Use the Proceed() method to continue loading without injecting.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Proceed() will be used.

**Important:** the engine will be blocked until you return control from the callback.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property InjectJsHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## InjectJsHandler

Gets or sets a handler that is used when the document element has been created and a custom JavaScript can be injected into the document.

```
IHandler<InjectJsParameters> InjectJsHandler { get; set; }
```

## Property Value

IHandler<InjectJsParameters>

## Remarks

You can inject a custom JavaScript code using the ExecuteJavaScript(string, bool) method.This callback is intended to give you an opportunity to inject a .NET object into JavaScript code or inject a custom JavaScript code for further execution before any scripts are executed in the particular frame. For example:

```
Browser.InjectJsHandler = new Handler<InjectJsParameters>(args =>
    {
        IJsObject window = args.Frame.ExecuteJavaScript<IJsObject>("window").Result;
        window["myObject"] = new MyObject();
    });
```

The MyObject class may look like this:

```
public class MyObject
    {
        public string SayHelloTo(string firstName) => "Hello " + firstName + "!";
    }
```

When the property is set, you can call methods of the injected .NET object from JavaScript:

```
window.myObject.SayHelloTo('John');
```

This handler may be invoked several times for the same frame.

**Important:** you should avoid executing a JavaScript code that modifies the DOM tree of the web page being loaded.You must not use DotNetBrowser DOM API to remove the frame for which this callback is invoked, otherwise the render process will crash.

**Important:** the engine will be blocked until you return control from the callback.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property JsDialogs

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## JsDialogs

Gets the IJsDialogs instance that can be used for configuring the JavaScript dialogs that can be shown by browser.

```
IJsDialogs JsDialogs { get; }
```

## Property Value

IJsDialogs

The IJsDialogs instance associated with this IBrowser.

# Property Keyboard

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Keyboard

Gets the IKeyboard instance that can be used for listening and simulating keyboard input events.

```
IKeyboard Keyboard { get; }
```

## Property Value

IKeyboard

The IKeyboard instance associated with this IBrowser.

# Property MainFrame

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## MainFrame

The main frame on the currently loaded web page, if it exists.

```
IFrame MainFrame { get; }
```

## Property Value

IFrame

The main frame on the currently loaded web page. Can be null if there is no main frame.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Mouse

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Mouse

Gets the IMouse instance that can be used for listening and simulating mouse input events.

```
IMouse Mouse { get; }
```

## Property Value

IMouse

The IMouse instance associated with this IBrowser.

# Property Navigation

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Navigation

Gets the INavigation instance that can be used for controlling navigation in the current browser instance.

```
INavigation Navigation { get; }
```

## Property Value

INavigation

The INavigation instance associated with this IBrowser.

# Property OpenExtensionActionPopupHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## OpenExtensionActionPopupHandler

Gets or sets a handler that is used for handling the popups that are requested to be shown by the extension action.

```
IHandler<OpenExtensionActionPopupParameters> OpenExtensionActionPopupHandler { get; set; }
```

## Property Value

IHandler<OpenExtensionActionPopupParameters>

## Remarks

The OpenExtensionActionPopupParameters object contains the popup browser.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property OpenPopupHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## OpenPopupHandler

Gets or sets a handler that is used when a new popup browser instance should be opened.

```
IHandler<OpenPopupParameters> OpenPopupHandler { get; set; }
```

## Property Value

IHandler<OpenPopupParameters>

## Remarks

**Important:** the engine will be blocked until you return control from the callback.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property Passwords

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Passwords

Gets the IPasswords instance that can be used for working with logins and passwords saved in the Chromium password store.

```
IPasswords Passwords { get; }
```

## Property Value

IPasswords

The IPasswords instance associated with this IBrowser.

# Property PrintHtmlContentHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## PrintHtmlContentHandler

Gets or sets a handler that is used when the HTML content printing is initiated.

```
IHandler<PrintHtmlContentParameters, PrintHtmlContentResponse> PrintHtmlContentHandler {
get; set; }
```

## Property Value

IHandler<PrintHtmlContentParameters, PrintHtmlContentResponse>

## Remarks

In this callback you can configure the print settings. The workflow is the following:

- Find the required printer using the IPrinters<TPrintSettings, TPdfPrintSettings> interface. You can obtain an instance of `Printers` using the Printers property.
- Get the current IPrintJob<TPrintSettings> from the printer.
- Configure the required settings for the print job using the IPrintSettings interface provided by the Settings property.
- Apply the configured settings using the Apply() method.
- Tell the browser to proceed with the printing using the configured settings: Print(SystemPrinter<IHtmlSettings>).

Use the Print(SystemPrinter<IHtmlSettings>) method to continue printing using the selected printer.

Use the Cancel() method to cancel printing.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel() will be used.

If printing is initiated when the browser is loading a web page, this callback is not invoked.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property PrintPdfContentHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## PrintPdfContentHandler

Gets or sets a handler that is used when the PDF content printing is initiated.

```
IHandler<PrintPdfContentParameters, PrintPdfContentResponse> PrintPdfContentHandler { get;
set; }
```

## Property Value

IHandler<PrintPdfContentParameters, PrintPdfContentResponse>

## Remarks

In this callback you can configure the print settings. The workflow is the following:

- Find the required printer using the IPrinters<TPrintSettings, TPdfPrintSettings> interface. You can obtain an instance of `Printers` using the Printers property.
- Get the current IPrintJob<TPrintSettings> from the printer.
- Configure the required settings for the print job using the IPrintSettings interface provided by the Settings property.
- Apply the configured settings using the Apply() method.
- Tell the browser to proceed with the printing using the configured settings: Print(SystemPrinter<IPdfSettings>).

Use the Print(SystemPrinter<IPdfSettings>) method to continue printing using the selected printer.

Use the Cancel() method to cancel printing.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel() will be used.

If printing is initiated when the browser is loading a web page, this callback is not invoked.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property Profile

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this browser.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this IBrowser.

# Property RequestPdfDocumentPasswordHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## RequestPdfDocumentPasswordHandler

Gets or sets a handler that is used when the frame requests the password for an encrypted PDF document.

```
IHandler<RequestPdfDocumentPasswordParameters, RequestPdfDocumentPasswordResponse>
RequestPdfDocumentPasswordHandler { get; set; }
```

## Property Value

IHandler<RequestPdfDocumentPasswordParameters, RequestPdfDocumentPasswordResponse>

## Remarks

Use the Password(string) method to provide the password for the encrypted PDF document.

Use theShowPasswordDialog method to invoke the default PDF viewer password dialog.

Use theCancel method to cancel the request.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property RequestPrintHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## RequestPrintHandler

Gets or sets a handler that is used when the printing is initiated.

```
IHandler<RequestPrintParameters, RequestPrintResponse> RequestPrintHandler { get; set; }
```

## Property Value

IHandler<RequestPrintParameters, RequestPrintResponse>

## Remarks

In this callback you can cancel or allow showing print preview dialog, or print the content programmatically.

By default, the printing is canceled if there is no browser view initialized from the browser.

Use the ShowPrintPreview() method to show the print preview.

Use the Print() method to continue printing and configure it programmatically in PrintHtmlContentHandler and PrintPdfContentHandler.

Use the Cancel() method to cancel printing.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel() will be used.

**Important:** do not use ShowPrintPreview() if the browser is not displayed. In this case, the dialog will not be visible - as a result, the end user will not be able to close it or confirm printing.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property SelectCertificateHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## SelectCertificateHandler

Gets or sets a handler that is used when the web server requires authorization via client certificate.

```
IHandler<SelectCertificateParameters, SelectCertificateResponse> SelectCertificateHandler {
get; set; }
```

## Property Value

IHandler<SelectCertificateParameters, SelectCertificateResponse>

## Remarks

The SelectCertificateParameters object contains information about the authorization request.

To access the list of the certificates from the system storage that match the server criteria you can use the Certificates property.

You can use this handler to display a dialog with the list of the available certificates that allows the user to select the required certificate. Please note that it is not necessary to display a dialog in this handler.

This handler itself is never called on the UI thread directly.

Use the Cancel() method to cancel the authorization.

Use the Select(int) method to inform the browser to use the client certificate located by the passed index in the certificate list.

Use the Select(Certificate) method to inform the browser to use the specified client certificate.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel() will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property Settings

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Settings

Gets the IBrowserSettings instance that can be used for modifying the settings of the browser.

```
IBrowserSettings Settings { get; }
```

## Property Value

IBrowserSettings

The IBrowserSettings instance associated with this IBrowser.

# Property ShowContextMenuHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## ShowContextMenuHandler

Gets or sets a handler that is used when the browser should show a context menu.

```
IHandler<ShowContextMenuParameters, ShowContextMenuResponse> ShowContextMenuHandler { get;
set; }
```

## Property Value

IHandler<ShowContextMenuParameters, ShowContextMenuResponse>

## Remarks

The passed ShowContextMenuParameters object contains information about the context menu to show.

Use the Close() method to close the context menu without choice.

Use the Select(ContextMenuItem) method to choose the context menu item.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Close() will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Property Size

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Size

Gets or sets the browser size.

```
Size Size { get; set; }
```

## Property Value

Size

The browser size. By default, browser's size is empty.

## Remarks

Setting this property updates the size of this Browser instance with the given width and height. Many web pages rely on the Browser's size and require that it's not empty. DOM document of a web page might not be loaded and displayed at all, because there's no sense in loading and rendering DOM document when it's empty.

As a result, you should set the browser size programmatically when you don't need to display content of the loaded web page, but the web page must "think" it has been loaded in a browser instance with a non-empty size.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property StartDownloadHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## StartDownloadHandler

Gets or sets a handler that is used when the browser is about to start downloading the file.

```
IHandler<StartDownloadParameters, StartDownloadResponse> StartDownloadHandler { get; set; }
```

## Property Value

IHandler<StartDownloadParameters, StartDownloadResponse>

## Remarks

Use the DownloadTo(string) method to confirm the download and specify the location to store the downloaded file.

Use theCancel() method if you do not need to download the file.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel() will be used.

## Exceptions

ObjectDisposedException

The IDownloads has already been disposed.

# Property TextFinder

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## TextFinder

Gets the ITextFinder instance that can be used for finding text on a web page loaded in the browser.

```
ITextFinder TextFinder { get; }
```

## Property Value

ITextFinder

The ITextFinder instance associated with this IBrowser.

# Property Title

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Title

Gets the title of the currently loaded web page.

```
string Title { get; }
```

## Property Value

string

The title of the currently loaded web page. Can be an empty string if the browser hasn't loaded any web page yet.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Touch

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Touch

Gets the ITouch instance that can be used for listening touch input events.

```
ITouch Touch { get; }
```

## Property Value

ITouch

The ITouch instance associated with this IBrowser.

# Property Url

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the currently loaded web page or an empty string if the browser hasn't loaded any web page yet.

```
string Url { get; }
```

## Property Value

string

The URL of the currently loaded web page. Can be an empty string if the browser hasn't loaded any web page yet.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property UserAgent

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## UserAgent

Gets or sets the `user-agent` of the current browser instance.

```
string UserAgent { get; set; }
```

## Property Value

string

The `user-agent` string for this browser. If the string has not been overridden, then the property is the same as the default `user-agent` string obtained through UserAgent

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property UserDataProfiles

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## UserDataProfiles

Gets the IUserDataProfiles instance that can be used for working with user data profiles saved in the Chromium user data store.

```
IUserDataProfiles UserDataProfiles { get; }
```

## Property Value

IUserDataProfiles

The IUserDataProfiles instance associated with this IBrowser.

# Property Zoom

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Zoom

Gets the IZoom instance that can be used for zooming content of a web page loaded in the current browser instance.

```
IZoom Zoom { get; }
```

## Property Value

IZoom

The IZoom instance associated with this IBrowser.

# Method DisposeAsync

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## DisposeAsync(BrowserDisposeOptions)

Asynchronously disposes the current browser instance according to the given options.

```
Task<bool> DisposeAsync(BrowserDisposeOptions browserDisposeOptions)
```

## Parameters

`browserDisposeOptions` BrowserDisposeOptions

browser dispose options.

## Returns

Task<bool>

The task that represents the result of the asynchronous dispose process. `True` indicates that the browser instance is successfully disposed without any dialogs, `false` if the dispose process was canceled. This may happen if the BeforeUnloadHandler handler was invoked and the Stay() response was sent.

## Remarks

If the currently loaded web page registers the `onbeforeunload` JavaScript event and the given options tell the browser to fire the `beforeunload` event, then this method will not dispose the browser right away.

The BeforeUnloadHandler will be invoked in this case and if the handler tells the browser to stay on the web page, then the browser will not be disposed.

The following example demonstrates how to dispose the browser instance as if a user manually clicking the close button of the tab/window.

```
browser.DisposeAsync(new BrowserDisposeOptions(){BeforeUnloadEventHandled = true});
```

If the browser is already disposed, then this method does nothing.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method Focus

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Focus()

Tells the browser that it has focus and must be activated.

```
void Focus()
```

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method ReplaceMisspelledWord

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## ReplaceMisspelledWord(string)

Replaces misspelled word under cursor on the currently loaded web page with the given `word`. If there is no misspelled word under cursor, this method does nothing.

```
void ReplaceMisspelledWord(string word)
```

## Parameters

`word` string

a string that represents the word for replacement

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

ArgumentException

The `word` is null, empty, or consists only of white-space characters.

# Method SaveWebPage

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## SaveWebPage(string, string, SavePageType)

Initiates the saving process of the currently loaded web page. The web page can be saved as a single HTML file or the file with resources. Before saving a web page make sure that it is not being loaded. It is recommended to completely loaded the web page and only then save it.

```
bool SaveWebPage(string filePath, string resourcesPath, SavePageType saveType)
```

### Parameters

`filePath` string

an absolute path to a file in which the web page will be saved.

`resourcesPath` string

an absolute path to a directory in which the resources (e.g. images, css) of the web page will be saved. If the directory does not exist, it will be created

`saveType` SavePageType

determines how the web page will be saved: as an HTML file with all the required resources (e.g. images, css etc.), a single HTML or MHTML file

### Returns

bool

`true` if the saving process has been initialized successfully, `false` otherwise.

### Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

ArgumentException

The `filePath` is null, empty, or consists only of white-space characters.

ArgumentException

The `resourcesPath` is null, empty, or consists only of white-space characters.

# Method SetPrivateKeyProviderPin

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## SetPrivateKeyProviderPin(CspParameters)

Sets the key exchange PIN for the particular cryptographic service provider (CSP) and the particular key container in it. This functionality can be used to set the PIN that is requested when trying to use a client certificate stored on the smart card.

```
bool SetPrivateKeyProviderPin(CspParameters cspParameters)
```

### Parameters

`cspParameters` CspParameters

The parameters of the private key CSP container.

### Returns

bool

`true` if the PIN was set successfully, `false` otherwise.

### Remarks

This functionality is Windows-only. The implementation will return false in other environments.

If the PIN was not set successfully, the actual error will be written to DotNetBrowser logs.

### Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

ArgumentException

The `cspParameters` is null.

ArgumentException

Any of the required parameters in `cspParameters` is null, empty, or consists only of white-space characters.

# Method TakeImage

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## TakeImage()

Creates and returns an image of the currently loaded web page.

```
Bitmap TakeImage()
```

## Returns

Bitmap

a Bitmap that contains the image of the currently loaded web page.

## Remarks

The bitmap size depends on the size of the current browser instance and the device scale factor of the display where it is located at the moment. If the current browser size is empty, then the bitmap will be empty as well.

For example, if the device scale factor of the display where the browser instance is located at the moment is 2.0 and the browser size is 100x100, then the size of the bitmap is expected to be 200x200.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

BitmapTimeoutException

The Bitmap can not be retrieved within `10` seconds.

# Method Unfocus

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Unfocus()

Tells the browser that it does not have focus and must be deactivated.

```
void Unfocus()
```

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Event BrowserBecameResponsive

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the browser instance has become responsive.

```
event EventHandler<BrowserBecameResponsiveEventArgs> BrowserBecameResponsive
```

## Returns

EventHandler<BrowserBecameResponsiveEventArgs>

Occurs when the browser instance has become responsive.

# Event BrowserBecameUnresponsive

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the browser instance has become unresponsive.

```
event EventHandler<BrowserBecameUnresponsiveEventArgs> BrowserBecameUnresponsive
```

## Returns

EventHandler<BrowserBecameUnresponsiveEventArgs>

Occurs when the browser instance has become unresponsive.

# Event ConsoleMessageReceived

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the message was added to the console.

```
event EventHandler<ConsoleMessageReceivedEventArgs> ConsoleMessageReceived
```

## Returns

EventHandler<ConsoleMessageReceivedEventArgs>

Occurs when the message was added to the console.

# Event FaviconChanged

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the web page favicon has changed.

```
event EventHandler<FaviconChangedEventArgs> FaviconChanged
```

## Returns

EventHandler<FaviconChangedEventArgs>

Occurs when the web page favicon has changed.

# Event FocusGained

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the browser instance has gained the focus.

```
event EventHandler<FocusGainedEventArgs> FocusGained
```

## Returns

EventHandler<FocusGainedEventArgs>

Occurs when the browser instance has gained the focus.

# Event FocusLost

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the browser instance has lost the focus.

```
event EventHandler<FocusLostEventArgs> FocusLost
```

## Returns

EventHandler<FocusLostEventArgs>

Occurs when the browser instance has lost the focus.

# Event FocusRequested

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when JavaScript sends a request to focus the Browser instance by calling the `window.focus()` method.

```
event EventHandler<FocusRequestedEventArgs> FocusRequested
```

## Returns

EventHandler<FocusRequestedEventArgs>

Occurs when JavaScript sends a request to focus the Browser instance by calling the window.focus() method.

# Event FrameCreated

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the IFrame has been created.

```
event EventHandler<FrameCreatedEventArgs> FrameCreated
```

## Returns

EventHandler<FrameCreatedEventArgs>

Occurs when the has been created.

# Event FrameDeleted

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the IFrame has been deleted.

```
event EventHandler<FrameDeletedEventArgs> FrameDeleted
```

## Returns

EventHandler<FrameDeletedEventArgs>

Occurs when the has been deleted.

# Event MediaStreamCaptureStarted

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the web page has started capturing an audio or video stream.

```
event EventHandler<MediaStreamCaptureStartedEventArgs> MediaStreamCaptureStarted
```

## Returns

EventHandler<MediaStreamCaptureStartedEventArgs>

Occurs when the web page has started capturing an audio or video stream.

# Event MediaStreamCaptureStopped

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the web page has stopped capturing an audio or video stream.

```
event EventHandler<MediaStreamCaptureStoppedEventArgs> MediaStreamCaptureStopped
```

## Returns

EventHandler<MediaStreamCaptureStoppedEventArgs>

Occurs when the web page has stopped capturing an audio or video stream.

# Event PrintPreviewClosed

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when print preview dialog is closed.

```
event EventHandler<PrintPreviewClosedEventArgs> PrintPreviewClosed
```

## Returns

EventHandler<PrintPreviewClosedEventArgs>

Occurs when print preview dialog is closed.

# Event PrintPreviewOpened

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when print preview dialog is opened.

```
event EventHandler<PrintPreviewOpenedEventArgs> PrintPreviewOpened
```

## Returns

EventHandler<PrintPreviewOpenedEventArgs>

Occurs when print preview dialog is opened.

# Event RenderProcessTerminated

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when a render process has been terminated.

```
event EventHandler<RenderProcessTerminatedEventArgs> RenderProcessTerminated
```

## Returns

EventHandler<RenderProcessTerminatedEventArgs>

Occurs when a render process has been terminated.

# Event SpellCheckCompleted

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when spell checking on the frame has been completed.

```
event EventHandler<SpellCheckCompletedEventArgs> SpellCheckCompleted
```

## Returns

EventHandler<SpellCheckCompletedEventArgs>

Occurs when spell checking on the frame has been completed.

# Event StatusChanged

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the status text has been changed.

```
event EventHandler<StatusChangedEventArgs> StatusChanged
```

## Returns

EventHandler<StatusChangedEventArgs>

Occurs when the status text has been changed.

# Event TitleChanged

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the web page title has been changed.

```
event EventHandler<TitleChangedEventArgs> TitleChanged
```

## Returns

EventHandler<TitleChangedEventArgs>

Occurs when the web page title has been changed.

# Event UpdateBoundsRequested

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when JavaScript requests to update bounds of the Browser instance.

```
event EventHandler<UpdateBoundsRequestedEventArgs> UpdateBoundsRequested
```

## Returns

EventHandler<UpdateBoundsRequestedEventArgs>

Occurs when JavaScript requests to update bounds of the Browser instance.

## Remarks

JavaScript can request to update bounds of the Browser instance through the following JavaScript functions:

- `window.moveTo()` - moves a window to the specified position.
- `window.moveBy()` - moves a window a specified number of pixels relative to its current coordinates.
- `window.resizeTo()` - resizes the window to the specified width and height.
- `window.resizeBy()` - resizes the window by the specified pixels.

# Interface IBrowserSettings

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

The settings of the browser.

```
public interface IBrowserSettings : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

AllowJavaScriptAccessClipboard

Allows or disallows JavaScript code on the web pages loaded in the browser to access clipboard.

AllowLoadingImagesAutomatically

Allows or disallows loading images automatically on the web pages loaded in the browser.

AllowRunningInsecureContent

Allows or disallows running an insecure content in the browser.

AllowScriptsToCloseWindows

Allows or disallows JavaScript code on the web pages loaded in the browser to close the browser.

DefaultBackgroundColor

Gets or sets the default browser background color.

DefaultEncoding

Gets or sets the default text encoding.

DefaultFontSize

Gets or sets the default browser font size.

ImagesEnabled

Enables or disables images displaying on the web pages loaded in the browser.

JavaScriptEnabled

Enables or disables JavaScript on the web pages loaded in the browser.

LocalStorageEnabled

Enables or disables the local storage in the browser.

OverscrollHistoryNavigationEnabled

Enables or disables the back/forward navigation with a left/right swipe in the browser.

PluginsEnabled

Enables or disables plugins on the web pages loaded in the browser.

PreferredColorScheme

Gets or sets the preferred color scheme for the web content in the browser.

ScrollbarsHidden

Hides or shows scrollbars on the web pages.

TransparentBackgroundEnabled

Enables or disables transparent background on the web pages.By default, the background is always opaque.

WebRtcIpHandlingPolicy

Gets or sets the WebRTC IP handling policy for the browser.

# Property AllowJavaScriptAccessClipboard

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## AllowJavaScriptAccessClipboard

Allows or disallows JavaScript code on the web pages loaded in the browser to access clipboard.

```
bool AllowJavaScriptAccessClipboard { get; set; }
```

## Property Value

bool

`true` if JavaScript is allowed, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property AllowLoadingImagesAutomatically

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## AllowLoadingImagesAutomatically

Allows or disallows loading images automatically on the web pages loaded in the browser.

```
bool AllowLoadingImagesAutomatically { get; set; }
```

## Property Value

bool

`true` if loading images automatically is allowed, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property AllowRunningInsecureContent

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## AllowRunningInsecureContent

Allows or disallows running an insecure content in the browser.

```
bool AllowRunningInsecureContent { get; set; }
```

## Property Value

bool

`true` if running an insecure content is allowed, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property AllowScriptsToCloseWindows

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## AllowScriptsToCloseWindows

Allows or disallows JavaScript code on the web pages loaded in the browser to close the browser.

```
bool AllowScriptsToCloseWindows { get; set; }
```

## Property Value

bool

`true` if closing the browser from JavaScript is allowed, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property DefaultBackgroundColor

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## DefaultBackgroundColor

Gets or sets the default browser background color.

```
Color DefaultBackgroundColor { get; set; }
```

## Property Value

Color

The default browser background color.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property DefaultEncoding

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## DefaultEncoding

Gets or sets the default text encoding.

```
string DefaultEncoding { get; set; }
```

## Property Value

string

The default text encoding.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property DefaultFontSize

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## DefaultFontSize

Gets or sets the default browser font size.

```
FontSize DefaultFontSize { get; set; }
```

## Property Value

FontSize

The default browser font size.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property ImagesEnabled

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## ImagesEnabled

Enables or disables images displaying on the web pages loaded in the browser.

```
bool ImagesEnabled { get; set; }
```

## Property Value

bool

`true` if displaying images is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property JavaScriptEnabled

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## JavaScriptEnabled

Enables or disables JavaScript on the web pages loaded in the browser.

```
bool JavaScriptEnabled { get; set; }
```

## Property Value

bool

`true` if JavaScript is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property LocalStorageEnabled

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## LocalStorageEnabled

Enables or disables the local storage in the browser.

```
bool LocalStorageEnabled { get; set; }
```

## Property Value

bool

`true` if the local storage is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property OverscrollHistoryNavigationEnabled

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## OverscrollHistoryNavigationEnabled

Enables or disables the back/forward navigation with a left/right swipe in the browser.

```
bool OverscrollHistoryNavigationEnabled { get; set; }
```

## Property Value

bool

`true` if back/forward navigation with a left/right swipe is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property PluginsEnabled

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## PluginsEnabled

Enables or disables plugins on the web pages loaded in the browser.

```
bool PluginsEnabled { get; set; }
```

## Property Value

bool

`true` if plugins are enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property PreferredColorScheme

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## PreferredColorScheme

Gets or sets the preferred color scheme for the web content in the browser.

```
PreferredColorScheme PreferredColorScheme { get; set; }
```

## Property Value

PreferredColorScheme

The preferred color scheme value.

## Remarks

The scheme is used to evaluate the `prefers-color-scheme` media query and resolve UA color scheme to be used based on the `supported-color-schemes` META tag and CSS property.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property ScrollbarsHidden

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## ScrollbarsHidden

Hides or shows scrollbars on the web pages.

```
bool ScrollbarsHidden { get; set; }
```

## Property Value

bool

`true` if scrollbars are hidden, `false` otherwise.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property TransparentBackgroundEnabled

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## TransparentBackgroundEnabled

Enables or disables transparent background on the web pages.By default, the background is always opaque.

```
bool TransparentBackgroundEnabled { get; set; }
```

## Property Value

bool

`true` if transparent background is enabled, `false` otherwise.

## Remarks

This property supports only the OffScreen rendering mode on Windows and Linux, and the both rendering modes on macOS.

The attempt to set this property to `true` in the unsupported rendering mode will result in an exception.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property WebRtcIpHandlingPolicy

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## WebRtcIpHandlingPolicy

Gets or sets the WebRTC IP handling policy for the browser.

```
WebRtcIpHandlingPolicy WebRtcIpHandlingPolicy { get; set; }
```

## Property Value

WebRtcIpHandlingPolicy

The current WebRTC IP handling policy for the browser.

## Remarks

Specifying the custom WebRTC IP handling policy can be used to protect against WebRTC leaks - the WebRTC itself will remain enabled.

## Exceptions

ObjectDisposedException

The IBrowserSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Interface IBrowserView

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

The interface that is implemented by UI components which are able to display the browser.

```
public interface IBrowserView
```

**Extension Methods**

BrowserViewExtensions.InitializeFrom(IBrowserView, IBrowser)

# Interface IOffScreenRenderProvider

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Provides access to an image rendered by the browser that works in the off-screen rendering mode.

```
public interface IOffScreenRenderProvider
```

## Properties

PaintHandler

Gets or sets a handler that is used when the browser contents should be rendered.

## Methods

Hide()

Tells the browser to stop rendering its content as if it was hidden.

Show()

Tells the browser to start rendering its content as if it was shown on the screen.

## Events

CursorChanged

Occurs when the cursor should be changed.

# Property PaintHandler

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## PaintHandler

Gets or sets a handler that is used when the browser contents should be rendered.

```
IHandler<PaintParameters> PaintHandler { get; set; }
```

## Property Value

IHandler<PaintParameters>

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Method Hide

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Hide()

Tells the browser to stop rendering its content as if it was hidden.

```
void Hide()
```

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Method Show

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

## Show()

Tells the browser to start rendering its content as if it was shown on the screen.

```
void Show()
```

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Event CursorChanged

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Occurs when the cursor should be changed.

```
event EventHandler<CursorChangedEventArgs> CursorChanged
```

## Returns

EventHandler<CursorChangedEventArgs>

Occurs when the cursor should be changed.

# Enum PreferredColorScheme

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

The preferred color scheme for the web content in the browser.

```
public enum PreferredColorScheme
```

## Fields

`Dark = 1`

The dark color scheme is preferred.

`Light = 2`

The light color scheme is preferred.

`Unspecified = 0`

The preferred color scheme has not been specified.

## Remarks

The scheme is used to evaluate the `prefers-color-scheme` media query and resolve UA color scheme to be used based on the `supported-color-schemes` META tag and CSS property.

# Enum SavePageType

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

Determines how the web page will be saved.

```
public enum SavePageType
```

## Fields

`CompletePage = 2`

HTML file with required resources (e.g. images, css etc.).

`Html = 1`

Single HTML.

`MHtml = 3`

MHTML file.

# Enum WebRtcIpHandlingPolicy

Namespace: DotNetBrowser.Browser

Assembly: DotNetBrowser.dll

The media performance/privacy tradeoffs which affect how WebRTC traffic will be routed and how much local address information will be exposed.

```
public enum WebRtcIpHandlingPolicy
```

## Fields

`Default = 1`

This is the default behavior. Currently, WebRTC has the right to enumerate all interfaces and bind them to discover public interfaces.

`DefaultPublicAndPrivateInterfaces = 3`

WebRTC should only use the default route used by HTTP. This also exposes the associated default private address. Default route is the route chosen by the OS on a multi-homed endpoint.

`DefaultPublicInterfacesOnly = 2`

WebRTC should only use the default route used by HTTP. This doesn't expose any local addresses.

`DisableNonProxiedUdp = 4`

WebRTC should only use TCP to contact peers or servers unless the proxy server supports UDP. This doesn't expose any local addresses either.

`Unspecified = 0`

The policy has not been specified.

# Namespace DotNetBrowser.Browser.Dialogs

## Interfaces

IDialogs
: The browser dialogs.

IJsDialogs
: The JavaScript dialogs.

# Interface IDialogs

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

The browser dialogs.

```
public interface IDialogs : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

OpenDirectoryHandler
Gets or sets a handler that is used when the browser requests to display a file chooser dialog to open a directory.

OpenExternalAppHandler
Gets or sets a handler that is used when the currently loaded web page wants to open a link in the associated external application.

OpenFileHandler
Gets or sets a handler that is used when the browser requests to display a file chooser dialog to open a file.

OpenMultipleFilesHandler
Gets or sets a handler that is used when the browser requests to display a file chooser dialog to open several files.

RepostFormHandler
Gets or sets a handler that is used when a web page with POST data is going to be reloaded and the user must confirm that the POST data can be resubmitted.

SaveAsPdfHandler
Gets or sets a handler that is used when the browser requests to display a file chooser dialog to save content as PDF.

SaveFileHandler

Gets or sets a handler that is used when the browser requests to display a file chooser dialog to save the file.

SelectColorHandler

Gets or sets a handler that is used when the user clicks an `<input type='color'>` HTML5 element.

# Property OpenDirectoryHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## OpenDirectoryHandler

Gets or sets a handler that is used when the browser requests to display a file chooser dialog to open a directory.

```
IHandler<OpenDirectoryParameters, OpenDirectoryResponse> OpenDirectoryHandler { get; set; }
```

## Property Value

IHandler<OpenDirectoryParameters, OpenDirectoryResponse>

## Remarks

You can use this handler to display the directory chooser dialog, or provide the chosen directory without displaying any dialogs.

Use the SelectDirectory(string) method to provide the selected directory to the browser.

Use the Cancel() method to cancel the dialog.

If the handler throws an exception, the Cancel() method will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property OpenExternalAppHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## OpenExternalAppHandler

Gets or sets a handler that is used when the currently loaded web page wants to open a link in the associated external application.

```
IHandler<OpenExternalAppParameters, OpenExternalAppResponse> OpenExternalAppHandler { get;
set; }
```

## Property Value

IHandler<OpenExternalAppParameters, OpenExternalAppResponse>

## Remarks

You can display your own dialog with the localized message that you can get from the parameters.

Use the Open() method to tell the browser that the link should be opened in the associated external application. If the application is not running, the operating system should launch the application and open the link in it.

If the handler throws an exception, the Cancel() method will be invoked.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property OpenFileHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## OpenFileHandler

Gets or sets a handler that is used when the browser requests to display a file chooser dialog to open a file.

```
IHandler<OpenFileParameters, OpenFileResponse> OpenFileHandler { get; set; }
```

## Property Value

IHandler<OpenFileParameters, OpenFileResponse>

## Remarks

You can use this handler to display the file chooser, or provide the file without displaying any dialogs.

Use the SelectFile(string) method to provide the selected file to the browser.

Use the Cancel() method to cancel the dialog.

If the handler throws an exception, the Cancel() method will be invoked.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property OpenMultipleFilesHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## OpenMultipleFilesHandler

Gets or sets a handler that is used when the browser requests to display a file chooser dialog to open several files.

```
IHandler<OpenMultipleFilesParameters, OpenMultipleFilesResponse> OpenMultipleFilesHandler {
get; set; }
```

## Property Value

IHandler<OpenMultipleFilesParameters, OpenMultipleFilesResponse>

## Remarks

You can use this handler to display the file chooser, or provide the files without displaying any dialogs.

Use the SelectFiles(params string[]) method to provide the selected files to the browser.

Use the Cancel() method to cancel the dialog.

If the handler throws an exception, the Cancel() method will be invoked.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property RepostFormHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## RepostFormHandler

Gets or sets a handler that is used when a web page with POST data is going to be reloaded and the user must confirm that the POST data can be resubmitted.

```
IHandler<RepostFormParameters, RepostFormResponse> RepostFormHandler { get; set; }
```

## Property Value

IHandler<RepostFormParameters, RepostFormResponse>

## Remarks

You can use this handler to display a confirmation dialog where you ask the user whether the web page can be reloaded or not.

Use the Repost() method to allow resubmitting POST data.

Use the Cancel() method to cancel the page reloading.

If the handler throws an exception, the Repost() method method will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property SaveAsPdfHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## SaveAsPdfHandler

Gets or sets a handler that is used when the browser requests to display a file chooser dialog to save content as PDF.

```
IHandler<SaveAsPdfParameters, SaveAsPdfResponse> SaveAsPdfHandler { get; set; }
```

## Property Value

IHandler<SaveAsPdfParameters, SaveAsPdfResponse>

## Remarks

You can use this handler to display the save dialog, or provide the full file path without displaying any dialogs.

Use the SaveToFile(string) method to provide the selected files to the browser.

Use the Cancel() method to cancel the dialog.

If the handler throws an exception, the Cancel() method will be invoked.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property SaveFileHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## SaveFileHandler

Gets or sets a handler that is used when the browser requests to display a file chooser dialog to save the file.

```
IHandler<SaveFileParameters, SaveFileResponse> SaveFileHandler { get; set; }
```

## Property Value

IHandler<SaveFileParameters, SaveFileResponse>

## Remarks

You can use this handler to display the save dialog, or provide the full file path without displaying any dialogs.

Use the SaveToFile(string) method to provide the selected files to the browser.

Use the Cancel() method to cancel the dialog.

If the handler throws an exception, the Cancel() method will be invoked.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property SelectColorHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## SelectColorHandler

Gets or sets a handler that is used when the user clicks an `<input type='color'>` HTML5 element.

```
IHandler<SelectColorParameters, SelectColorResponse> SelectColorHandler { get; set; }
```

## Property Value

IHandler<SelectColorParameters, SelectColorResponse>

## Remarks

You can use this handler to display your own color chooser dialog or set the required color programmatically without displaying any dialogs.

Use the SelectColor(Color) method to provide the selected color to the browser.

Use the Cancel() method to cancel the dialog.

If the handler throws an exception, the Cancel() method will be invoked.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Interface IJsDialogs

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

The JavaScript dialogs.

```
public interface IJsDialogs : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

AlertHandler

Gets or sets a handler that is used when JavaScript alert dialog should be displayed.

BeforeUnloadHandler

Gets or sets a handler that is used when the web page is about to be unloaded.

ConfirmHandler

Gets or sets a handler that is used when JavaScript confirmation dialog should be displayed.

PromptHandler

Gets or sets a handler that is used when JavaScript dialog prompting the user to input some text should be displayed.

# Property AlertHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## AlertHandler

Gets or sets a handler that is used when JavaScript alert dialog should be displayed.

```
IHandler<AlertParameters> AlertHandler { get; set; }
```

## Property Value

IHandler<AlertParameters>

## Remarks

In this callback you can display standard modal dialog with the message that you can get from Alert Parameters.

Please note that it is not necessary to display a dialog in this handler.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property BeforeUnloadHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## BeforeUnloadHandler

Gets or sets a handler that is used when the web page is about to be unloaded.

```
IHandler<BeforeUnloadParameters, BeforeUnloadResponse> BeforeUnloadHandler { get; set; }
```

## Property Value

IHandler<BeforeUnloadParameters, BeforeUnloadResponse>

## Remarks

Some web pages may override the `window.onbeforeunload` JavaScript function, so that a confirmation dialog will be shown every time when the user tries to reload a web page or navigate to another web page. You can use this callback to display an appropriate confirmation message dialog to ask the user if they really want to leave or reload the web page. To find out whether the web page is going to be reloaded you can check the Cause property.

Use the Stay() method to stay on the current web page.

Use the Leave() method to reload or navigate to another web page.

If the handler throws an exception, the Leave() method will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property ConfirmHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## ConfirmHandler

Gets or sets a handler that is used when JavaScript confirmation dialog should be displayed.

```
IHandler<ConfirmParameters, ConfirmResponse> ConfirmHandler { get; set; }
```

## Property Value

IHandler<ConfirmParameters, ConfirmResponse>

## Remarks

In this handler you can display standard modal dialog with the message that you can get from Confirm Parameters. Please note that it is not necessary to display a dialog in this handler.

Use the Ok() to notify that the JavaScript dialog is closed with the "OK" action.

Use the Cancel() method to dismiss the dialog.

If the handler throws an exception, the Cancel() method will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property PromptHandler

Namespace: DotNetBrowser.Browser.Dialogs

Assembly: DotNetBrowser.dll

## PromptHandler

Gets or sets a handler that is used when JavaScript dialog prompting the user to input some text should be displayed.

```
IHandler<PromptParameters, PromptResponse> PromptHandler { get; set; }
```

## Property Value

IHandler<PromptParameters, PromptResponse>

## Remarks

The passed PromptParameters object contains the text of the prompt message and the default response text. You can use this information to display standard modal dialog to obtain a prompt response from user. Please note that it is not necessary to display a dialog in this callback.

Use the SubmitText(string) method to close the dialog and pass the text value passed as a parameter to the engine.

Use the Cancel() method to cancel the dialog.

If the callback throws an exception, the Cancel() method will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Namespace DotNetBrowser.Browser.Dialogs.Handlers

## Classes

AlertParameters
The base class for common dialog parameters.

BeforeUnloadParameters
The base class for common dialog parameters.

BeforeUnloadResponse
The response to the BeforeUnloadHandler.

CommonDialogParameters
The base class for common dialog parameters.

ConfirmParameters
The base class for common dialog parameters.

ConfirmResponse
The response to the ConfirmHandler.

DialogParameters
The base class for dialog parameters.

FileChooserParameters
The base class for the common file chooser parameters.

FilteredFileChooserParameters
The base class for the common file chooser parameters.

OpenDirectoryParameters
The parameters of the OpenDirectoryHandler.

OpenDirectoryResponse
The response to the OpenDirectoryHandler.

OpenExternalAppParameters
The parameters of the OpenExternalAppHandler.

OpenExternalAppResponse
The response to the OpenExternalAppHandler.

OpenFileParameters
The parameters of the OpenFileHandler.

OpenFileResponse
The response to the OpenFileHandler.

OpenMultipleFilesParameters
The parameters of the OpenMultipleFilesHandler.

OpenMultipleFilesResponse
The response to the OpenMultipleFilesHandler.

PromptParameters
The base class for common dialog parameters.

PromptResponse
The response to the PromptHandler.

RepostFormParameters
The base class for common dialog parameters.

RepostFormResponse
The response to the RepostFormHandler.

SaveAsPdfParameters
The parameters of the SaveAsPdfHandler.

SaveAsPdfResponse
The response to the SaveAsPdfHandler.

SaveFileParameters
The parameters of the SaveFileHandler.

SaveFileResponse
The response to the SaveFileHandler.

SelectCertificateParameters
The parameters of the SelectCertificateHandler.

SelectCertificateResponse
The response to SelectCertificateHandler

SelectColorParameters
The parameters for the SelectColorHandler.

SelectColorResponse

The response to the SelectColorHandler.

## Enums

BeforeUnloadCause

Describes the cause of the `onbeforeunload` dialog.

# Class AlertParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for common dialog parameters.

```
public sealed class AlertParameters : CommonDialogParameters
```

**Inheritance**

object ← DialogParameters ← CommonDialogParameters ← AlertParameters

**Inherited Members**

CommonDialogParameters.Message , CommonDialogParameters.Title , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

OkActionText

Gets the localized text of the "OK" action.

Url

Gets the URL of the web page that requested to display the dialog.

# Property OkActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## OkActionText

Gets the localized text of the "OK" action.

```
public string OkActionText { get; }
```

## Property Value

string

The localized text of the "OK" action.

# Property Url

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the web page that requested to display the dialog.

```
public string Url { get; }
```

## Property Value

string

The URL of the web page that requested to display the dialog.

# Enum BeforeUnloadCause

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

Describes the cause of the `onbeforeunload` dialog.

```
public enum BeforeUnloadCause
```

## Fields

`LoadOtherPage = 1`

The other web page is about to be loaded.

`ReloadCurrentPage = 0`

The current web page is about to be reloaded.

# Class BeforeUnloadParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for common dialog parameters.

```
public sealed class BeforeUnloadParameters : CommonDialogParameters
```

**Inheritance**

object ← DialogParameters ← CommonDialogParameters ← BeforeUnloadParameters

**Inherited Members**

CommonDialogParameters.Message , CommonDialogParameters.Title , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Cause

Gets the action that caused the dialog to appear.

LeaveActionText

The localized text of the "Leave" action.

StayActionText

Gets the localized text of the "Stay" action.

# Property Cause

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cause

Gets the action that caused the dialog to appear.

```
public BeforeUnloadCause Cause { get; }
```

## Property Value

BeforeUnloadCause

The BeforeUnloadCause describing the action that caused the dialog to appear.

# Property LeaveActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## LeaveActionText

The localized text of the "Leave" action.

```
public string LeaveActionText { get; }
```

## Property Value

string

# Property StayActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## StayActionText

Gets the localized text of the "Stay" action.

```
public string StayActionText { get; }
```

## Property Value

string

The localized text of the "Stay" action.

# Class BeforeUnloadResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the BeforeUnloadHandler.

```
public sealed class BeforeUnloadResponse
```

**Inheritance**

object ← BeforeUnloadResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Leave()

Creates a BeforeUnloadResponse that notifies the browser that the user decided to leave the current page.

Stay()

Creates a BeforeUnloadResponse that notifies the browser that the user decided to stay on the current page.

# Method Leave

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Leave()

Creates a BeforeUnloadResponse that notifies the browser that the user decided to leave the current page.

```
public static BeforeUnloadResponse Leave()
```

## Returns

BeforeUnloadResponse

The BeforeUnloadResponse instance that can be used as a return value in BeforeUnloadHandler implementation.

# Method Stay

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Stay()

Creates a BeforeUnloadResponse that notifies the browser that the user decided to stay on the current page.

```
public static BeforeUnloadResponse Stay()
```

## Returns

BeforeUnloadResponse

The BeforeUnloadResponse instance that can be used as a return value in BeforeUnloadHandler implementation.

# Class CommonDialogParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for common dialog parameters.

```
public class CommonDialogParameters : DialogParameters
```

**Inheritance**

object ← DialogParameters ← CommonDialogParameters

**Derived**

AlertParameters, BeforeUnloadParameters, ConfirmParameters, OpenExternalAppParameters, PromptParameters, RepostFormParameters, SelectCertificateParameters

**Inherited Members**

DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Message

Gets the localized dialog message.

Title

Gets the localized dialog title.

# Property Message

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Message

Gets the localized dialog message.

```
public string Message { get; }
```

## Property Value

string

The localized dialog message.

# Property Title

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Title

Gets the localized dialog title.

```
public string Title { get; }
```

## Property Value

string

The localized dialog title.

# Class ConfirmParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for common dialog parameters.

```
public sealed class ConfirmParameters : CommonDialogParameters
```

**Inheritance**

object ← DialogParameters ← CommonDialogParameters ← ConfirmParameters

**Inherited Members**

CommonDialogParameters.Message , CommonDialogParameters.Title , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CancelActionText

Gets the localized text of the "Cancel" action.

OkActionText

Gets the localized text of the "OK" action.

Url

Gets the URL of the web page that requested to display the dialog.

# Property CancelActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## CancelActionText

Gets the localized text of the "Cancel" action.

```
public string CancelActionText { get; }
```

## Property Value

string

The localized text of the "Cancel" action.

# Property OkActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## OkActionText

Gets the localized text of the "OK" action.

```
public string OkActionText { get; }
```

## Property Value

string

The localized text of the "OK" action.

# Property Url

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the web page that requested to display the dialog.

```
public string Url { get; }
```

## Property Value

string

The URL of the web page that requested to display the dialog.

# Class ConfirmResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the ConfirmHandler.

```
public sealed class ConfirmResponse
```

**Inheritance**

object ← ConfirmResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a ConfirmResponse that notifies the browser that the JavaScript confirmation dialog should be canceled.

Ok()

Creates a ConfirmResponse that notifies the browser that the JavaScript confirmation dialog should be closed with the "OK" action.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a ConfirmResponse that notifies the browser that the JavaScript confirmation dialog should be canceled.

```
public static ConfirmResponse Cancel()
```

## Returns

ConfirmResponse

The ConfirmResponse instance that can be used as a return value in ConfirmHandler implementation.

# Method Ok

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Ok()

Creates a ConfirmResponse that notifies the browser that the JavaScript confirmation dialog should be closed with the "OK" action.

```
public static ConfirmResponse Ok()
```

## Returns

ConfirmResponse

The ConfirmResponse instance that can be used as a return value in ConfirmHandler implementation.

# Class DialogParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for dialog parameters.

```
public class DialogParameters
```

**Inheritance**

object ← DialogParameters

**Derived**

CommonDialogParameters, FileChooserParameters, SelectColorParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Browser

Gets the browser associated with the dialog.

# Property Browser

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Browser

Gets the browser associated with the dialog.

```
public IBrowser Browser { get; }
```

## Property Value

IBrowser

The browser associated with the dialog.

# Class FileChooserParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for the common file chooser parameters.

```
public abstract class FileChooserParameters : DialogParameters
```

### Inheritance

object ← DialogParameters ← FileChooserParameters

### Derived

FilteredFileChooserParameters, OpenDirectoryParameters, SaveAsPdfParameters, SaveFileParameters

### Inherited Members

DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class FilteredFileChooserParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for the common file chooser parameters.

```
public abstract class FilteredFileChooserParameters : FileChooserParameters
```

**Inheritance**

object ← DialogParameters ← FileChooserParameters ← FilteredFileChooserParameters

**Derived**

OpenFileParameters, OpenMultipleFilesParameters

**Inherited Members**

DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AcceptableExtensions

Gets the file extensions acceptable by the file chooser.

FilterDescription

Gets the meaningful (e.g. "Image files", "PDF File (.pdf)", etc.) description of the acceptable extensions.

# Property AcceptableExtensions

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## AcceptableExtensions

Gets the file extensions acceptable by the file chooser.

```
public IEnumerable<string> AcceptableExtensions { get; }
```

## Property Value

IEnumerable<string>

The file extensions acceptable by the file chooser. The collection is empty in case no `accept` attribute is specified or its value is empty.

## Remarks

The acceptable file extensions are based on the `accept` attribute value of the HTML input element. The extensions do not include the preceding dot character.

# Property FilterDescription

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## FilterDescription

Gets the meaningful (e.g. "Image files", "PDF File (.pdf)", etc.) description of the acceptable extensions.

```
public string FilterDescription { get; }
```

## Property Value

string

The meaningful description of the acceptable extensions. In case no extensions are provided, the value is empty, which implies that any file type is acceptable.

# Class OpenDirectoryParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The parameters of the OpenDirectoryHandler.

```
public sealed class OpenDirectoryParameters : FileChooserParameters
```

**Inheritance**

object ← DialogParameters ← FileChooserParameters ← OpenDirectoryParameters

**Inherited Members**

DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

InitialDirectory

Gets the initial directory that should be displayed by the file dialog box.

# Property InitialDirectory

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## InitialDirectory

Gets the initial directory that should be displayed by the file dialog box.

```
public string InitialDirectory { get; }
```

## Property Value

string

The initial directory that should be displayed by the file dialog box.

# Class OpenDirectoryResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the OpenDirectoryHandler.

```
public class OpenDirectoryResponse
```

**Inheritance**

object ← OpenDirectoryResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Canceled

Indicates whether the dialog was canceled.

SelectedDirectory

Gets the selected directory.

## Methods

Cancel()

Creates a OpenDirectoryResponse that notifies the browser that the dialog was canceled.

SelectDirectory(string)

Creates a OpenDirectoryResponse that notifies the browser that the directory was selected in the dialog.

# Property Canceled

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Canceled

Indicates whether the dialog was canceled.

```
public bool Canceled { get; }
```

## Property Value

bool

`true` if the dialog was canceled, `false` otherwise.

# Property SelectedDirectory

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectedDirectory

Gets the selected directory.

```
public string SelectedDirectory { get; }
```

## Property Value

string

The selected directory.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a OpenDirectoryResponse that notifies the browser that the dialog was canceled.

```
public static OpenDirectoryResponse Cancel()
```

## Returns

OpenDirectoryResponse

The OpenDirectoryResponse instance that can be used as a return value in OpenDirectoryHandler implementation.

# Method SelectDirectory

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectDirectory(string)

Creates a OpenDirectoryResponse that notifies the browser that the directory was selected in the dialog.

```
public static OpenDirectoryResponse SelectDirectory(string directory)
```

## Parameters

`directory` string

The selected directory.

## Returns

OpenDirectoryResponse

The OpenDirectoryResponse instance that can be used as a return value in OpenDirectoryHandler implementation.

# Class OpenExternalAppParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The parameters of the OpenExternalAppHandler.

```
public sealed class OpenExternalAppParameters : CommonDialogParameters
```

**Inheritance**

object ← DialogParameters ← CommonDialogParameters ← OpenExternalAppParameters

**Inherited Members**

CommonDialogParameters.Message , CommonDialogParameters.Title , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CancelActionText

Gets the localized text of the "Cancel" action.

OpenActionText

Gets the localized text of the "Open..." action.

# Property CancelActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## CancelActionText

Gets the localized text of the "Cancel" action.

```
public string CancelActionText { get; }
```

## Property Value

string

The localized text of the "Cancel" action.

# Property OpenActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## OpenActionText

Gets the localized text of the "Open..." action.

```
public string OpenActionText { get; }
```

## Property Value

string

The localized text of the "Open..." action.

# Class OpenExternalAppResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the OpenExternalAppHandler.

```
public sealed class OpenExternalAppResponse
```

**Inheritance**

object ← OpenExternalAppResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a OpenExternalAppResponse that notifies the browser that the link should not be opened in the external application.

Open()

Creates a OpenExternalAppResponse that notifies the browser that the link should be opened in the associated external application.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a OpenExternalAppResponse that notifies the browser that the link should not be opened in the external application.

```
public static OpenExternalAppResponse Cancel()
```

## Returns

OpenExternalAppResponse

The OpenExternalAppResponse instance that can be used as a return value in OpenExternalApp Handler implementation.

# Method Open

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Open()

Creates a OpenExternalAppResponse that notifies the browser that the link should be opened in the associated external application.

```
public static OpenExternalAppResponse Open()
```

### Returns

OpenExternalAppResponse

The OpenExternalAppResponse instance that can be used as a return value in OpenExternalApp Handler implementation.

### Remarks

If the application is not running, the operating system should launch the application and open the link in it.

# Class OpenFileParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The parameters of the OpenFileHandler.

```
public sealed class OpenFileParameters : FilteredFileChooserParameters
```

**Inheritance**

object ← DialogParameters ← FileChooserParameters ← FilteredFileChooserParameters ← OpenFileParameters

**Inherited Members**

FilteredFileChooserParameters.AcceptableExtensions , FilteredFileChooserParameters.FilterDescription , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

InitialDirectory

Gets the initial directory that should be displayed by the file dialog box.

# Property InitialDirectory

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## InitialDirectory

Gets the initial directory that should be displayed by the file dialog box.

```
public string InitialDirectory { get; }
```

## Property Value

string

The initial directory that should be displayed by the file dialog box.

# Class OpenFileResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the OpenFileHandler.

```
public sealed class OpenFileResponse
```

**Inheritance**

object ← OpenFileResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Canceled

Indicates whether the dialog was canceled.

SelectedFile

The selected file.

## Methods

Cancel()

Creates an OpenFileResponse that notifies the browser that the dialog was canceled.

SelectFile(string)

Creates an OpenFileResponse that notifies the browser that the file was selected in the file chooser dialog.

# Property Canceled

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Canceled

Indicates whether the dialog was canceled.

```
public bool Canceled { get; }
```

## Property Value

bool

# Property SelectedFile

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectedFile

The selected file.

```
public string SelectedFile { get; }
```

## Property Value

string

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates an OpenFileResponse that notifies the browser that the dialog was canceled.

```
public static OpenFileResponse Cancel()
```

## Returns

OpenFileResponse

The OpenFileResponse instance that can be used as a return value in OpenFileHandler implementation.

# Method SelectFile

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectFile(string)

Creates an OpenFileResponse that notifies the browser that the file was selected in the file chooser dialog.

```
public static OpenFileResponse SelectFile(string file)
```

## Parameters

`file` string

the selected file.

## Returns

OpenFileResponse

The OpenFileResponse instance that can be used as a return value in OpenFileHandler implementation.

# Class OpenMultipleFilesParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The parameters of the OpenMultipleFilesHandler.

```
public sealed class OpenMultipleFilesParameters : FilteredFileChooserParameters
```

### Inheritance

object ← DialogParameters ← FileChooserParameters ← FilteredFileChooserParameters ← OpenMultipleFilesParameters

### Inherited Members

FilteredFileChooserParameters.AcceptableExtensions , FilteredFileChooserParameters.FilterDescription , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class OpenMultipleFilesResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the OpenMultipleFilesHandler.

```
public sealed class OpenMultipleFilesResponse
```

**Inheritance**

object ← OpenMultipleFilesResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Canceled

Indicates whether the dialog was canceled.

SelectedFiles

Gets the selected files.

## Methods

Cancel()

Creates a OpenMultipleFilesResponse that notifies the browser that the dialog was canceled.

SelectFiles(params string[])

Creates a OpenMultipleFilesResponse that notifies the browser that the files were selected in the file chooser dialog.

# Property Canceled

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Canceled

Indicates whether the dialog was canceled.

```
public bool Canceled { get; }
```

## Property Value

bool

`true` if the dialog was canceled, `false` otherwise.

# Property SelectedFiles

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectedFiles

Gets the selected files.

```
public IEnumerable<string> SelectedFiles { get; }
```

## Property Value

IEnumerable<string>

A collection of the selected files.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a OpenMultipleFilesResponse that notifies the browser that the dialog was canceled.

```
public static OpenMultipleFilesResponse Cancel()
```

## Returns

OpenMultipleFilesResponse

The OpenMultipleFilesResponse instance that can be used as a return value in OpenMultipleFilesHandler implementation.

# Method SelectFiles

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectFiles(params string[])

Creates a OpenMultipleFilesResponse that notifies the browser that the files were selected in the file chooser dialog.

```
public static OpenMultipleFilesResponse SelectFiles(params string[] files)
```

### Parameters

`files` string[]

The selected files.

### Returns

OpenMultipleFilesResponse

The OpenMultipleFilesResponse instance that can be used as a return value in OpenMultipleFilesHandler implementation.

### Exceptions

ArgumentException

The `files` collection is null, empty, contains null or whitespace strings or contains strings that do not represent the existing files.

# Class PromptParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for common dialog parameters.

```
public sealed class PromptParameters : CommonDialogParameters
```

**Inheritance**

object ← DialogParameters ← CommonDialogParameters ← PromptParameters

**Inherited Members**

CommonDialogParameters.Message , CommonDialogParameters.Title , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CancelActionText

Gets the localized text of the "Cancel" action.

OkActionText

Gets the localized text of the "OK" action.

PromptText

Gets the default prompt text.

Url

Gets the URL of the web page that requested to display the dialog.

# Property CancelActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## CancelActionText

Gets the localized text of the "Cancel" action.

```
public string CancelActionText { get; }
```

## Property Value

string

The localized text of the "Cancel" action.

# Property OkActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## OkActionText

Gets the localized text of the "OK" action.

```
public string OkActionText { get; }
```

## Property Value

string

The localized text of the "OK" action.

# Property PromptText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## PromptText

Gets the default prompt text.

```
public string PromptText { get; }
```

## Property Value

string

The default text passed to the `window.prompt()` JavaScript function or an empty string, if no default value is passed.

# Property Url

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the web page that requested to display the dialog.

```
public string Url { get; }
```

## Property Value

string

The URL of the web page that requested to display the dialog.

# Class PromptResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the PromptHandler.

```
public sealed class PromptResponse
```

**Inheritance**

object ← PromptResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Canceled

Indicates whether the prompt dialog has been canceled.

Text

Gets the text input entered in the prompt dialog.

## Methods

Cancel()

Creates a PromptResponse that notifies the browser that the prompt dialog has been canceled.

SubmitText(string)

Creates a PromptResponse that notifies the browser that the user input was submitted in the prompt dialog.

# Property Canceled

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Canceled

Indicates whether the prompt dialog has been canceled.

```
public bool Canceled { get; }
```

## Property Value

bool

`true` if the dialog has been canceled, `false` otherwise.

# Property Text

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Text

Gets the text input entered in the prompt dialog.

```
public string Text { get; }
```

## Property Value

string

The text input entered in the prompt dialog.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a PromptResponse that notifies the browser that the prompt dialog has been canceled.

```
public static PromptResponse Cancel()
```

## Returns

PromptResponse

The PromptResponse instance that can be used as a return value in PromptHandler implementation.

# Method SubmitText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SubmitText(string)

Creates a PromptResponse that notifies the browser that the user input was submitted in the prompt dialog.

```
public static PromptResponse SubmitText(string text)
```

## Parameters

`text` string

The user input that was submitted in the prompt dialog.

## Returns

PromptResponse

The PromptResponse instance that can be used as a return value in PromptHandler implementation.

# Class RepostFormParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The base class for common dialog parameters.

```
public sealed class RepostFormParameters : CommonDialogParameters
```

**Inheritance**

object ← DialogParameters ← CommonDialogParameters ← RepostFormParameters

**Inherited Members**

CommonDialogParameters.Message , CommonDialogParameters.Title , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CancelActionText

Gets the localized text of the "Cancel" action.

RepostActionText

Gets the localized text of the "Repost" action.

# Property CancelActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## CancelActionText

Gets the localized text of the "Cancel" action.

```
public string CancelActionText { get; }
```

## Property Value

string

The localized text of the "Cancel" action.

# Property RepostActionText

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## RepostActionText

Gets the localized text of the "Repost" action.

```
public string RepostActionText { get; }
```

## Property Value

string

The localized text of the "Repost" action.

# Class RepostFormResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the RepostFormHandler.

```
public sealed class RepostFormResponse
```

**Inheritance**

object ← RepostFormResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a RepostFormResponse that notifies the browser that the page reloading should be canceled.

Repost()

Creates a RepostFormResponse that notifies the browser that resubmitting POST data is allowed.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a RepostFormResponse that notifies the browser that the page reloading should be canceled.

```
public static RepostFormResponse Cancel()
```

## Returns

RepostFormResponse

The RepostFormResponse instance that can be used as a return value in RepostFormHandler implementation.

# Method Repost

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Repost()

Creates a RepostFormResponse that notifies the browser that resubmitting POST data is allowed.

```
public static RepostFormResponse Repost()
```

## Returns

RepostFormResponse

The RepostFormResponse instance that can be used as a return value in RepostFormHandler implementation.

# Class SaveAsPdfParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SaveAsPdfHandler.

```
public sealed class SaveAsPdfParameters : FileChooserParameters
```

**Inheritance**

object ← DialogParameters ← FileChooserParameters ← SaveAsPdfParameters

**Inherited Members**

DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

SuggestedDirectory

Gets the suggested directory where to save the PDF file.

SuggestedFileName

Gets the suggested name of the PDF file to save.

# Property SuggestedDirectory

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SuggestedDirectory

Gets the suggested directory where to save the PDF file.

```
public string SuggestedDirectory { get; }
```

## Property Value

string

The suggested directory where to save the PDF file.

# Property SuggestedFileName

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SuggestedFileName

Gets the suggested name of the PDF file to save.

```
public string SuggestedFileName { get; }
```

## Property Value

string

The suggested name of the PDF file to save.

# Class SaveAsPdfResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the SaveAsPdfHandler.

```
public sealed class SaveAsPdfResponse
```

**Inheritance**

object ← SaveAsPdfResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Canceled

Indicates whether the dialog was canceled.

SelectedFile

Gets the selected file.

## Methods

Cancel()

Creates a SaveAsPdfResponse that notifies the browser that the dialog was canceled.

SaveToFile(string)

Creates a SaveAsPdfResponse that notifies the browser that the file was selected in the file chooser dialog.

# Property Canceled

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Canceled

Indicates whether the dialog was canceled.

```
public bool Canceled { get; }
```

## Property Value

bool

`true` if the dialog was canceled, `false` otherwise.

# Property SelectedFile

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectedFile

Gets the selected file.

```
public string SelectedFile { get; }
```

## Property Value

string

The selected file.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a SaveAsPdfResponse that notifies the browser that the dialog was canceled.

```
public static SaveAsPdfResponse Cancel()
```

## Returns

SaveAsPdfResponse

The SaveAsPdfResponse instance that can be used as a return value in SaveAsPdfHandler implementation.

# Method SaveToFile

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SaveToFile(string)

Creates a SaveAsPdfResponse that notifies the browser that the file was selected in the file chooser dialog.

```
public static SaveAsPdfResponse SaveToFile(string file)
```

### Parameters

`file` string

the selected file.

### Returns

SaveAsPdfResponse

The SaveAsPdfResponse instance that can be used as a return value in SaveAsPdfHandler implementation.

# Class SaveFileParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SaveFileHandler.

```
public class SaveFileParameters : FileChooserParameters
```

**Inheritance**

object ← DialogParameters ← FileChooserParameters ← SaveFileParameters

**Inherited Members**

DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AcceptableExtensions

Gets a list of the file extensions are based on the `accept` attribute value of the HTML input element acceptable by the file chooser.

FilterDescription

Gets a description of the acceptable extensions.

IsAllFilesEnabled

Indicates whether the file dialog should accept all files and show the "All files" filter.

SuggestedDirectory

Gets the suggested directory where to save the file.

SuggestedFileName

Gets the suggested name of the file to save.

# Property AcceptableExtensions

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## AcceptableExtensions

Gets a list of the file extensions are based on the `accept` attribute value of the HTML input element acceptable by the file chooser.

```
public List<string> AcceptableExtensions { get; }
```

## Property Value

List<string>

A List<T> of the file extensions acceptable by the file chooser. The extensions do not include the preceding dot character. The list is empty in case no `accept` attribute is specified or its value is empty.

# Property FilterDescription

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## FilterDescription

Gets a description of the acceptable extensions.

```
public string FilterDescription { get; }
```

## Property Value

string

A meaningful (e.g. "Image files", "PDF File (.pdf)", etc.) description of the acceptable extensions. In case no extensions are provided, the value is empty, which implies that any file type is acceptable.

# Property IsAllFilesEnabled

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## IsAllFilesEnabled

Indicates whether the file dialog should accept all files and show the "All files" filter.

```
public bool IsAllFilesEnabled { get; }
```

## Property Value

bool

`true` if the file dialog should accept all files and show the "All files" filter, `false` otherwise.

# Property SuggestedDirectory

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SuggestedDirectory

Gets the suggested directory where to save the file.

```
public string SuggestedDirectory { get; }
```

## Property Value

string

The suggested directory where to save the file.

# Property SuggestedFileName

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SuggestedFileName

Gets the suggested name of the file to save.

```
public string SuggestedFileName { get; }
```

## Property Value

string

The suggested name of the file to save.

# Class SaveFileResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the SaveFileHandler.

```
public class SaveFileResponse
```

**Inheritance**

object ← SaveFileResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Canceled

Indicates whether the dialog was canceled.

SelectedFile

Gets the selected file.

## Methods

Cancel()

Creates a SaveFileResponse that notifies the browser that the dialog was canceled.

SaveToFile(string)

Creates a SaveFileResponse that notifies the browser that the file was selected in the file chooser dialog.

# Property Canceled

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Canceled

Indicates whether the dialog was canceled.

```
public bool Canceled { get; }
```

## Property Value

bool

`true` if the dialog was canceled, `false` otherwise.

# Property SelectedFile

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectedFile

Gets the selected file.

```
public string SelectedFile { get; }
```

## Property Value

string

The selected file.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a SaveFileResponse that notifies the browser that the dialog was canceled.

```
public static SaveFileResponse Cancel()
```

## Returns

SaveFileResponse

The SaveFileResponse instance that can be used as a return value in SaveFileHandler implementation.

# Method SaveToFile

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SaveToFile(string)

Creates a SaveFileResponse that notifies the browser that the file was selected in the file chooser dialog.

```
public static SaveFileResponse SaveToFile(string file)
```

## Parameters

`file` string

the selected file.

## Returns

SaveFileResponse

The SaveFileResponse instance that can be used as a return value in SaveFileHandler implementation.

# Class SelectCertificateParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SelectCertificateHandler.

```
public sealed class SelectCertificateParameters : CommonDialogParameters
```

**Inheritance**

object ← DialogParameters ← CommonDialogParameters ← SelectCertificateParameters

**Inherited Members**

CommonDialogParameters.Message , CommonDialogParameters.Title , DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Certificates

Gets a collection of the certificates allowed by the server.

RequestIssuedByProxy

Indicates whether the request that requires the certificate selection is issued by an HTTPS proxy server.

SslOrigin

Gets the host and port of the server that requested the client authentication.

# Property Certificates

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Certificates

Gets a collection of the certificates allowed by the server.

```
public IEnumerable<Certificate> Certificates { get; }
```

## Property Value

IEnumerable<Certificate>

The collection of the certificates allowed by the server. The collection is empty when there are no matching certificates installed in the operating system. In this case you can either cancel the SSL client authentication, or provide a custom Certificate.

# Property RequestIssuedByProxy

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## RequestIssuedByProxy

Indicates whether the request that requires the certificate selection is issued by an HTTPS proxy server.

```
public bool RequestIssuedByProxy { get; }
```

## Property Value

bool

`true` if the request is issued by a proxy server, `false` otherwise.

# Property SslOrigin

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SslOrigin

Gets the host and port of the server that requested the client authentication.

```
public HostPortPair SslOrigin { get; }
```

## Property Value

HostPortPair

The host and port of the server that requested the client authentication.

# Class SelectCertificateResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to SelectCertificateHandler

```
public sealed class SelectCertificateResponse
```

**Inheritance**

object ← SelectCertificateResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a SelectCertificateResponse that notifies the browser that the SSL authorization should be cancelled.

Select(Certificate)

Creates a SelectCertificateResponse that notifies the browser that the given certificate should be selected.

Select(int)

Creates a SelectCertificateResponse that notifies the browser that the client certificate from the list with the given index should be selected.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a SelectCertificateResponse that notifies the browser that the SSL authorization should be cancelled.

```
public static SelectCertificateResponse Cancel()
```

## Returns

SelectCertificateResponse

The SelectCertificateResponse instance that can be used as a return value in SelectCertificateHandler implementation.

# Method Select

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Select(Certificate)

Creates a SelectCertificateResponse that notifies the browser that the given certificate should be selected.

```
public static SelectCertificateResponse Select(Certificate certificate)
```

### Parameters

`certificate` Certificate

The certificate that should be selected.

### Returns

SelectCertificateResponse

The SelectCertificateResponse instance that can be used as a return value in SelectCertificateHandler implementation.

## Select(int)

Creates a SelectCertificateResponse that notifies the browser that the client certificate from the list with the given index should be selected.

```
public static SelectCertificateResponse Select(int index)
```

### Parameters

`index` int

The index of the certificate that should be selected.

## Returns

SelectCertificateResponse

The SelectCertificateResponse instance that can be used as a return value in SelectCertificateHandler implementation.

# Class SelectColorParameters

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The parameters for the SelectColorHandler.

```
public class SelectColorParameters : DialogParameters
```

**Inheritance**

object ← DialogParameters ← SelectColorParameters

**Inherited Members**

DialogParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

DefaultColor

Gets the default color value.

# Property DefaultColor

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## DefaultColor

Gets the default color value.

```
public Color DefaultColor { get; }
```

## Property Value

Color

The default color value represents the value of the `value` attribute of the `<input type="color">` element. If this attribute is not specified, the default color is black.

# Class SelectColorResponse

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

The response to the SelectColorHandler.

```
public class SelectColorResponse
```

**Inheritance**

object ← SelectColorResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Cancelled

Indicates whether the dialog was cancelled.

SelectedColor

Gets the selected color.

## Methods

Cancel()

Creates a SelectColorResponse that notifies the browser that the user has cancelled color selection.

SelectColor(Color)

Creates a SelectColorResponse that notifies the browser that the user has selected a color.

ShowDialog()

Creates a SelectColorResponse that notifies the browser that the default Chromium color picker should be shown.

# Property Cancelled

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancelled

Indicates whether the dialog was cancelled.

```
public bool Cancelled { get; }
```

## Property Value

bool

`true` if the dialog was cancelled, `false` otherwise.

# Property SelectedColor

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectedColor

Gets the selected color.

```
public Color SelectedColor { get; }
```

## Property Value

Color

The selected color.

# Method Cancel

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a SelectColorResponse that notifies the browser that the user has cancelled color selection.

```
public static SelectColorResponse Cancel()
```

## Returns

SelectColorResponse

The SelectColorResponse instance that can be used as a return value in SelectColorHandler implementation.

# Method SelectColor

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## SelectColor(Color)

Creates a SelectColorResponse that notifies the browser that the user has selected a color.

```
public static SelectColorResponse SelectColor(Color color)
```

## Parameters

`color` Color

The selected color.

## Returns

SelectColorResponse

The SelectColorResponse instance that can be used as a return value in SelectColorHandler implementation.

# Method ShowDialog

Namespace: DotNetBrowser.Browser.Dialogs.Handlers

Assembly: DotNetBrowser.dll

## ShowDialog()

Creates a SelectColorResponse that notifies the browser that the default Chromium color picker should be shown.

```
public static SelectColorResponse ShowDialog()
```

## Returns

SelectColorResponse

The SelectColorResponse instance that can be used as a return value in SelectColorHandler implementation.

# Namespace DotNetBrowser.Browser.Events

## Classes

BrowserBecameResponsiveEventArgs

Event arguments for the BrowserBecameResponsive event.

BrowserBecameUnresponsiveEventArgs

Event arguments for the BrowserBecameUnresponsive event.

BrowserEventArgs

The base class for IBrowser events.

ConsoleMessageReceivedEventArgs

Event arguments for the ConsoleMessageReceived event.

CursorChangedEventArgs

The CursorChanged event data which can be used to update the cursor for the off-screen view.

FaviconChangedEventArgs

Event arguments for the FaviconChanged event.

FocusGainedEventArgs

Event arguments for the FocusGained event.

FocusLostEventArgs

Event arguments for the FocusLost event.

FocusRequestedEventArgs

Event arguments for the FocusRequested event.

FrameCreatedEventArgs

Event arguments for the FrameCreated event.

FrameDeletedEventArgs

Event arguments for the FrameDeleted event.

FrameEventArgs

The base class for IBrowser events related to IFrame.

MediaStreamCaptureStartedEventArgs

Event arguments for the MediaStreamCaptureStarted

MediaStreamCaptureStoppedEventArgs

Event arguments for the MediaStreamCaptureStopped

MediaStreamEventArgs
The base class for IBrowser events related to media stream events.

PrintPreviewClosedEventArgs
Event arguments for the PrintPreviewClosed

PrintPreviewOpenedEventArgs
Event arguments for the PrintPreviewOpened

RenderProcessTerminatedEventArgs
Event arguments for the RenderProcessTerminated event.

SessionStartedEventArgs
Event arguments for the SessionStarted event.

SpellCheckCompletedEventArgs
Event arguments for the SpellCheckCompleted event.

StatusChangedEventArgs
Event arguments for the StatusChanged event.

TitleChangedEventArgs
Event arguments for the TitleChanged event.

UpdateBoundsRequestedEventArgs
Event arguments for the UpdateBoundsRequested event.

## Enums

ConsoleMessageReceivedEventArgs.MessageLevel
Console message levels.

MediaStreamType
The media type of the captured stream.

TerminationStatus
The termination status of the render process.

# Class BrowserBecameResponsiveEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the BrowserBecameResponsive event.

```
public class BrowserBecameResponsiveEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← BrowserBecameResponsiveEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class BrowserBecameUnresponsiveEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the BrowserBecameUnresponsive event.

```
public class BrowserBecameUnresponsiveEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← BrowserBecameUnresponsiveEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class BrowserEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

The base class for IBrowser events.

```
public class BrowserEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs

**Derived**

BrowserBecameResponsiveEventArgs, BrowserBecameUnresponsiveEventArgs, ConsoleMessageReceivedEventArgs, FaviconChangedEventArgs, FocusGainedEventArgs, FocusLostEventArgs, FocusRequestedEventArgs, FrameEventArgs, MediaStreamEventArgs, PrintPreviewClosedEventArgs, PrintPreviewOpenedEventArgs, RenderProcessTerminatedEventArgs, SessionStartedEventArgs, StatusChangedEventArgs, TitleChangedEventArgs, FullScreenEnteredEventArgs, FullScreenExitedEventArgs, CastSessionStartFailedEventArgs, AudioEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Browser

Gets the IBrowser instance associated with the event.

# Property Browser

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with the event.

```
public IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with the event.

# Class ConsoleMessageReceivedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the ConsoleMessageReceived event.

```
public class ConsoleMessageReceivedEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← ConsoleMessageReceivedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Properties

Level

Gets the console message level.

LineNumber

Gets the line number that caused the message.

Message

Gets the console message text.

Source

Gets the source of the console message.

## Methods

ToString()

# Property Level

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Level

Gets the console message level.

```
public ConsoleMessageReceivedEventArgs.MessageLevel Level { get; }
```

## Property Value

ConsoleMessageReceivedEventArgs.MessageLevel

The console message level (Debug, Log, Warning or Error).

# Property LineNumber

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## LineNumber

Gets the line number that caused the message.

```
public int LineNumber { get; }
```

## Property Value

int

The line number that caused the message.

# Property Message

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Message

Gets the console message text.

```
public string Message { get; }
```

## Property Value

string

The console message text.

# Property Source

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Source

Gets the source of the console message.

```
public string Source { get; }
```

## Property Value

string

A string that describes the source of the console message. Usually the source is a path to a file that causes this console message.

# Method ToString

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Enum ConsoleMessageReceivedEventArgs.MessageLevel

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Console message levels.

```
public enum ConsoleMessageReceivedEventArgs.MessageLevel
```

## Fields

`Debug = 1`

Debug level.

`Error = 4`

Error level.

`Log = 2`

Log level.

`Unspecified = 0`

Unspecified level.

`Verbose = 5`

Verbose level.

`Warning = 3`

Warning level.

# Class CursorChangedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

The CursorChanged event data which can be used to update the cursor for the off-screen view.

```
public sealed class CursorChangedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← CursorChangedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Bitmap

The cursor image. Available for custom cursors only.

CursorType

The cursor type. If set to Custom, the cursor image and hotspot are provided in the corresponding fields.

Hotspot

The cursor hotspot. Available for custom cursors only.

# Property Bitmap

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Bitmap

The cursor image. Available for custom cursors only.

```
public Bitmap Bitmap { get; }
```

## Property Value

Bitmap

# Property CursorType

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## CursorType

The cursor type. If set to Custom, the cursor image and hotspot are provided in the corresponding fields.

```
public CursorType CursorType { get; }
```

## Property Value

CursorType

# Property Hotspot

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Hotspot

The cursor hotspot. Available for custom cursors only.

```
public Point Hotspot { get; }
```

## Property Value

Point

# Class FaviconChangedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the FaviconChanged event.

```
public class FaviconChangedEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← FaviconChangedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

NewFavicon

Gets the new favicon.

# Property NewFavicon

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## NewFavicon

Gets the new favicon.

```
public Bitmap NewFavicon { get; }
```

## Property Value

Bitmap

The new favicon.

# Class FocusGainedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the FocusGained event.

```
public class FocusGainedEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← FocusGainedEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class FocusLostEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the FocusLost event.

```
public class FocusLostEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← FocusLostEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class FocusRequestedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the FocusRequested event.

```
public class FocusRequestedEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← FocusRequestedEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class FrameCreatedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the FrameCreated event.

```
public class FrameCreatedEventArgs : FrameEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← FrameEventArgs ← FrameCreatedEventArgs

### Inherited Members

FrameEventArgs.Frame , BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class FrameDeletedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the FrameDeleted event.

```
public class FrameDeletedEventArgs : FrameEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← FrameEventArgs ← FrameDeletedEventArgs

### Inherited Members

FrameEventArgs.Frame , BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class FrameEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

The base class for IBrowser events related to IFrame.

```
public class FrameEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← FrameEventArgs

**Derived**

FrameCreatedEventArgs, FrameDeletedEventArgs, SpellCheckCompletedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Frame

Gets the IFrame instance associated with the event.

# Property Frame

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Frame

Gets the IFrame instance associated with the event.

```
public IFrame Frame { get; }
```

## Property Value

IFrame

The IFrame instance associated with the event.

# Class MediaStreamCaptureStartedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the MediaStreamCaptureStarted

```
public class MediaStreamCaptureStartedEventArgs : MediaStreamEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← MediaStreamEventArgs ← MediaStreamCaptureStartedEventArgs

### Inherited Members

MediaStreamEventArgs.MediaStreamType , BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class MediaStreamCaptureStoppedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the MediaStreamCaptureStopped

```
public class MediaStreamCaptureStoppedEventArgs : MediaStreamEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← MediaStreamEventArgs ← MediaStreamCaptureStoppedEventArgs

### Inherited Members

MediaStreamEventArgs.MediaStreamType , BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class MediaStreamEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

The base class for IBrowser events related to media stream events.

```
public class MediaStreamEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← MediaStreamEventArgs

**Derived**

MediaStreamCaptureStartedEventArgs, MediaStreamCaptureStoppedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

MediaStreamType

Gets the media type of the captured stream.

# Property MediaStreamType

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## MediaStreamType

Gets the media type of the captured stream.

```
public MediaStreamType MediaStreamType { get; }
```

## Property Value

MediaStreamType

The media type of the captured stream.

# Enum MediaStreamType

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

The media type of the captured stream.

```
public enum MediaStreamType
```

## Fields

`Audio = 0`

An audio stream from the microphone.

`Video = 1`

A video stream coming from the web camera.

# Class PrintPreviewClosedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the PrintPreviewClosed

```
public class PrintPreviewClosedEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← PrintPreviewClosedEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class PrintPreviewOpenedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the PrintPreviewOpened

```
public class PrintPreviewOpenedEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← PrintPreviewOpenedEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class RenderProcessTerminatedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the RenderProcessTerminated event.

```
public class RenderProcessTerminatedEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← RenderProcessTerminatedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Properties

ExitCode

Gets the render process exit code.

TerminationStatus

Gets the status of the render process termination.

## Methods

ToString()

# Property ExitCode

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## ExitCode

Gets the render process exit code.

```
public int ExitCode { get; }
```

## Property Value

int

The process exit code that indicates the reason of unexpected render process exit or crash.

# Property TerminationStatus

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## TerminationStatus

Gets the status of the render process termination.

```
public TerminationStatus TerminationStatus { get; }
```

## Property Value

TerminationStatus

The status of the render process termination.

# Method ToString

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class SessionStartedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the SessionStarted event.

```
public sealed class SessionStartedEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← SessionStartedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Session

Gets the ISession instance that has been started.

# Property Session

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Session

Gets the ISession instance that has been started.

```
public ISession Session { get; }
```

## Property Value

ISession

The ISession instance that has been started.

# Class SpellCheckCompletedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the SpellCheckCompleted event.

```
public class SpellCheckCompletedEventArgs : FrameEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← FrameEventArgs ← SpellCheckCompletedEventArgs

**Inherited Members**

FrameEventArgs.Frame , BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CheckedText

Gets the text that has been checked by the spell checker.

Results

Gets the collection of the spell checking results.

# Property CheckedText

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## CheckedText

Gets the text that has been checked by the spell checker.

```
public string CheckedText { get; }
```

## Property Value

string

The text that has been checked by the spell checker.

# Property Results

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Results

Gets the collection of the spell checking results.

```
public IEnumerable<SpellCheckingResult> Results { get; }
```

## Property Value

IEnumerable<SpellCheckingResult>

The collection of the spell checking results. Can be empty.

# Class StatusChangedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the StatusChanged event.

```
public class StatusChangedEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← StatusChangedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Properties

Text

Gets the new status text.

## Methods

ToString()

# Property Text

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Text

Gets the new status text.

```
public string Text { get; }
```

## Property Value

string

The new status text. Can be an empty string.

# Method ToString

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Enum TerminationStatus

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

The termination status of the render process.

```
public enum TerminationStatus
```

## Fields

`Abnormal = 2`

Abnormal termination.

`LaunchFailed = 6`

Launch failed.

`Normal = 1`

Normal termination.

`OutOfMemory = 7`

Out of memory.

`ProcessCrashed = 4`

Process crashed.

`ProcessWasKilled = 3`

Process was killed.

`StillRunning = 5`

Still running.

`Unspecified = 0`

Termination status unspecified.

# Class TitleChangedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the TitleChanged event.

```
public class TitleChangedEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← TitleChangedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Properties

Title

Gets the new title.

## Methods

ToString()

# Property Title

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Title

Gets the new title.

```
public string Title { get; }
```

## Property Value

string

The new title.

# Method ToString

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class UpdateBoundsRequestedEventArgs

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

Event arguments for the UpdateBoundsRequested event.

```
public sealed class UpdateBoundsRequestedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← UpdateBoundsRequestedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Bounds

Gets the new bounds.

# Property Bounds

Namespace: DotNetBrowser.Browser.Events

Assembly: DotNetBrowser.dll

## Bounds

Gets the new bounds.

```
public Rectangle Bounds { get; }
```

## Property Value

Rectangle

The new bounds in screen coordinates.

# Namespace DotNetBrowser.Browser.FullScreen

## Interfaces

IFullScreen

A service that is used for controlling the browser's fullscreen mode.

# Interface IFullScreen

Namespace: DotNetBrowser.Browser.FullScreen

Assembly: DotNetBrowser.dll

A service that is used for controlling the browser's fullscreen mode.

```
public interface IFullScreen : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Methods

Exit()

Tells the browser to exit fullscreen mode.

## Events

Entered

Occurs when the browser instance has been toggled into full-screen mode.

Exited

Occurs when the browser instance has been toggled out of full-screen mode.

# Method Exit

Namespace: DotNetBrowser.Browser.FullScreen

Assembly: DotNetBrowser.dll

## Exit()

Tells the browser to exit fullscreen mode.

```
void Exit()
```

## Remarks

This method iterates over the frames in the browser and exits fullscreen mode for each of them. This also includes internal frames of the PDF viewer that are not accessible through the public API. If there is no element being presented in fullscreen mode, this method does nothing.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Event Entered

Namespace: DotNetBrowser.Browser.FullScreen

Assembly: DotNetBrowser.dll

Occurs when the browser instance has been toggled into full-screen mode.

```
event EventHandler<FullScreenEnteredEventArgs> Entered
```

## Returns

EventHandler<FullScreenEnteredEventArgs>

Occurs when the browser instance has been toggled into full-screen mode.

# Event Exited

Namespace: DotNetBrowser.Browser.FullScreen

Assembly: DotNetBrowser.dll

Occurs when the browser instance has been toggled out of full-screen mode.

```
event EventHandler<FullScreenExitedEventArgs> Exited
```

## Returns

EventHandler<FullScreenExitedEventArgs>

Occurs when the browser instance has been toggled out of full-screen mode.

# Namespace DotNetBrowser.Browser.FullScreen.Events

## Classes

FullScreenEnteredEventArgs

Event arguments for the Entered event.

FullScreenExitedEventArgs

Event arguments for the Exited event.

# Class FullScreenEnteredEventArgs

Namespace: DotNetBrowser.Browser.FullScreen.Events

Assembly: DotNetBrowser.dll

Event arguments for the Entered event.

```
public class FullScreenEnteredEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← FullScreenEnteredEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class FullScreenExitedEventArgs

Namespace: DotNetBrowser.Browser.FullScreen.Events

Assembly: DotNetBrowser.dll

Event arguments for the Exited event.

```
public class FullScreenExitedEventArgs : BrowserEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← FullScreenExitedEventArgs

### Inherited Members

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Namespace DotNetBrowser.Browser.Handlers

## Classes

BrowserParameters

The base class for IBrowser handlers parameters.

CreatePopupParameters

The parameters of the CreatePopupHandler.

CreatePopupResponse

The response to the CreatePopupHandler.

FrameParameters

Represents the parameters of IBrowser-related handlers associated with IFrame.

InjectCssParameters

The parameters of the InjectCssHandler.

InjectCssResponse

The response to the InjectCssHandler.

InjectJsParameters

The parameters of the InjectJsHandler.

OpenExtensionActionPopupParameters

The parameters of the OpenExtensionActionPopupHandler.

OpenPopupParameters

The parameters of the OpenPopupHandler.

RequestPdfDocumentPasswordParameters

The parameters of the RequestPdfDocumentPasswordHandler.

RequestPdfDocumentPasswordResponse

The response to the RequestPdfDocumentPasswordHandler.

RequestPrintParameters

The parameters of the RequestPrintHandler.

RequestPrintResponse

The response to the RequestPrintHandler.

ShowContextMenuParameters

The parameters of the ShowContextMenuHandler.

ShowContextMenuResponse

The response to the ShowContextMenuHandler.

# Class BrowserParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The base class for IBrowser handlers parameters.

```
public class BrowserParameters
```

**Inheritance**

object ← BrowserParameters

**Derived**

CreatePopupParameters, FrameParameters, OpenExtensionActionPopupParameters, OpenPopupParameters, RequestPrintParameters, ShowContextMenuParameters, StartSessionParameters, SaveCreditCardParameters, StartPresentationParameters, PresentationRequest, StartDownloadParameters, SavePasswordParameters, UpdatePasswordParameters, SaveUserDataProfileParameters, UpdateUserDataProfileParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Browser

Gets the IBrowser instance the handler is associated with.

# Property Browser

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance the handler is associated with.

```
public IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance the handler is associated with.

# Class CreatePopupParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The parameters of the CreatePopupHandler.

```
public sealed class CreatePopupParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← CreatePopupParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object)

## Properties

TargetName

Gets the popup window target name that was passed through the `window.open()` JavaScript function.

TargetUrl

Gets the target URL that will be loaded in the popup browser.

## Methods

ToString()

# Property TargetName

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## TargetName

Gets the popup window target name that was passed through the `window.open()` JavaScript function.

```
public string TargetName { get; }
```

## Property Value

string

The popup window target name that was passed through the `window.open()` JavaScript function.

# Property TargetUrl

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## TargetUrl

Gets the target URL that will be loaded in the popup browser.

```
public string TargetUrl { get; }
```

## Property Value

string

The target URL that will be loaded in the popup browser.

# Method ToString

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class CreatePopupResponse

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The response to the CreatePopupHandler.

```
public sealed class CreatePopupResponse
```

**Inheritance**

object ← CreatePopupResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Create()

Creates a CreatePopupResponse that notifies the browser that the popup can be created.

Suppress()

Creates a CreatePopupResponse that notifies the browser that the popup creation should be suppressed.

# Method Create

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Create()

Creates a CreatePopupResponse that notifies the browser that the popup can be created.

```
public static CreatePopupResponse Create()
```

## Returns

CreatePopupResponse

The CreatePopupResponse instance that can be used as a return value in CreatePopupHandler implementation.

# Method Suppress

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Suppress()

Creates a CreatePopupResponse that notifies the browser that the popup creation should be suppressed.

```
public static CreatePopupResponse Suppress()
```

## Returns

CreatePopupResponse

The CreatePopupResponse instance that can be used as a return value in CreatePopupHandler implementation.

# Class FrameParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

Represents the parameters of IBrowser-related handlers associated with IFrame.

```
public class FrameParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← FrameParameters

**Derived**

InjectCssParameters, InjectJsParameters, RequestPdfDocumentPasswordParameters, ConvertJsNameParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Frame

Gets the IFrame instance associated with the handler.

# Property Frame

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Frame

Gets the IFrame instance associated with the handler.

```
public IFrame Frame { get; }
```

## Property Value

IFrame

The IFrame instance associated with the handler.

# Class InjectCssParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The parameters of the InjectCssHandler.

```
public class InjectCssParameters : FrameParameters
```

### Inheritance

object ← BrowserParameters ← FrameParameters ← InjectCssParameters

### Inherited Members

FrameParameters.Frame , BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class InjectCssResponse

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The response to the InjectCssHandler.

```
public class InjectCssResponse
```

**Inheritance**

object ← InjectCssResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Inject(string)

Creates an InjectCssResponse that injects the given `customStylesheet` string that represents a custom stylesheet (CSS) into the document that is loaded in this browser instance.

Proceed()

Creates a InjectCssResponse that notifies the browser that it can proceed without injecting a custom stylesheet.

# Method Inject

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Inject(string)

Creates an InjectCssResponse that injects the given `customStylesheet` string that represents a custom stylesheet (CSS) into the document that is loaded in this browser instance.

```
public static InjectCssResponse Inject(string customStylesheet)
```

### Parameters

`customStylesheet` string

The CSS code that will be injected into the document.

### Returns

InjectCssResponse

The InjectCssResponse instance that can be used as a return value in InjectCssHandler implementation.

### Remarks

If the CSS property defined in the given `customStylesheet` string already exists on the loaded HTML document, then the existing CSS property won't be overridden. The CSS properties defined in the given `customStylesheet` string will be applied only if these properties aren't defined on the loaded document.

### Exceptions

ArgumentException

The `customStylesheet` is null, empty or blank.

# Method Proceed

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Proceed()

Creates a InjectCssResponse that notifies the browser that it can proceed without injecting a custom stylesheet.

```
public static InjectCssResponse Proceed()
```

## Returns

InjectCssResponse

The InjectCssResponse instance that can be used as a return value in InjectCssHandler implementation.

# Class InjectJsParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The parameters of the InjectJsHandler.

```
public class InjectJsParameters : FrameParameters
```

### Inheritance

object ← BrowserParameters ← FrameParameters ← InjectJsParameters

### Inherited Members

FrameParameters.Frame , BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class OpenExtensionActionPopupParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The parameters of the OpenExtensionActionPopupHandler.

```
public sealed class OpenExtensionActionPopupParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← OpenExtensionActionPopupParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ExtensionAction

Gets the extension action that requests opening the popup.

PopupBrowser

Gets the created popup browser.

# Property ExtensionAction

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## ExtensionAction

Gets the extension action that requests opening the popup.

```
public IExtensionAction ExtensionAction { get; }
```

## Property Value

IExtensionAction

# Property PopupBrowser

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## PopupBrowser

Gets the created popup browser.

```
public IBrowser PopupBrowser { get; }
```

## Property Value

IBrowser

# Class OpenPopupParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The parameters of the OpenPopupHandler.

```
public sealed class OpenPopupParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← OpenPopupParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object)

## Properties

PopupBrowser

Gets the browser instance which should be opened in a popup window.

Rectangle

Gets the initial bounds of the popup.

## Methods

ToString()

# Property PopupBrowser

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## PopupBrowser

Gets the browser instance which should be opened in a popup window.

```
public IBrowser PopupBrowser { get; }
```

## Property Value

IBrowser

The browser instance which should be opened in a popup window.

# Property Rectangle

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Rectangle

Gets the initial bounds of the popup.

```
public Rectangle Rectangle { get; }
```

## Property Value

Rectangle

The initial bounds of the popup. If the popup is about to open as a result of the `window.open()` JavaScript function, this value represents the corresponding attributes of the `WindowFeatures` parameter passed to this function. The value is empty in case no bounds were passed.

# Method ToString

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class RequestPdfDocumentPasswordParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The parameters of the RequestPdfDocumentPasswordHandler.

```
public sealed class RequestPdfDocumentPasswordParameters : FrameParameters
```

**Inheritance**

object ← BrowserParameters ← FrameParameters ← RequestPdfDocumentPasswordParameters

**Inherited Members**

FrameParameters.Frame , BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Url

Gets the URL of the encrypted PDF document.

# Property Url

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the encrypted PDF document.

```
public string Url { get; }
```

## Property Value

string

The URL of the encrypted PDF document.

# Class RequestPdfDocumentPasswordResponse

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The response to the RequestPdfDocumentPasswordHandler.

```
public sealed class RequestPdfDocumentPasswordResponse
```

**Inheritance**

object ← RequestPdfDocumentPasswordResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Cancel

Creates a RequestPdfDocumentPasswordResponse that notifies the browser to cancels the password request.

ShowPasswordDialog

Creates a RequestPdfDocumentPasswordResponse that notifies the browser to use the standard PDF viewer password dialog UI to load the encrypted PDF document.

## Methods

Password(string)

Creates a RequestPdfDocumentPasswordResponse that notifies the browser to use the `password` password to load the encrypted PDF document.

# Field Cancel

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

Creates a RequestPdfDocumentPasswordResponse that notifies the browser to cancels the password request.

```
public static RequestPdfDocumentPasswordResponse Cancel
```

## Returns

RequestPdfDocumentPasswordResponse

The RequestPdfDocumentPasswordResponse instance that can be used as a return value in RequestPdfDocumentPasswordHandler implementation.

## Remarks

Without a password, the browser will fail to load the frame with the encrypted PDF document and a FrameLoadFailed event will be fired for that frame.

# Field ShowPasswordDialog

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

Creates a RequestPdfDocumentPasswordResponse that notifies the browser to use the standard PDF viewer password dialog UI to load the encrypted PDF document.

```
public static RequestPdfDocumentPasswordResponse ShowPasswordDialog
```

## Returns

RequestPdfDocumentPasswordResponse

The RequestPdfDocumentPasswordResponse instance that can be used as a return value in RequestPdfDocumentPasswordHandler implementation.

# Method Password

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Password(string)

Creates a RequestPdfDocumentPasswordResponse that notifies the browser to use the `password` password to load the encrypted PDF document.

```
public static RequestPdfDocumentPasswordResponse Password(string password)
```

## Parameters

`password` string

The password for the encrypted PDF document.

## Returns

RequestPdfDocumentPasswordResponse

The RequestPdfDocumentPasswordResponse instance that can be used as a return value in RequestPdfDocumentPasswordHandler implementation.

## Remarks

If the provided password is incorrect, the browser will fail to load the frame with the PDF document.

## Exceptions

ArgumentException

The `password` is null, empty, or contains only blank characters.

# Class RequestPrintParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The parameters of the RequestPrintHandler.

```
public class RequestPrintParameters : BrowserParameters
```

### Inheritance

object ← BrowserParameters ← RequestPrintParameters

### Inherited Members

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class RequestPrintResponse

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The response to the RequestPrintHandler.

```
public sealed class RequestPrintResponse
```

**Inheritance**

object ← RequestPrintResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a RequestPrintResponse that notifies the browser that the printing should be canceled.

Print()

Creates a RequestPrintResponse that notifies the browser to proceed with printing and configure the print settings via the PrintHtmlContentHandler and PrintPdfContentHandler handlers.

ShowPrintPreview()

Creates a RequestPrintResponse that notifies the browser to display the Print Preview dialog.

# Method Cancel

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a RequestPrintResponse that notifies the browser that the printing should be canceled.

```
public static RequestPrintResponse Cancel()
```

## Returns

RequestPrintResponse

The RequestPrintResponse instance that can be used as a return value in RequestPrintHandler implementation.

# Method Print

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Print()

Creates a RequestPrintResponse that notifies the browser to proceed with printing and configure the print settings via the PrintHtmlContentHandler and PrintPdfContentHandler handlers.

```
public static RequestPrintResponse Print()
```

## Returns

RequestPrintResponse

The RequestPrintResponse instance that can be used as a return value in RequestPrintHandler implementation.

# Method ShowPrintPreview

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## ShowPrintPreview()

Creates a RequestPrintResponse that notifies the browser to display the Print Preview dialog.

```
public static RequestPrintResponse ShowPrintPreview()
```

## Returns

RequestPrintResponse

The RequestPrintResponse instance that can be used as a return value in RequestPrintHandler implementation.

# Class ShowContextMenuParameters

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The parameters of the ShowContextMenuHandler.

```
public sealed class ShowContextMenuParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← ShowContextMenuParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object)

## Properties

CustomMenuItems

Gets the collection of the custom context menu items.

FrameCharset

Gets the character encoding of the frame on which the menu is invoked.

FrameUrl

Gets the URL of the sub-frame that the context menu was invoked on.

IsMainFrame

Indicates whether the context menu is invoked on the main frame.

LinkText

Gets the text associated with the link.

LinkUrl

Gets the URL of the link that encloses the node the context menu was invoked on.

Location

Gets the context menu location. This location is related to the frame the context menu is invoked on.

MediaType

Gets the media type of the node the context menu is being invoked on.

PageUrl
Gets the URL of the top level page that the context menu was invoked on.

SelectedText
Gets the text of the selection that the context menu was invoked on.

SourceUrl
Gets the source URL for the element that the context menu was invoked on.

SpellCheckMenu
Gets the spell check submenu.

## Methods

ToString()
Represent object as string

# Property CustomMenuItems

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## CustomMenuItems

Gets the collection of the custom context menu items.

```
public IEnumerable<ContextMenuItem> CustomMenuItems { get; }
```

## Property Value

IEnumerable<ContextMenuItem>

The collection of the custom context menu items. Can be empty if the context menu does not contain any custom items.

# Property FrameCharset

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## FrameCharset

Gets the character encoding of the frame on which the menu is invoked.

```
public string FrameCharset { get; }
```

## Property Value

string

The character encoding of the frame.

# Property FrameUrl

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## FrameUrl

Gets the URL of the sub-frame that the context menu was invoked on.

```
public string FrameUrl { get; }
```

## Property Value

string

The URL of the sub-frame.

# Property IsMainFrame

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## IsMainFrame

Indicates whether the context menu is invoked on the main frame.

```
public bool IsMainFrame { get; }
```

## Property Value

bool

`true` if the context menu is invoked on the main frame, `false` otherwise.

# Property LinkText

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## LinkText

Gets the text associated with the link.

```
public string LinkText { get; }
```

## Property Value

string

The text associated with the link. May be an empty string if the link contains an image.

# Property LinkUrl

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## LinkUrl

Gets the URL of the link that encloses the node the context menu was invoked on.

```
public string LinkUrl { get; }
```

## Property Value

string

The URL of the link that encloses the node the context menu was invoked on.

# Property Location

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Location

Gets the context menu location. This location is related to the frame the context menu is invoked on.

```
public Point Location { get; }
```

## Property Value

Point

The context menu location.

# Property MediaType

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## MediaType

Gets the media type of the node the context menu is being invoked on.

```
public MediaType MediaType { get; }
```

## Property Value

MediaType

The media type of the node the context menu is being invoked on.

# Property PageUrl

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## PageUrl

Gets the URL of the top level page that the context menu was invoked on.

```
public string PageUrl { get; }
```

## Property Value

string

The URL of the top level page.

# Property SelectedText

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## SelectedText

Gets the text of the selection that the context menu was invoked on.

```
public string SelectedText { get; }
```

## Property Value

string

the text of the selection.

# Property SourceUrl

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## SourceUrl

Gets the source URL for the element that the context menu was invoked on.

```
public string SourceUrl { get; }
```

## Property Value

string

The source URL for the element. Example of the elements with source URLs are `img`, `audio`, and `video`.

# Property SpellCheckMenu

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## SpellCheckMenu

Gets the spell check submenu.

```
public SpellCheckMenu SpellCheckMenu { get; }
```

## Property Value

SpellCheckMenu

the spell check submenu containing suggestions for replacing misspelled word.

# Method ToString

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## ToString()

Represent object as string

```
public override string ToString()
```

## Returns

string

a string that represents the object

# Class ShowContextMenuResponse

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

The response to the ShowContextMenuHandler.

```
public sealed class ShowContextMenuResponse
```

**Inheritance**

object ← ShowContextMenuResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Close()

Creates a ShowContextMenuResponse that notifies the browser the context menu should be closed.

Select(ContextMenuItem)

Creates a ShowContextMenuResponse that notifies the browser that the `item` should be selected. The browser will execute the corresponding functionality of the selected context menu item. The context menu state will be changed to closed.

# Method Close

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Close()

Creates a ShowContextMenuResponse that notifies the browser the context menu should be closed.

```
public static ShowContextMenuResponse Close()
```

## Returns

ShowContextMenuResponse

The ShowContextMenuResponse instance that can be used as a return value in ShowContextMenuHandler implementation.

# Method Select

Namespace: DotNetBrowser.Browser.Handlers

Assembly: DotNetBrowser.dll

## Select(ContextMenuItem)

Creates a ShowContextMenuResponse that notifies the browser that the `item` should be selected. The browser will execute the corresponding functionality of the selected context menu item. The context menu state will be changed to closed.

```
public static ShowContextMenuResponse Select(ContextMenuItem item)
```

### Parameters

`item` ContextMenuItem

The menu item that should be selected.

### Returns

ShowContextMenuResponse

The ShowContextMenuResponse instance that can be used as a return value in ShowContextMenuHandler implementation.

### Exceptions

ArgumentException

The `item` is null.

# Namespace DotNetBrowser.Cache

## Interfaces

IHttpAuthCache

An HTTP Authentication cache service.

IHttpCache

An HTTP cache service.

# Interface IHttpAuthCache

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

An HTTP Authentication cache service.

```
public interface IHttpAuthCache : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Remarks

This service stores HTTP authentication identities and challenge info.

## Properties

Engine
: Gets the IEngine instance associated with this object.

Profile
: Gets the IProfile instance associated with this object.

## Methods

Clear()
: Clears all added HTTP authentication entries.

# Property Engine

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Profile

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Method Clear

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

## Clear()

Clears all added HTTP authentication entries.

```
void Clear()
```

## Exceptions

ObjectDisposedException

The IHttpCache has already been disposed.

# Interface IHttpCache

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

An HTTP cache service.

```
public interface IHttpCache : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Remarks

By default, HTTP cache stores the resources fetched from the web on a disk or in the memory. Chromium itself decides how to cache the resources for optimal performance.

The memory cache stores and loads the resources to and from the process memory (RAM). It is a fast, but non-persistent way.

The disk cache is persistent. The cached resources are stored and loaded to and from the disk. The disk cache is stored in the HttpCache folder in the User Data directory.

## Properties

Engine
: Gets the IEngine instance associated with this object.

Profile
: Gets the IProfile instance associated with this object.

## Methods

Clear()
: Marks all the cache entries for deletion. The deletion of the entries is performed asynchronously by the Chromium engine itself. If the engine is closed during this task executing - the operation will be canceled.

# Events

DiskCacheCleared

Occurs when the disk cache has been cleared.

# Property Engine

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Profile

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Method Clear

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

## Clear()

Marks all the cache entries for deletion. The deletion of the entries is performed asynchronously by the Chromium engine itself. If the engine is closed during this task executing - the operation will be canceled.

```
Task Clear()
```

### Returns

Task

a Task which is completed when the disk cache is cleared.

### Exceptions

ObjectDisposedException

The IHttpCache has already been disposed.

# Event DiskCacheCleared

Namespace: DotNetBrowser.Cache

Assembly: DotNetBrowser.dll

Occurs when the disk cache has been cleared.

```
event EventHandler<DiskCacheClearedEventArgs> DiskCacheCleared
```

## Returns

EventHandler<DiskCacheClearedEventArgs>

Occurs when the disk cache has been cleared.

# Namespace DotNetBrowser.Cache.Events

## Classes

DiskCacheClearedEventArgs

Event arguments for the DiskCacheCleared event.

# Class DiskCacheClearedEventArgs

Namespace: DotNetBrowser.Cache.Events

Assembly: DotNetBrowser.dll

Event arguments for the DiskCacheCleared event.

```
public class DiskCacheClearedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← DiskCacheClearedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

HttpCache

The IHttpCache instance associated with the event.

# Property HttpCache

Namespace: DotNetBrowser.Cache.Events

Assembly: DotNetBrowser.dll

## HttpCache

The IHttpCache instance associated with the event.

```
public IHttpCache HttpCache { get; }
```

## Property Value

IHttpCache

# Namespace DotNetBrowser.Capture

## Classes

Source
: The source for the content capture session.

Sources
: Provides the access to the sources available for content capture.

## Interfaces

ICapture
: A service that can be used for listening and handling capturing sessions.

ISession
: A content capture session.

## Enums

AudioMode
: The mode of capturing audio during the content capture session.

SourceType
: The type of the content capture source.

# Enum AudioMode

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

The mode of capturing audio during the content capture session.

```
public enum AudioMode
```

## Fields

`Capture = 1`

The audio content should be captured along with the video content.

`Ignore = 0`

The audio content should not be captured along with the video content.

# Interface ICapture

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

A service that can be used for listening and handling capturing sessions.

```
public interface ICapture
```

## Properties

Sessions
: Gets all active capture sessions of the current browser.

StartSessionHandler
: Gets or sets a handler that is used when the browser is about to start capturing sessions.

## Events

SessionStarted
: Occurs when a capture session is started for this browser instance.

# Property Sessions

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## Sessions

Gets all active capture sessions of the current browser.

```
IReadOnlyList<ISession> Sessions { get; }
```

## Property Value

IReadOnlyList<ISession>

All the capture sessions of the browser. If the current browser did not have any capture sessions, this will be an empty collection.

# Property StartSessionHandler

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## StartSessionHandler

Gets or sets a handler that is used when the browser is about to start capturing sessions.

```
IHandler<StartSessionParameters, StartSessionResponse> StartSessionHandler { get; set; }
```

## Property Value

IHandler<StartSessionParameters, StartSessionResponse>

## Remarks

Use the ShowSelectSourceDialog() method to display the default dialog for choosing the capture source.

Use theSelectSource(Source, AudioMode) method to use the given capture source.

Use the SelectSource(IBrowser, AudioMode) method to use the given browser as the capture source.

Use theCancel() method if the capture session request should be canceled.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel() will be used.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Event SessionStarted

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

Occurs when a capture session is started for this browser instance.

```
event EventHandler<SessionStartedEventArgs> SessionStarted
```

## Returns

EventHandler<SessionStartedEventArgs>

Occurs when a capture session is started for this browser instance.

# Interface ISession

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

A content capture session.

```
public interface ISession : IDisposable, IAutoDisposable
```

**Inherited Members**

IDisposable.Dispose() , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

IsActive

Indicates whether this capture session is active.

Source

Gets the Source instance for this capture session.

## Methods

Stop()

Stops the capture session.

## Events

Stopped

Occurs when the browser has stopped a content capture session.

# Property IsActive

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## IsActive

Indicates whether this capture session is active.

```
bool IsActive { get; }
```

## Property Value

bool

`true` if the capture session is active, `false` otherwise.

## Exceptions

ObjectDisposedException

The ISession has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Source

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## Source

Gets the Source instance for this capture session.

```
Source Source { get; }
```

## Property Value

Source

The Source instance for this capture session.

# Method Stop

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## Stop()

Stops the capture session.

```
void Stop()
```

## Remarks

This method has no effect when this capture session is already stopped.

## Exceptions

ObjectDisposedException

The ISession object has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Event Stopped

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

Occurs when the browser has stopped a content capture session.

```
event EventHandler<SessionStoppedEventArgs> Stopped
```

## Returns

EventHandler<SessionStoppedEventArgs>

Occurs when the browser has stopped a content capture session.

# Class Source

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

The source for the content capture session.

```
public sealed class Source
```

**Inheritance**

object ← Source

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Name

Gets the name of the capture source.

Thumbnail

Gets the Bitmap instance that contains the image of the capture source.

Type

Gets the type of this capture source.

# Property Name

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## Name

Gets the name of the capture source.

```
public string Name { get; }
```

## Property Value

string

The name of the capture source. The name is not guaranteed to be unique.

# Property Thumbnail

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## Thumbnail

Gets the Bitmap instance that contains the image of the capture source.

```
public Bitmap Thumbnail { get; }
```

## Property Value

Bitmap

The Bitmap instance that contains the image of the capture source.

# Property Type

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## Type

Gets the type of this capture source.

```
public SourceType Type { get; }
```

## Property Value

SourceType

The type of this capture source.

# Enum SourceType

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

The type of the content capture source.

```
public enum SourceType
```

## Fields

`ApplicationWindow = 2`

The content is captured from an application window.

`Browser = 3`

The content is captured from a IBrowser instance.

`Screen = 1`

The content is captured from a screen.

`Unspecified = 0`

The content is captured from an unspecified source.

# Class Sources

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

Provides the access to the sources available for content capture.

```
public sealed class Sources
```

**Inheritance**

object ← Sources

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ApplicationWindows

Gets all of the available application windows for capture.

Browsers

Gets all of the available browsers for capture.

Screens

Gets all of the available screens for capture.

# Property ApplicationWindows

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## ApplicationWindows

Gets all of the available application windows for capture.

```
public IReadOnlyList<Source> ApplicationWindows { get; }
```

## Property Value

IReadOnlyList<Source>

# Property Browsers

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## Browsers

Gets all of the available browsers for capture.

```
public IReadOnlyList<Source> Browsers { get; }
```

## Property Value

IReadOnlyList<Source>

# Property Screens

Namespace: DotNetBrowser.Capture

Assembly: DotNetBrowser.dll

## Screens

Gets all of the available screens for capture.

```
public IReadOnlyList<Source> Screens { get; }
```

## Property Value

IReadOnlyList<Source>

# Namespace DotNetBrowser.Capture.Events

## Classes

SessionStoppedEventArgs

Event arguments for the Stopped event.

# Class SessionStoppedEventArgs

Namespace: DotNetBrowser.Capture.Events

Assembly: DotNetBrowser.dll

Event arguments for the Stopped event.

```
public sealed class SessionStoppedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← SessionStoppedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Session

Gets the ISession instance that has been stopped.

# Property Session

Namespace: DotNetBrowser.Capture.Events

Assembly: DotNetBrowser.dll

## Session

Gets the ISession instance that has been stopped.

```
public ISession Session { get; }
```

## Property Value

ISession

The ISession instance that has been stopped.

# Namespace DotNetBrowser.Capture.Handlers

## Classes

StartSessionParameters

The parameters of the StartSessionHandler.

StartSessionResponse

A response to the StartSessionHandler.

# Class StartSessionParameters

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

The parameters of the StartSessionHandler.

```
public sealed class StartSessionParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← StartSessionParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AudioMode

Gets the mode of capturing audio during the content capture session.

Frame

Gets the IFrame instance that requests the content capture.

Sources

Gets the available Sources instance.

Url

Gets the URL of the currently loaded web page.

# Property AudioMode

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

## AudioMode

Gets the mode of capturing audio during the content capture session.

```
public AudioMode AudioMode { get; }
```

## Property Value

AudioMode

The mode of capturing audio during the content capture session.

# Property Frame

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

## Frame

Gets the IFrame instance that requests the content capture.

```
public IFrame Frame { get; }
```

## Property Value

IFrame

The IFrame instance that requests the content capture.

# Property Sources

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

## Sources

Gets the available Sources instance.

```
public Sources Sources { get; }
```

## Property Value

Sources

the available Sources instance.

# Property Url

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the currently loaded web page.

```
public string Url { get; }
```

## Property Value

string

The URL of the currently loaded web page.

# Class StartSessionResponse

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

A response to the StartSessionHandler.

```
public sealed class StartSessionResponse
```

**Inheritance**

object ← StartSessionResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a StartSessionResponse that notifies the browser that the capture session request should be canceled.

SelectSource(IBrowser, AudioMode)

Creates a StartSessionResponse that notifies the browser to use the given browser as the capture source.

SelectSource(Source, AudioMode)

Creates a StartSessionResponse that notifies the browser to use the given capture source.

ShowSelectSourceDialog()

Creates a StartSessionResponse that notifies the browser to display the default dialog for choosing the capture source.

# Method Cancel

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a StartSessionResponse that notifies the browser that the capture session request should be canceled.

```
public static StartSessionResponse Cancel()
```

## Returns

StartSessionResponse

The StartSessionResponse instance that can be used as a return value in StartSessionHandler implementation.

# Method SelectSource

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

## SelectSource(Source, AudioMode)

Creates a StartSessionResponse that notifies the browser to use the given capture source.

```
public static StartSessionResponse SelectSource(Source source, AudioMode audioMode)
```

## Parameters

`source` Source

The content source to use for the capture.

`audioMode` AudioMode

The mode of capturing the audio from the browser.

## Returns

StartSessionResponse

The StartSessionResponse instance that can be used as a return value in StartSessionHandler implementation.

## Remarks

The

```
source
```

must be present in the Sources received from the Sources.

If the screen capture is forbidden by the system security permissions the request will be cancelled.This is usual for macOS where the screen or application window capture must be explicitly allowed in the System Preferences.

## Exceptions

ArgumentNullException

The `source` is null.

### SelectSource(IBrowser, AudioMode)

Creates a StartSessionResponse that notifies the browser to use the given browser as the capture source.

```
public static StartSessionResponse SelectSource(IBrowser browser, AudioMode audioMode)
```

## Parameters

`browser` IBrowser

The browser which contents will be captured.

`audioMode` AudioMode

The mode of capturing the audio from the browser.

## Returns

StartSessionResponse

The StartSessionResponse instance that can be used as a return value in StartSessionHandler implementation.

## Remarks

The passed browser instance and the browser instance requesting the capture session must be created by the same IEngine instance. Otherwise, the content capture will be cancelled.

## Exceptions

ArgumentNullException

The `browser` is null.

# Method ShowSelectSourceDialog

Namespace: DotNetBrowser.Capture.Handlers

Assembly: DotNetBrowser.dll

## ShowSelectSourceDialog()

Creates a StartSessionResponse that notifies the browser to display the default dialog for choosing the capture source.

```
public static StartSessionResponse ShowSelectSourceDialog()
```

## Returns

StartSessionResponse

The StartSessionResponse instance that can be used as a return value in StartSessionHandler implementation.

# Namespace DotNetBrowser.Card

## Classes

CreditCard

The credit card information persisted in ICreditCardStore the credit card store.

## Interfaces

ICreditCardStore

A service that allows working with CreditCard credit cards in the Chromium credit cards store.

ICreditCards

A service that allows managing credit cards.

## Enums

CreditCardNetworkType

The type of the credit card network.

# Class CreditCard

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

The credit card information persisted in ICreditCardStore the credit card store.

```
public sealed class CreditCard
```

**Inheritance**

object ← CreditCard

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Cardholder

Gets the name of the cardholder entered by the user.

ExpirationMonth

Gets the expiration month of the credit card.

ExpirationYear

Gets the expiration year of the credit card.

Network

Gets the payment network type of the card.

Number

Gets the card number.

## Methods

Equals(object)

GetHashCode()

# Property Cardholder

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## Cardholder

Gets the name of the cardholder entered by the user.

```
public string Cardholder { get; }
```

## Property Value

string

The cardholder.

# Property ExpirationMonth

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## ExpirationMonth

Gets the expiration month of the credit card.

```
public int ExpirationMonth { get; }
```

## Property Value

int

The expiration month.

## Remarks

The value may be between 1 and 12.

# Property ExpirationYear

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## ExpirationYear

Gets the expiration year of the credit card.

```
public int ExpirationYear { get; }
```

## Property Value

int

The expiration year.

# Property Network

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## Network

Gets the payment network type of the card.

```
public CreditCardNetworkType Network { get; }
```

## Property Value

CreditCardNetworkType

The payment network type of the card.

# Property Number

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## Number

Gets the card number.

```
public string Number { get; }
```

## Property Value

string

The card number.

# Method Equals

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Enum CreditCardNetworkType

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

The type of the credit card network.

```
public enum CreditCardNetworkType
```

## Fields

`AmericanExpress = 1`

The AmericanExpress network type.

`CreditCardNetworkTypeUnspecified = 0`

Unspecified Credit card network type.

`Diners = 2`

The Diners network type.

`Discover = 3`

The Discover network type.

`Elo = 4`

The Elo network type.

`Generic = 5`

The Generic network type.

`Jcb = 6`

The Jcb network type.

`MasterCard = 7`

The MasterCard network type.

`Mir = 8`

The Mir network type.

`Troy = 9`

The Troy network type.

`UnionPay = 10`

The UnionPay network type.

`Visa = 11`

The Visa network type.

# Interface ICreditCardStore

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

A service that allows working with CreditCard credit cards in the Chromium credit cards store.

```
public interface ICreditCardStore
```

## Properties

All
Gets all records from the credit cards store.

Profile
Gets the IProfile instance associated wit this ICreditCardStore.

## Methods

Clear()
Clears all records in the store.

Remove(CreditCard)
Removes the credit card from the store.

# Property All

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## All

Gets all records from the credit cards store.

```
IReadOnlyList<CreditCard> All { get; }
```

## Property Value

IReadOnlyList<CreditCard>

All records from the credit cards store.

## Remarks

Credit card records are saved to the store via SaveCreditCardHandler.

## Exceptions

ObjectDisposedException

The ICreditCardStore has already been disposed.

# Property Profile

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated wit this ICreditCardStore.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated wit this ICreditCardStore.

# Method Clear

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## Clear()

Clears all records in the store.

```
void Clear()
```

## Exceptions

ObjectDisposedException

The ICreditCardStore has already been disposed.

# Method Remove

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## Remove(CreditCard)

Removes the credit card from the store.

```
void Remove(CreditCard creditCard)
```

## Parameters

`creditCard` CreditCard

The credit card associated with the removed credit card records.

## Remarks

Removed credit cards are not displayed in the autofill suggestion pop-up.

## Exceptions

ObjectDisposedException

The ICreditCardStore has already been disposed.

ArgumentException

The `creditCard` is null.

# Interface ICreditCards

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

A service that allows managing credit cards.

```
public interface ICreditCards : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Browser
: Gets the IBrowser instance associated with this browser.

SaveCreditCardHandler
: Gets or sets a handler that is used when the user is prompted to save a credit card to the credit card store

# Property Browser

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with this browser.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with this ICreditCards instance.

# Property SaveCreditCardHandler

Namespace: DotNetBrowser.Card

Assembly: DotNetBrowser.dll

## SaveCreditCardHandler

Gets or sets a handler that is used when the user is prompted to save a credit card to the credit card store

```
IHandler<SaveCreditCardParameters, SaveCreditCardResponse> SaveCreditCardHandler { get;
set; }
```

## Property Value

IHandler<SaveCreditCardParameters, SaveCreditCardResponse>

## Remarks

The callback is invoked when the user submits a form with credit card information (a cardholder name, number, expiration date, CVV/CVC).

This callback is equivalent to the "Save Card?" bubble in Chromium.

Use the Save to save the credit card to the credit card store. All saved credit cards are shown in the suggestion pop-up when focusing the web form control.

Use the Decline to decline to save the credit card. If the current credit card is declined then the callback will be invoked again when submitting a web form with the same credit card.

The handler is not invoked if autofill is disabled.

## Exceptions

ObjectDisposedException

The ICreditCards has already been disposed.

# Namespace DotNetBrowser.Card.Handlers

## Classes

SaveCreditCardParameters

The parameters of the SaveCreditCardHandler.

SaveCreditCardResponse

A response to the SaveCreditCardHandler.

# Class SaveCreditCardParameters

Namespace: DotNetBrowser.Card.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SaveCreditCardHandler.

```
public sealed class SaveCreditCardParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← SaveCreditCardParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CreditCard

Gets the credit card that the user entered in the web form.

# Property CreditCard

Namespace: DotNetBrowser.Card.Handlers

Assembly: DotNetBrowser.dll

## CreditCard

Gets the credit card that the user entered in the web form.

```
public CreditCard CreditCard { get; }
```

## Property Value

CreditCard

The credit card that the user entered in the web form.

# Class SaveCreditCardResponse

Namespace: DotNetBrowser.Card.Handlers

Assembly: DotNetBrowser.dll

A response to the SaveCreditCardHandler.

```
public sealed class SaveCreditCardResponse
```

**Inheritance**

object ← SaveCreditCardResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Decline

Creates a SaveCreditCardResponse that notifies the browser to decline to save the credit card.

Save

Creates a SaveCreditCardResponse that notifies the browser to save the credit card to the ICreditCard Store credit card store.

# Field Decline

Namespace: DotNetBrowser.Card.Handlers

Assembly: DotNetBrowser.dll

Creates a SaveCreditCardResponse that notifies the browser to decline to save the credit card.

```
public static SaveCreditCardResponse Decline
```

## Returns

SaveCreditCardResponse

The SaveCreditCardResponse instance that can be used as a return value in SaveCreditCardHandler implementation.

# Field Save

Namespace: DotNetBrowser.Card.Handlers

Assembly: DotNetBrowser.dll

Creates a SaveCreditCardResponse that notifies the browser to save the credit card to the ICreditCard Store credit card store.

```
public static SaveCreditCardResponse Save
```

## Returns

SaveCreditCardResponse

The SaveCreditCardResponse instance that can be used as a return value in SaveCreditCardHandler implementation.

# Namespace DotNetBrowser.Cast

## Classes

CastSessionStartFailedException

Thrown when the cast session start has been failed.

MediaRoutingException

Thrown when the media routing is disabled.

PresentationRequest

The JavaScript

```
PresentationRequest
```

.

ReceiverDisconnectedException

Thrown when the receiver has been disconnected.

ReceiverNotDiscoveredException

Thrown when the receiver has not been discovered within the specified timeout.

Screen

The screen whose content can be cast.

ScreenCastOptions

Configuration options for screen casting.

## Interfaces

ICast

A service that provides access for casting media on receivers.

ICastSession

A session of casting media content to a media IMediaReceiver receiver.

The session is Discovered discovered when the user starts casting the browser/screen content or a presentation of media content via the DotNetBrowser API, or another application, i.e.Google Chrome. To indicate that the cast session has been started by another profile and, accordingly, a Chromium instance the IsLocal property is provided.

ICastSessions

A service that allows observing IsAlive cast ICastSession sessions.

IMediaCasting

A service that provides access to all the required media casting services.

IMediaReceiver

A media receiver to which media content can be cast.

Usually, a media receiver is a device that supports the ChromeCast technology, but in Chromium's logic it can be even a wired display (HDMI, DVI, or similar).

IMediaReceivers

The service that allows observing media IMediaReceiver receivers in the environment.

IScreens

The service that allows obtaining connected screens whose content can be cast.

## Enums

AudioMode

The audio casting mode for the content cast session.

CastMode

The types of content that can be cast to a media receiver.

MediaReceiverState

The state of the media receiver.

ResultCode

Contains the codes indicating the result of creating a cast session.

# Enum AudioMode

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

The audio casting mode for the content cast session.

```
public enum AudioMode
```

## Fields

`Cast = 1`

The audio content should be cast along with the video content.

`Ignore = 0`

The audio content should not be cast along with the video content.

# Enum CastMode

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

The types of content that can be cast to a media receiver.

```
public enum CastMode
```

## Fields

`Browser = 2`

Cast the browser's web content.

`Presentation = 1`

Cast the content specified by the presentation URLs or frame's origin provided via the Presentation API.

`Screen = 3`

Cast the screen's content.

`Unspecified = 0`

Unspecified mode.

# Class CastSessionStartFailedException

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Thrown when the cast session start has been failed.

```
public sealed class CastSessionStartFailedException : Exception, ISerializable
```

**Inheritance**

object ← Exception ← CastSessionStartFailedException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Constructors

CastSessionStartFailedException(string, ResultCode)

## Properties

ResultCode

Gets the error code obtained from Chromium.

# Constructor CastSessionStartFailedException

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## CastSessionStartFailedException(string, ResultCode)

```
public CastSessionStartFailedException(string message, ResultCode resultCode)
```

## Parameters

`message` string

`resultCode` ResultCode

# Property ResultCode

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## ResultCode

Gets the error code obtained from Chromium.

```
public ResultCode ResultCode { get; }
```

## Property Value

ResultCode

The error code obtained from Chromium.

# Interface ICast

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

A service that provides access for casting media on receivers.

```
public interface ICast
```

## Properties

StartPresentationHandler

Gets or sets a handler that is used when a presentation has been requested via the JavaScript Presentation API.

## Methods

CastContent(IMediaReceiver)

Starts casting the browser content to IMediaReceiver.

CastScreen(IMediaReceiver)

Starts casting screen's content by selecting the screen in the picker dialog.

CastScreen(IMediaReceiver, ScreenCastOptions)

Starts casting the screen's content to the media receiver.

DefaultPresentationRequest()

The default JavaScript PresentationRequest specified on the page.

Usually, the default PresentationRequest is specified on resources like YouTube. The developer can specify it this way:

```
const presentationRequest = new PresentationRequest(['receiver/index.html']);
    navigator.presentation.defaultRequest = presentationRequest;
```

# Property StartPresentationHandler

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## StartPresentationHandler

Gets or sets a handler that is used when a presentation has been requested via the JavaScript Presentation API.

```
IHandler<StartPresentationParameters, StartPresentationResponse> StartPresentationHandler {
get; set; }
```

## Property Value

IHandler<StartPresentationParameters, StartPresentationResponse>

## Remarks

Use the Start(IMediaReceiver) method to use the given IMediaReceiver receiver.

Use theCancel() method if the cast session request should be canceled.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel() will be used.

Presentation API is available only in secure contexts (HTTPS).

The callback can be invoked only if media routing is IsMediaRoutingEnabled enabled.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Method CastContent

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## CastContent(IMediaReceiver)

Starts casting the browser content to IMediaReceiver.

```
Task<ICastSession> CastContent(IMediaReceiver receiver)
```

## Parameters

`receiver` IMediaReceiver

media receiver for casting.

## Returns

Task<ICastSession>

The task that represents the result of the asynchronous start of the cast session. If the cast session has not been started, the task is completed with CastSessionStartFailedException. If the browser is closed during the casting start,the task will be canceled.

## Remarks

In case when the web page has the default JavaScript PresentationRequest:

```
const presentationRequest = new PresentationRequest(['receiver/index.html']);
    navigator.presentation.defaultRequest = presentationRequest;
```

The presentation of the media content specified in this request is started instead of casting the browser content. This implies that the receiver should Supports(CastMode, PresentationRequest) support the presentation as a media source. Otherwise, if there is no default JavaScript PresentationRequest, the

receiver should Supports(CastMode, PresentationRequest) support any browser's content as Browser media source.

## Exceptions

MediaRoutingException

The media routing switch was not set for IEngine.

# Method CastScreen

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## CastScreen(IMediaReceiver)

Starts casting screen's content by selecting the screen in the picker dialog.

```
Task<ICastSession> CastScreen(IMediaReceiver receiver)
```

### Parameters

`receiver` IMediaReceiver

media receiver for casting.

### Returns

Task<ICastSession>

The task that represents the result of the asynchronous start of the cast session. If the cast session has not been started, the task is completed with CastSessionStartFailedException. If the browser is closed during the casting start,the task will be canceled.

### Remarks

The receiver should support the Screen cast mode.

### Exceptions

MediaRoutingException

The media routing switch was not set for IEngine.

## CastScreen(IMediaReceiver, ScreenCastOptions)

Starts casting the screen's content to the media receiver.

```
Task<ICastSession> CastScreen(IMediaReceiver receiver, ScreenCastOptions options)
```

### Parameters

`receiver` IMediaReceiver

media receiver for casting.

`options` ScreenCastOptions

screen cast options for casting.

### Returns

Task<ICastSession>

The task that represents the result of the asynchronous start of the cast session. If the cast session has not been started, the task is completed with CastSessionStartFailedException. If the browser is closed during the casting start,the task will be canceled.

### Remarks

The receiver should support the Screen cast mode.

### Exceptions

MediaRoutingException

The media routing switch was not set for IEngine.

# Method DefaultPresentationRequest

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## DefaultPresentationRequest()

The default JavaScript PresentationRequest specified on the page.

Usually, the default PresentationRequest is specified on resources like YouTube. The developer can specify it this way:

```
const presentationRequest = new PresentationRequest(['receiver/index.html']);
    navigator.presentation.defaultRequest = presentationRequest;
```

```
PresentationRequest DefaultPresentationRequest()
```

## Returns

PresentationRequest

The instance of the default JavaScript PresentationRequest.

# Interface ICastSession

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

A session of casting media content to a media IMediaReceiver receiver.

The session is Discovered discovered when the user starts casting the browser/screen content or a presentation of media content via the DotNetBrowser API, or another application, i.e.Google Chrome. To indicate that the cast session has been started by another profile and, accordingly, a Chromium instance the IsLocal property is provided.

```
public interface ICastSession
```

## Properties

Description
: Gets the description of the session. Examples: "Mirroring tab (www.example.com)", "Casting media", "Casting YouTube".

IsAlive
: Indicates whether this cast session is alive.

IsLocal
: Indicates whether this cast session is initiated and managed by the current IProfile.

MediaReceiver
: Gets the media receiver of this session.

Mode
: Gets the mode of the session.

## Methods

Stop()
: Stops the cast session.

# Events

Stopped

Occurs when a ICastSession session has been stopped.

Also, the cast session is stopped when the user starts a new cast session to the MediaReceiver receiver of this cast session.

# Property Description

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Description

Gets the description of the session. Examples: "Mirroring tab (www.example.com)", "Casting media", "Casting YouTube".

```
string Description { get; }
```

## Property Value

string

The description of the session.

## Remarks

The description may be empty when the session just started and Chromium didn't define the cast content yet.

# Property IsAlive

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## IsAlive

Indicates whether this cast session is alive.

```
bool IsAlive { get; }
```

## Property Value

bool

`true` if the session is alive, `false` otherwise.

# Property IsLocal

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## IsLocal

Indicates whether this cast session is initiated and managed by the current IProfile.

```
bool IsLocal { get; }
```

## Property Value

bool

`true` if the session is local, `false` otherwise.

# Property MediaReceiver

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## MediaReceiver

Gets the media receiver of this session.

```
IMediaReceiver MediaReceiver { get; }
```

## Property Value

IMediaReceiver

The media receiver of this session.

## Remarks

The method may block the current thread if the receiver is not discovered yet. It may happen when the cast session is not IsLocal local and started on the receiver not yet discovered by the current instance of Chromium.

# Property Mode

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Mode

Gets the mode of the session.

```
CastMode Mode { get; }
```

## Property Value

CastMode

The mode of the session.

# Method Stop

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Stop()

Stops the cast session.

```
void Stop()
```

## Remarks

This method has no effect when this cast session is already stopped.

# Event Stopped

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Occurs when a ICastSession session has been stopped.

Also, the cast session is stopped when the user starts a new cast session to the MediaReceiver receiver of this cast session.

```
event EventHandler<CastSessionStoppedEventArgs> Stopped
```

## Returns

EventHandler<CastSessionStoppedEventArgs>

Occurs when a session has been stopped. Also, the cast session is stopped when the user starts a new cast session to the receiver of this cast session.

# Interface ICastSessions

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

A service that allows observing IsAlive cast ICastSession sessions.

```
public interface ICastSessions
```

## Properties

AllAlive

Gets the list of IsAlive alive cast sessions.

## Events

Discovered

Occurs when a new ICastSession session has been discovered in the environment.

StartFailed

Occurs when a start of a ICastSession session has been failed.

# Property AllAlive

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## AllAlive

Gets the list of IsAlive alive cast sessions.

```
IReadOnlyList<ICastSession> AllAlive { get; }
```

## Property Value

IReadOnlyList<ICastSession>

The list of IsAlive alive cast sessions.

# Event Discovered

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Occurs when a new ICastSession session has been discovered in the environment.

```
event EventHandler<CastSessionDiscoveredEventArgs> Discovered
```

## Returns

EventHandler<CastSessionDiscoveredEventArgs>

Occurs when a new session has been discovered in the environment.

# Event StartFailed

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Occurs when a start of a ICastSession session has been failed.

```
event EventHandler<CastSessionStartFailedEventArgs> StartFailed
```

## Returns

EventHandler<CastSessionStartFailedEventArgs>

Occurs when a start of a session has been failed.

# Interface IMediaCasting

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

A service that provides access to all the required media casting services.

```
public interface IMediaCasting
```

## Properties

CastSessions

Gets the service that allows observing alive cast sessions.

Receivers

Gets the service that allows observing media receivers.

Screens

Gets the service that allows obtaining connected screens for casting.

# Property CastSessions

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## CastSessions

Gets the service that allows observing alive cast sessions.

```
ICastSessions CastSessions { get; }
```

## Property Value

ICastSessions

The service that allows observing alive cast sessions.

# Property Receivers

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Receivers

Gets the service that allows observing media receivers.

```
IMediaReceivers Receivers { get; }
```

## Property Value

IMediaReceivers

The service that allows observing media receivers.

# Property Screens

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Screens

Gets the service that allows obtaining connected screens for casting.

```
IScreens Screens { get; }
```

## Property Value

IScreens

The service that allows obtaining connected screens for casting.

# Interface IMediaReceiver

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

A media receiver to which media content can be cast.

Usually, a media receiver is a device that supports the ChromeCast technology, but in Chromium's logic it can be even a wired display (HDMI, DVI, or similar).

```
public interface IMediaReceiver
```

## Properties

Name
: Gets the name of the receiver.

State
: Gets the current state of the receiver.

## Methods

Supports(CastMode, PresentationRequest)
: Checks if the receiver supports casting of CastMode.

## Events

Disconnected
: Occurs when the receiver disconnected and is not observable by Chromium anymore.

# Property Name

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Name

Gets the name of the receiver.

```
string Name { get; }
```

## Property Value

string

Returns the name of the receiver.

# Property State

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## State

Gets the current state of the receiver.

```
MediaReceiverState State { get; }
```

## Property Value

MediaReceiverState

Returns the current state of the receiver.

# Method Supports

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Supports(CastMode, PresentationRequest)

Checks if the receiver supports casting of CastMode.

```
bool Supports(CastMode castMode, PresentationRequest presentationRequest = null)
```

## Parameters

`castMode` CastMode

The parameter of the types of content that can be cast to a media receiver.

`presentationRequest` PresentationRequest

The parameter of the JavaScript PresentationRequest.

## Returns

bool

`true` if the receiver supports casting of CastMode, `false` otherwise.

## Remarks

The PresentationRequest parameter should be set only for the Presentation mode.

# Event Disconnected

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Occurs when the receiver disconnected and is not observable by Chromium anymore.

```
event EventHandler<MediaReceiverDisconnectedEventArgs> Disconnected
```

## Returns

EventHandler<MediaReceiverDisconnectedEventArgs>

Occurs when the receiver disconnected and is not observable by Chromium anymore.

# Interface IMediaReceivers

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

The service that allows observing media IMediaReceiver receivers in the environment.

```
public interface IMediaReceivers
```

## Properties

AllAvailable

Gets the list of connected (not Unavailable unavailable) media receivers.

The initial list of media receivers can be empty or not full. This is because receivers are MediaReceiver DiscoveredEventArgs discovered asynchronously in Chromium.

## Methods

Refresh()

Asynchronously updates the list of available media receivers.

This method forces Chromium to discover media receivers available in the local network. Use it before calling the RetrieveAsync(Predicate<IMediaReceiver>, TimeSpan) method to shorten the time to discover new receivers.

RetrieveAsync(Predicate<IMediaReceiver>)

Waiting till the first receiver matching the `predicate` is discovered.

RetrieveAsync(Predicate<IMediaReceiver>, TimeSpan)

Waiting till the first receiver matching the `predicate` is discovered.

## Events

Discovered

Occurs when a new media IMediaReceiver receiver has been discovered in the environment.

# Property AllAvailable

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## AllAvailable

Gets the list of connected (not Unavailable unavailable) media receivers.

The initial list of media receivers can be empty or not full. This is because receivers are MediaReceiver DiscoveredEventArgs discovered asynchronously in Chromium.

```
IReadOnlyList<IMediaReceiver> AllAvailable { get; }
```

## Property Value

IReadOnlyList<IMediaReceiver>

The list of connected (not Unavailable unavailable) media receivers.

# Method Refresh

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Refresh()

Asynchronously updates the list of available media receivers.

This method forces Chromium to discover media receivers available in the local network. Use it before calling the RetrieveAsync(Predicate<IMediaReceiver>, TimeSpan) method to shorten the time to discover new receivers.

```
void Refresh()
```

# Method RetrieveAsync

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## RetrieveAsync(Predicate<IMediaReceiver>, TimeSpan)

Waiting till the first receiver matching the `predicate` is discovered.

```
Task<IMediaReceiver> RetrieveAsync(Predicate<IMediaReceiver> predicate, TimeSpan timeout)
```

## Parameters

`predicate` Predicate<IMediaReceiver>

The predicated media receiver.

`timeout` TimeSpan

The predefined timeout.

## Returns

Task<IMediaReceiver>

The task that can be used to wait for completion and obtain the result with the first discovered receiver matching the `predicate`.

## Remarks

If a matching receiver has already been discovered, returns it immediately.

Example of usage:

```
browser.Cast.StartPresentationHandler =
    new AsyncHandler<StartPresentationParameters, StartPresentationResponse>(async p =>
    {
        IMediaReceiver mediaReceiver
```

```
            = await p.MediaReceivers
                    .RetrieveAsync(receiver => receiver.Name.Contains("Samsung
TV"), timeout);
        return StartPresentationResponse.Start(mediaReceiver);
    });
```

## Exceptions

ReceiverNotDiscoveredException

When the receiver has not been discovered within `timeout`.

### RetrieveAsync(Predicate<IMediaReceiver>)

Waiting till the first receiver matching the `predicate` is discovered.

```
Task<IMediaReceiver> RetrieveAsync(Predicate<IMediaReceiver> predicate)
```

## Parameters

`predicate` Predicate<IMediaReceiver>

The predicated media receiver.

## Returns

Task<IMediaReceiver>

The task that can be used to wait for completion and obtain the result with the first discovered receiver matching the `predicate`.

## Remarks

If a matching receiver has already been discovered, returns it immediately.

Example of usage:

```
browser.Cast.StartPresentationHandler =
    new AsyncHandler<StartPresentationParameters, StartPresentationResponse>(async p =>
    {
        IMediaReceiver mediaReceiver
            = await p.MediaReceivers
                     .RetrieveAsync(receiver => receiver.Name.Contains("Samsung TV"));
        return StartPresentationResponse.Start(mediaReceiver);
    });
```

## Exceptions

ReceiverNotDiscoveredException

When the receiver has not been discovered within 45 seconds.

# Event Discovered

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Occurs when a new media IMediaReceiver receiver has been discovered in the environment.

```
event EventHandler<MediaReceiverDiscoveredEventArgs> Discovered
```

## Returns

EventHandler<MediaReceiverDiscoveredEventArgs>

Occurs when a new media receiver has been discovered in the environment.

# Interface IScreens

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

The service that allows obtaining connected screens whose content can be cast.

```
public interface IScreens
```

## Properties

All

Gets the list of connected screens whose content can be cast to a media receiver.

DefaultScreen

Gets the default (main) screen.

# Property All

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## All

Gets the list of connected screens whose content can be cast to a media receiver.

```
IReadOnlyList<Screen> All { get; }
```

## Property Value

IReadOnlyList<Screen>

The list of connected screens whose content can be cast to a media receiver.

# Property DefaultScreen

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## DefaultScreen

Gets the default (main) screen.

```
Screen DefaultScreen { get; }
```

## Property Value

Screen

The default (main) screen.

# Enum MediaReceiverState

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

The state of the media receiver.

```
public enum MediaReceiverState
```

## Fields

`Available = 1`

The receiver is available to be cast to.

`Connected = 3`

Content is currently cast to the receiver.

`Connecting = 2`

The receiver is starting a new cast session. A receiver temporarily enters this state when transitioning from Available to Connected.

`Disconnecting = 4`

The receiver is still connected but is in the process of disconnecting. A receiver temporarily enters this state when transitioning from Connected to Available.

`Unavailable = 5`

The receiver is disconnected/cached (not available right now).

`Undefined = 0`

The undefined state of the media receiver.

# Class MediaRoutingException

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Thrown when the media routing is disabled.

```
public sealed class MediaRoutingException : Exception, ISerializable
```

**Inheritance**

object ← Exception ← MediaRoutingException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Remarks

Media routing can be enabled by setting `true` in the MediaRoutingEnabled property.

Should be set before the IEngine instance is created.

## Constructors

MediaRoutingException()

# Constructor MediaRoutingException

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## MediaRoutingException()

```
public MediaRoutingException()
```

# Class PresentationRequest

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

The JavaScript

```
PresentationRequest
```

.

```
public sealed class PresentationRequest : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← PresentationRequest

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Frame

Gets the frame from which the presentation has been requested.

OriginUrl

Gets the origin URL of the IFrame where the presentation has been requested.

PresentationUrls

Gets all of URLs whose content will be cast.

# Property Frame

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Frame

Gets the frame from which the presentation has been requested.

```
public IFrame Frame { get; }
```

## Property Value

IFrame

The frame from which the presentation has been requested.

# Property OriginUrl

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## OriginUrl

Gets the origin URL of the IFrame where the presentation has been requested.

```
public string OriginUrl { get; }
```

## Property Value

string

The origin URL of the IFrame where the presentation has been requested.

# Property PresentationUrls

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## PresentationUrls

Gets all of URLs whose content will be cast.

```
public IReadOnlyList<string> PresentationUrls { get; }
```

## Property Value

IReadOnlyList<string>

All presentation URLs whose content will be cast.

# Class ReceiverDisconnectedException

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Thrown when the receiver has been disconnected.

```
public sealed class ReceiverDisconnectedException : Exception, ISerializable
```

### Inheritance

object ← Exception ← ReceiverDisconnectedException

### Implements

ISerializable

### Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Constructors

ReceiverDisconnectedException(string)

# Constructor ReceiverDisconnectedException

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## ReceiverDisconnectedException(string)

```
public ReceiverDisconnectedException(string message)
```

## Parameters

`message` string

# Class ReceiverNotDiscoveredException

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Thrown when the receiver has not been discovered within the specified timeout.

```
public sealed class ReceiverNotDiscoveredException : Exception, ISerializable
```

### Inheritance

object ← Exception ← ReceiverNotDiscoveredException

### Implements

ISerializable

### Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Constructors

ReceiverNotDiscoveredException()

# Constructor ReceiverNotDiscoveredException

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## ReceiverNotDiscoveredException()

```
public ReceiverNotDiscoveredException()
```

# Enum ResultCode

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Contains the codes indicating the result of creating a cast session.

```
public enum ResultCode
```

## Fields

`Cancelled = 6`

The cast session has been canceled, i.e. via the "Cancel" button in the screen picker.

`CastSessionAlreadyExists = 7`

The cast session can not be created since it already exists.

`CastSessionNotFound = 3`

The cast session was not found.

`DesktopPickerFailed = 8`

An attempt to show the screen picker has failed.

`NoSupportedProvider = 5`

The media source provider is not supported.

`Ok = 1`

The cast session successfully started.

`ReceiverNotFound = 4`

The receiver was not found when starting the cast session.

`ResultCodeUnspecified = 0`

Reserved value.

`TimedOut = 2`

The cast session has not been created within the specified time.

# Class Screen

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

The screen whose content can be cast.

```
public sealed class Screen
```

**Inheritance**

object ← Screen

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Name

Gets the screen whose content can be cast.

## Methods

Equals(object)

GetHashCode()

# Property Name

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Name

Gets the screen whose content can be cast.

```
public string Name { get; }
```

## Property Value

string

Returns the name of the screen assigned by Chromium, for example: "Entire screen", "Screen 1".

# Method Equals

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class ScreenCastOptions

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

Configuration options for screen casting.

```
public sealed class ScreenCastOptions
```

**Inheritance**

object ← ScreenCastOptions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AudioMode

Gets or sets the mode of casting audio during the content cast session.

Screen

Gets or sets the screen to cast.

# Property AudioMode

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## AudioMode

Gets or sets the mode of casting audio during the content cast session.

```
public AudioMode AudioMode { get; set; }
```

## Property Value

AudioMode

The mode of casting audio during the content cast session.

# Property Screen

Namespace: DotNetBrowser.Cast

Assembly: DotNetBrowser.dll

## Screen

Gets or sets the screen to cast.

```
public Screen Screen { get; set; }
```

## Property Value

Screen

The screen whose content will be cast.

# Namespace DotNetBrowser.Cast.Events

## Classes

CastSessionDiscoveredEventArgs

Event arguments for the Discovered event.

The new cast session is discovered after the user started casting media content via DotNetBrowser API, or another application, i.e.Google Chrome.

CastSessionStartFailedEventArgs

Event arguments for the StartFailed event.

CastSessionStoppedEventArgs

Event arguments for the Stopped event.

Also, the cast session is stopped when the user starts a new cast session to the MediaReceiver receiver of this cast session.

MediaReceiverDisconnectedEventArgs

Event arguments for the Disconnected event.

MediaReceiverDiscoveredEventArgs

Event arguments for the Discovered event.

# Class CastSessionDiscoveredEventArgs

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

Event arguments for the Discovered event.

The new cast session is discovered after the user started casting media content via DotNetBrowser API, or another application, i.e.Google Chrome.

```
public sealed class CastSessionDiscoveredEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← CastSessionDiscoveredEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CastSession

Gets the discovered session.

CastSessions

Gets the ICastSessions instance initiated this event.

# Property CastSession

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## CastSession

Gets the discovered session.

```
public ICastSession CastSession { get; }
```

## Property Value

ICastSession

The discovered session.

# Property CastSessions

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## CastSessions

Gets the ICastSessions instance initiated this event.

```
public ICastSessions CastSessions { get; }
```

## Property Value

ICastSessions

The ICastSessions instance initiated this event.

# Class CastSessionStartFailedEventArgs

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

Event arguments for the StartFailed event.

```
public sealed class CastSessionStartFailedEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← CastSessionStartFailedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CastSessions

Gets the ICastSessions instance initiated this event.

Code

Gets the error code obtained from Chromium.

ErrorMessage

Gets the error message obtained from Chromium.

MediaReceiver

Gets the media receiver of the failed cast session.

Mode

Gets the mode of the requested cast session.

# Property CastSessions

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## CastSessions

Gets the ICastSessions instance initiated this event.

```
public ICastSessions CastSessions { get; }
```

## Property Value

ICastSessions

The ICastSessions instance initiated this event.

# Property Code

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## Code

Gets the error code obtained from Chromium.

```
public ResultCode Code { get; }
```

## Property Value

ResultCode

The error code obtained from Chromium.

# Property ErrorMessage

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## ErrorMessage

Gets the error message obtained from Chromium.

```
public string ErrorMessage { get; }
```

## Property Value

string

The error message obtained from Chromium.

# Property MediaReceiver

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## MediaReceiver

Gets the media receiver of the failed cast session.

```
public IMediaReceiver MediaReceiver { get; }
```

## Property Value

IMediaReceiver

The media receiver of the failed cast session.

# Property Mode

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## Mode

Gets the mode of the requested cast session.

```
public CastMode Mode { get; }
```

## Property Value

CastMode

The mode of the requested cast session.

# Class CastSessionStoppedEventArgs

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

Event arguments for the Stopped event.

Also, the cast session is stopped when the user starts a new cast session to the MediaReceiver receiver of this cast session.

```
public sealed class CastSessionStoppedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← CastSessionStoppedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CastSession

Gets the cast session that has been stopped.

# Property CastSession

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## CastSession

Gets the cast session that has been stopped.

```
public ICastSession CastSession { get; }
```

## Property Value

ICastSession

The cast session that has been stopped.

# Class MediaReceiverDisconnectedEventArgs

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

Event arguments for the Disconnected event.

```
public sealed class MediaReceiverDisconnectedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← MediaReceiverDisconnectedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

MediaReceiver

Gets the media receiver that has been disconnected.

# Property MediaReceiver

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## MediaReceiver

Gets the media receiver that has been disconnected.

```
public IMediaReceiver MediaReceiver { get; }
```

## Property Value

IMediaReceiver

The media receiver that has been disconnected.

# Class MediaReceiverDiscoveredEventArgs

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

Event arguments for the Discovered event.

```
public sealed class MediaReceiverDiscoveredEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← MediaReceiverDiscoveredEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

MediaReceiver

Gets the discovered receiver.

MediaReceivers

Gets the IMediaReceivers instance initiated this event.

# Property MediaReceiver

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## MediaReceiver

Gets the discovered receiver.

```
public IMediaReceiver MediaReceiver { get; }
```

## Property Value

IMediaReceiver

The discovered receiver.

# Property MediaReceivers

Namespace: DotNetBrowser.Cast.Events

Assembly: DotNetBrowser.dll

## MediaReceivers

Gets the IMediaReceivers instance initiated this event.

```
public IMediaReceivers MediaReceivers { get; }
```

## Property Value

IMediaReceivers

The IMediaReceivers instance initiated this event.

# Namespace DotNetBrowser.Cast.Handlers

## Classes

StartPresentationParameters

The parameters of the StartPresentationHandler.

StartPresentationResponse

A response to the StartPresentationHandler.

# Class StartPresentationParameters

Namespace: DotNetBrowser.Cast.Handlers

Assembly: DotNetBrowser.dll

The parameters of the StartPresentationHandler.

```
public sealed class StartPresentationParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← StartPresentationParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AvailableReceivers

Gets the service that allows observing available media receivers.

PresentationRequest

Gets the JavaScript

```
PresentationRequest
```

initiated this presentation.

SupportedReceivers

Gets the list of media receivers that are able to start presentation from PresentationRequest.

# Property AvailableReceivers

Namespace: DotNetBrowser.Cast.Handlers

Assembly: DotNetBrowser.dll

## AvailableReceivers

Gets the service that allows observing available media receivers.

```
public IMediaReceivers AvailableReceivers { get; }
```

## Property Value

IMediaReceivers

The service that allows observing available media receivers.

# Property PresentationRequest

Namespace: DotNetBrowser.Cast.Handlers

Assembly: DotNetBrowser.dll

## PresentationRequest

Gets the JavaScript

```
PresentationRequest
```

initiated this presentation.

```
public PresentationRequest PresentationRequest { get; }
```

## Property Value

PresentationRequest

The JavaScript

```
PresentationRequest
```

initiated this presentation.

# Property SupportedReceivers

Namespace: DotNetBrowser.Cast.Handlers

Assembly: DotNetBrowser.dll

## SupportedReceivers

Gets the list of media receivers that are able to start presentation from PresentationRequest.

```
public IReadOnlyList<IMediaReceiver> SupportedReceivers { get; }
```

## Property Value

IReadOnlyList<IMediaReceiver>

The list of media receivers that are able to start presentation from PresentationRequest.

# Class StartPresentationResponse

Namespace: DotNetBrowser.Cast.Handlers

Assembly: DotNetBrowser.dll

A response to the StartPresentationHandler.

```
public sealed class StartPresentationResponse
```

**Inheritance**

object ← StartPresentationResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a StartPresentationResponse that notifies the browser that the presentation should be canceled.

Start(IMediaReceiver)

Creates a StartPresentationResponse that notifies the browser that the presentation should be started on the IMediaReceiver receiver.

# Method Cancel

Namespace: DotNetBrowser.Cast.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a StartPresentationResponse that notifies the browser that the presentation should be canceled.

```
public static StartPresentationResponse Cancel()
```

## Returns

StartPresentationResponse

The StartPresentationResponse instance that can be used as a return value in StartPresentationHandler implementation.

# Method Start

Namespace: DotNetBrowser.Cast.Handlers

Assembly: DotNetBrowser.dll

## Start(IMediaReceiver)

Creates a StartPresentationResponse that notifies the browser that the presentation should be started on the IMediaReceiver receiver.

```
public static StartPresentationResponse Start(IMediaReceiver mediaReceiver)
```

### Parameters

`mediaReceiver` IMediaReceiver

### Returns

StartPresentationResponse

The StartPresentationResponse instance that can be used as a return value in StartPresentationHandler implementation.

# Namespace DotNetBrowser.Chromium

## Classes

ChromiumBinariesExtractor

A tool that is used to extract the corresponding Chromium binaries.

ChromiumInfo

Provides information about the Chromium engine used in DotNetBrowser.

# Class ChromiumBinariesExtractor

Namespace: DotNetBrowser.Chromium

Assembly: DotNetBrowser.Core.dll

A tool that is used to extract the corresponding Chromium binaries.

```
public sealed class ChromiumBinariesExtractor
```

**Inheritance**

object ← ChromiumBinariesExtractor

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

ChromiumBinariesExtractor(BinariesExtractionOptions)

Creates and initializes an instance of ChromiumBinariesExtractor.

## Properties

DefaultChromiumDirectory

The default directory that will be used for binaries extraction if a different location is not specified in the ChromiumDirectory.

## Methods

ExtractBinariesIfNecessary(EngineOptions)

Locate and extract the Chromium binaries if the corresponding binaries were not extracted yet or cannot be found at the location specified in ChromiumDirectory.

If the ChromiumDirectory is not specified, the DefaultChromiumDirectory value will be used for binaries extraction.

# Constructor ChromiumBinariesExtractor

Namespace: DotNetBrowser.Chromium

Assembly: DotNetBrowser.Core.dll

## ChromiumBinariesExtractor(BinariesExtractionOptions)

Creates and initializes an instance of ChromiumBinariesExtractor.

```
public ChromiumBinariesExtractor(BinariesExtractionOptions extractionOptions = null)
```

## Parameters

`extractionOptions` BinariesExtractionOptions

the extraction options that are used to configure the binaries extractor.

# Property DefaultChromiumDirectory

Namespace: DotNetBrowser.Chromium

Assembly: DotNetBrowser.Core.dll

## DefaultChromiumDirectory

The default directory that will be used for binaries extraction if a different location is not specified in the ChromiumDirectory.

```
public string DefaultChromiumDirectory { get; }
```

## Property Value

string

# Method ExtractBinariesIfNecessary

Namespace: DotNetBrowser.Chromium

Assembly: DotNetBrowser.Core.dll

## ExtractBinariesIfNecessary(EngineOptions)

Locate and extract the Chromium binaries if the corresponding binaries were not extracted yet or cannot be found at the location specified in ChromiumDirectory.

If the ChromiumDirectory is not specified, the DefaultChromiumDirectory value will be used for binaries extraction.

```
public void ExtractBinariesIfNecessary(EngineOptions options)
```

## Parameters

`options` EngineOptions

the EngineOptions instance to use.

# Class ChromiumInfo

Namespace: DotNetBrowser.Chromium

Assembly: DotNetBrowser.Core.dll

Provides information about the Chromium engine used in DotNetBrowser.

```
public static class ChromiumInfo
```

**Inheritance**

object ← ChromiumInfo

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Version

The current version of the underlying Chromium engine.

# Field Version

Namespace: DotNetBrowser.Chromium

Assembly: DotNetBrowser.Core.dll

The current version of the underlying Chromium engine.

```
public static readonly Version Version
```

## Returns

Version

The current version of the underlying Chromium engine.

# Namespace DotNetBrowser.ContextMenu

## Classes

ContextMenuItem
: A custom context menu item.

## Enums

ContextMenuItemType
: The context menu item types.

# Class ContextMenuItem

Namespace: DotNetBrowser.ContextMenu

Assembly: DotNetBrowser.dll

A custom context menu item.

```
public sealed class ContextMenuItem
```

**Inheritance**

object ← ContextMenuItem

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

IsChecked

Indicates whether context menu item is checked or not.

IsEnabled

Indicates whether context menu item is enabled or not.

SubItems

Gets a collection of sub-items.

Text

Gets the text of the context menu item.

Type

Gets the type of the context menu item.

# Property IsChecked

Namespace: DotNetBrowser.ContextMenu

Assembly: DotNetBrowser.dll

## IsChecked

Indicates whether context menu item is checked or not.

```
public bool IsChecked { get; }
```

## Property Value

bool

`true` if the item is checked, `false` otherwise.

# Property IsEnabled

Namespace: DotNetBrowser.ContextMenu

Assembly: DotNetBrowser.dll

## IsEnabled

Indicates whether context menu item is enabled or not.

```
public bool IsEnabled { get; }
```

## Property Value

bool

`true` if the item is enabled, `false` otherwise.

# Property SubItems

Namespace: DotNetBrowser.ContextMenu

Assembly: DotNetBrowser.dll

## SubItems

Gets a collection of sub-items.

```
public IEnumerable<ContextMenuItem> SubItems { get; }
```

## Property Value

IEnumerable<ContextMenuItem>

The collection of sub-items or an empty collection if the menu item doesn't have sub-items.

# Property Text

Namespace: DotNetBrowser.ContextMenu

Assembly: DotNetBrowser.dll

## Text

Gets the text of the context menu item.

```
public string Text { get; }
```

## Property Value

string

The text of the context menu item. If this menu item represents a separator, its text will be an empty string.

# Property Type

Namespace: DotNetBrowser.ContextMenu

Assembly: DotNetBrowser.dll

## Type

Gets the type of the context menu item.

```
public ContextMenuItemType Type { get; }
```

## Property Value

ContextMenuItemType

The type of the context menu item.

# Enum ContextMenuItemType

Namespace: DotNetBrowser.ContextMenu

Assembly: DotNetBrowser.dll

The context menu item types.

```
public enum ContextMenuItemType
```

## Fields

`CheckableItem = 2`

The context menu item is a checkable item.

`Group = 3`

The context menu item is a sub-menu with items.

`Item = 1`

The context menu item is a simple item.

`Separator = 4`

The context menu item is a separator.

`SubMenu = 5`

The context menu item is a sub-menu with items.

`Unspecified = 0`

The context menu item type is not specified.

# Namespace DotNetBrowser.Cookies

## Classes

Cookie

Represents an HTTP cookie.

Cookie.Builder

A builder class to construct cookie.

## Interfaces

ICookieStore

The system for storing and retrieving cookies. The cookies can be stored in the process memory (session cookies) or in files (persistent cookies). The ICookieStore instance provides access to both session and persistent cookies.

## Enums

SameSite

The SameSite cookie attribute values of the `Set-Cookie` HTTP response header. This attribute is used to declare in which context the cookies can be sent.

# Class Cookie

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

Represents an HTTP cookie.

```
public sealed class Cookie
```

**Inheritance**

object ← Cookie

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object)

## Properties

CreationTime

Gets the cookie creation time.

DomainName

Gets the domain name set for this cookie. It specifies allowed hosts to receive the cookie.

ExpirationTime

Gets the cookie expiration time.

IsHttpOnly

Gets the `HttpOnly` attribute value of the cookie. The `HttpOnly` cookie attribute can help to mitigate hijacking and XSS attacks by preventing access to cookie value through JavaScript.

IsSecure

Gets the `Secure` attribute value of the cookie. A secure cookie is only sent to the server with an encrypted request over the HTTPS protocol.

IsSession

Indicates whether this cookie is a session cookie without expiration time.

Name

Gets the name of the cookie.

Path
: Gets the path on the server to which the browser returns this cookie. The cookie is visible to all sub-paths on the server.

SameSite
: Gets the `SameSite` attribute value of the cookie. This attribute denotes if your cookie is restricted to a first-party or same-site context.

Value
: Gets the value of the cookie.

## Methods

ToString()

# Property CreationTime

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## CreationTime

Gets the cookie creation time.

```
public DateTime CreationTime { get; }
```

## Property Value

DateTime

The creation time.

# Property DomainName

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## DomainName

Gets the domain name set for this cookie. It specifies allowed hosts to receive the cookie.

```
public string DomainName { get; }
```

## Property Value

string

The domain name.

# Property ExpirationTime

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## ExpirationTime

Gets the cookie expiration time.

```
public DateTime? ExpirationTime { get; }
```

## Property Value

DateTime?

The expiration time or null if the expiration time is not set.

# Property IsHttpOnly

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## IsHttpOnly

Gets the `HttpOnly` attribute value of the cookie. The `HttpOnly` cookie attribute can help to mitigate hijacking and XSS attacks by preventing access to cookie value through JavaScript.

```
public bool IsHttpOnly { get; }
```

## Property Value

bool

The `HttpOnly` attribute value.

# Property IsSecure

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## IsSecure

Gets the `Secure` attribute value of the cookie. A secure cookie is only sent to the server with an encrypted request over the HTTPS protocol.

```
public bool IsSecure { get; }
```

## Property Value

bool

The `Secure` attribute value.

# Property IsSession

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## IsSession

Indicates whether this cookie is a session cookie without expiration time.

```
public bool IsSession { get; }
```

## Property Value

bool

`true` if this cookie is a session cookie without expiration time, `false` if this cookie is a persistent cookie that has expiration time.

# Property Name

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Name

Gets the name of the cookie.

```
public string Name { get; }
```

## Property Value

string

The cookie's name.

# Property Path

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Path

Gets the path on the server to which the browser returns this cookie. The cookie is visible to all sub-paths on the server.

```
public string Path { get; }
```

## Property Value

string

The path on the server, for example, */catalog*.

## Remarks

`Path` indicates a URL path that must exist in the requested URL in order to send the `Cookie` header. The %x2F ("/") character is considered a directory separator, and subdirectories will match as well.

For example, if `Path=/docs` is set, these paths will match:

- `/docs`
- `/docs/Web/`
- `/docs/Web/HTTP`

# Property SameSite

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## SameSite

Gets the `SameSite` attribute value of the cookie. This attribute denotes if your cookie is restricted to a first-party or same-site context.

```
public SameSite SameSite { get; }
```

## Property Value

SameSite

The `SameSite` attribute value.

## Remarks

By default, the attribute value is set to LaxMode which means cookies are only set when the domain in the URL of the browser matches the domain of the cookie — a first-party cookie.

# Property Value

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Value

Gets the value of the cookie.

```
public string Value { get; }
```

## Property Value

string

The cookie's value.

# Method ToString

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class Cookie.Builder

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

A builder class to construct cookie.

```
public class Cookie.Builder
```

**Inheritance**

object ← Cookie.Builder

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Remarks

Each of the properties modifies the state of the builder. Builders are not thread-safe and should not be used concurrently from multiple threads without external synchronization.

## Constructors

Builder(string)
: Creates a new Cookie builder.

## Properties

CreationTime
: Gets or sets cookie creation time.

DomainName
: Gets the domain name for this cookie.

ExpirationTime
: Gets or sets cookie expiration time.

HttpOnly
: Gets or sets the *HttpOnly* attribute. The `HttpOnly` cookie attribute can help to mitigate hijacking and XSS attacks by preventing access to cookie value through JavaScript.

Name
: Gets or sets the name of the cookie.

Path
: Gets or sets the path on the server to which the browser returns this cookie. The cookie is visible to all sub-paths on the server.

SameSite
: Gets or sets the `SameSite` attribute value of the cookie. This attribute denotes if your cookie should be restricted to a first-party or same-site context.

Secure
: Gets or sets the `Secure` attribute value of the cookie. A secure cookie is only sent to the server with an encrypted request over the HTTPS protocol.

Value
: Gets or sets the value of the cookie.

## Methods

Build()
: Builds an HTTP cookie.

# Constructor Builder

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Builder(string)

Creates a new Cookie builder.

```
public Builder(string domainName)
```

## Parameters

`domainName` string

The name of the domain this cookie belongs too. Please note, that the domain is also required for cookies with `__Host` prefix.

# Property CreationTime

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## CreationTime

Gets or sets cookie creation time.

```
public DateTime CreationTime { get; set; }
```

## Property Value

DateTime

The cookie creation time.

# Property DomainName

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## DomainName

Gets the domain name for this cookie.

```
public string DomainName { get; }
```

## Property Value

string

The domain name.

# Property ExpirationTime

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## ExpirationTime

Gets or sets cookie expiration time.

```
public DateTime? ExpirationTime { get; set; }
```

## Property Value

DateTime?

The cookie expiration time. This value is null for session cookies.

# Property HttpOnly

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## HttpOnly

Gets or sets the *HttpOnly* attribute. The `HttpOnly` cookie attribute can help to mitigate hijacking and XSS attacks by preventing access to cookie value through JavaScript.

```
public bool HttpOnly { get; set; }
```

## Property Value

bool

The `HttpOnly` attribute value.

# Property Name

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Name

Gets or sets the name of the cookie.

```
public string Name { get; set; }
```

## Property Value

string

The cookie's name.

## Exceptions

ArgumentException

The `value` is null, empty or blank.

# Property Path

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Path

Gets or sets the path on the server to which the browser returns this cookie. The cookie is visible to all sub-paths on the server.

```
public string Path { get; set; }
```

## Property Value

string

The path on the server, for example, */catalog*.

## Remarks

`Path` indicates a URL path that must exist in the requested URL in order to send the `Cookie` header. The %x2F ("/") character is considered a directory separator, and subdirectories will match as well.

For example, if `Path=/docs` is set, these paths will match:

- `/docs`
- `/docs/Web/`
- `/docs/Web/HTTP`

## Exceptions

ArgumentException

The `value` is null, empty or blank.

# Property SameSite

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## SameSite

Gets or sets the `SameSite` attribute value of the cookie. This attribute denotes if your cookie should be restricted to a first-party or same-site context.

```
public SameSite SameSite { get; set; }
```

## Property Value

SameSite

The `SameSite` attribute value.

## Remarks

By default, the attribute value is set to LaxMode which means cookies are only set when the domain in the URL of the browser matches the domain of the cookie — a first-party cookie.

The None value requires that the Secure attribute is set to `true`.

# Property Secure

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Secure

Gets or sets the `Secure` attribute value of the cookie. A secure cookie is only sent to the server with an encrypted request over the HTTPS protocol.

```
public bool Secure { get; set; }
```

## Property Value

bool

The `Secure` attribute value.

# Property Value

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Value

Gets or sets the value of the cookie.

```
public string Value { get; set; }
```

## Property Value

string

The cookie's value.

## Exceptions

ArgumentException

The `value` is null, empty or blank.

# Method Build

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Build()

Builds an HTTP cookie.

```
public Cookie Build()
```

## Returns

Cookie

The constructed Cookie instance.

# Interface ICookieStore

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

The system for storing and retrieving cookies. The cookies can be stored in the process memory (session cookies) or in files (persistent cookies). The ICookieStore instance provides access to both session and persistent cookies.

```
public interface ICookieStore : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Remarks

The persistent cookies are stored in the Browser cache directory.

So, if Browser A and B have the same cache directory, then they will access the cookies of each other.

If you need to configure each Browser to use unique cookie storage which is not accessible for other Browser instances, you need to provide unique user data directory for each Browser instance. The user data directory path can be provided via configured UserDataDirectory object that must be passed into the `EngineFactory.Create` method.

## Properties

Engine
Gets the IEngine instance associated with this object.

Profile
Gets the IProfile instance associated with this object.

## Methods

Delete(Cookie)

Deletes one specific cookie. The cookie instance can be received from a list of cookies returned from the GetAllCookies(string) method.

DeleteAllCookies()

Deletes all of the cookies including session, secure or HTTP only cookies.

Flush()

Use this method to save all the changes you apply to this cookie storage. By default all the changes to the cookie store are made in memory, so when you restart the application you will not see the changes you made if you don't invoke this method on application exit. You can invoke this method after every change you made to cookie storage.

GetAllCookies(string)

Returns all the cookies for given url including HTTP only cookies.

SetCookie(Cookie)

Sets a session cookie given explicit user-provided cookie attributes.

# Property Engine

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Profile

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Method Delete

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Delete(Cookie)

Deletes one specific cookie. The cookie instance can be received from a list of cookies returned from the GetAllCookies(string) method.

```
Task Delete(Cookie cookie)
```

### Parameters

`cookie` Cookie

cookie to delete.

### Returns

Task

A task that completes when the cookie was deleted.

### Exceptions

ObjectDisposedException

The ICookieStore has already been disposed.

# Method DeleteAllCookies

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## DeleteAllCookies()

Deletes all of the cookies including session, secure or HTTP only cookies.

```
Task<int> DeleteAllCookies()
```

## Returns

Task<int>

The number of deleted cookies.

## Exceptions

ObjectDisposedException

The ICookieStore has already been disposed.

# Method Flush

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## Flush()

Use this method to save all the changes you apply to this cookie storage. By default all the changes to the cookie store are made in memory, so when you restart the application you will not see the changes you made if you don't invoke this method on application exit. You can invoke this method after every change you made to cookie storage.

```
void Flush()
```

## Exceptions

ObjectDisposedException

The ICookieStore has already been disposed.

# Method GetAllCookies

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## GetAllCookies(string)

Returns all the cookies for given url including HTTP only cookies.

```
Task<IEnumerable<Cookie>> GetAllCookies(string url = null)
```

## Parameters

`url` string

The URL associated with the returned cookies.

## Returns

Task<IEnumerable<Cookie>>

The collection of all the cookies or an empty collection when there's no cookies.

## Exceptions

ObjectDisposedException

The ICookieStore has already been disposed.

# Method SetCookie

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

## SetCookie(Cookie)

Sets a session cookie given explicit user-provided cookie attributes.

```
Task<bool> SetCookie(Cookie cookie)
```

### Parameters

`cookie` Cookie

An HTTP cookie.

### Returns

Task<bool>

`true` when session cookie was inserted successfully, `false` otherwise.

### Remarks

If you set the cookie successfully and the method returns `true` and you decide to find the cookie in the list of all cookies, please note that the cookie storage can modify some cookies attributes, such as domain or expiration time.

### Exceptions

ArgumentException

The `cookie` is null or the cookie cannot be set.

ObjectDisposedException

The ICookieStore has already been disposed.

# Enum SameSite

Namespace: DotNetBrowser.Cookies

Assembly: DotNetBrowser.dll

The SameSite cookie attribute values of the `Set-Cookie` HTTP response header. This attribute is used to declare in which context the cookies can be sent.

```
public enum SameSite
```

## Fields

`LaxMode = 2`

The cookie can be sent with top-level navigations from a third-party site to the web site with the domain matching the cookie. This is the default value in modern browsers.

`None = 3`

The cookie can be sent in a third-party context. To use this mode the cookie must also be marked as secure.

`StrictMode = 1`

The cookie can only be sent in a first-party context, when the cookie matches the domain of the current site.

`Unspecified = 0`

Reserved value.

# Namespace DotNetBrowser.DevTools

## Interfaces

IDevTools

Allows working with Chromium Developer Tools and access the remote debugging URL of the currently loaded web page in the browser instance associated with this DevTools instance.

# Interface IDevTools

Namespace: DotNetBrowser.DevTools

Assembly: DotNetBrowser.dll

Allows working with Chromium Developer Tools and access the remote debugging URL of the currently loaded web page in the browser instance associated with this DevTools instance.

```
public interface IDevTools : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

RemoteDebuggingUrl
: Gets the remote debugging URL of the currently loaded web page in the browser instance associated with this DevTools instance.

## Methods

Hide()
: Closes the Chromium Developer Tools window if any is shown.

Show()
: Opens the Chromium Developer Tools panel in a new window. This method does nothing if the DevTools window is already shown.

# Property RemoteDebuggingUrl

Namespace: DotNetBrowser.DevTools

Assembly: DotNetBrowser.dll

## RemoteDebuggingUrl

Gets the remote debugging URL of the currently loaded web page in the browser instance associated with this DevTools instance.

```
string RemoteDebuggingUrl { get; }
```

## Property Value

string

The remote debugging URL. Can be `null` if the remote debugging URL is not available, for instance, when the IEngine instance has not been configured with the remote debugging port.

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Method Hide

Namespace: DotNetBrowser.DevTools

Assembly: DotNetBrowser.dll

## Hide()

Closes the Chromium Developer Tools window if any is shown.

```
void Hide()
```

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Method Show

Namespace: DotNetBrowser.DevTools

Assembly: DotNetBrowser.dll

## Show()

Opens the Chromium Developer Tools panel in a new window. This method does nothing if the DevTools window is already shown.

```
void Show()
```

## Exceptions

ObjectDisposedException

The IBrowser has already been disposed.

# Namespace DotNetBrowser.Dom

## Classes

DomException
: Thrown when an operation on the DOM element fails.

PointInspection
: Provides information about a DOM node at the specified point inside the loaded document.

## Interfaces

IAttribute
: This interface is implemented by all INode implementations to support DOM event model.

IAttributes
: A dictionary that contains IElement attributes. Modifying this dictionary will lead to modifying the element attributes.

IDocument
: Represents DOM HTML document of the web page.

IElement
: This interface is implemented by all INode implementations to support DOM event model.

IFormControlElement
: Represents the form control element.

IFormElement
: Represents DOM HTML Form element.

IFrameElement
: Represents an HTML `<frame>` or `<iframe>` element.

IImageElement
: Represents DOM HTML image element `<img>`.

IInputElement
: Represents the DOM element with `<input>` tag.

INode
: This interface is implemented by all INode implementations to support DOM event model.

INodeCollection

Represents a collection of the DOM nodes.

IOptionElement
: Represents the DOM HTML `<option>` element.

ISearchContext
: The base interface for search that is implemented by the DOM objects that provide search mechanisms.

ISelectElement
: Represents DOM HTML `<select>` element.

ITextAreaElement
: Represents DOM HTML `<textarea>` element.

## Enums

AlignTo
: Represents the values that can be used to describe how the element will be aligned to the visible area of the scrollable ancestor.

DocumentPosition
: Enumeration of the document position that represent the relationship between two nodes within the Document.

NodeType
: Represents the node types that can be used to distinguish different kind of nodes (such as an HTML element, text, attribute) from each other.

# Enum AlignTo

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents the values that can be used to describe how the element will be aligned to the visible area of the scrollable ancestor.

```
public enum AlignTo
```

## Fields

`Bottom = 1`

The bottom of the element will be aligned to the bottom of the visible area of the scrollable ancestor.

`Top = 0`

The top of the element will be aligned to the top of the visible area of the scrollable ancestor.

## See Also

ScrollIntoView(AlignTo)

# Enum DocumentPosition

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Enumeration of the document position that represent the relationship between two nodes within the Document.

```
[Flags]
public enum DocumentPosition
```

## Fields

`ContainedBy = 16`

The other node contains the reference node.

`Contains = 8`

The reference node contains the other node.

`Disconnected = 1`

The nodes are disconnected.

`Following = 4`

The position of the other node follows the position of the reference node.

`ImplementationSpecific = 32`

The node positions depend on the DOM implementation and cannot be compared.

`Preceding = 2`

The position of the other node precedes the position of the reference node.

`Unspecified = 0`

The position is not specified.

# Class DomException

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Thrown when an operation on the DOM element fails.

```
public sealed class DomException : Exception, ISerializable
```

**Inheritance**

object ← Exception ← DomException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Constructors

DomException(string)

Initializes a new DomException instance.

# Constructor DomException

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## DomException(string)

Initializes a new DomException instance.

```
public DomException(string message)
```

## Parameters

`message` string

The message that describes the error.

# Interface IAttribute

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

This interface is implemented by all INode implementations to support DOM event model.

```
public interface IAttribute : INode, IEventTarget, ISearchContext, IAutoDisposable
```

**Inherited Members**

INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Owner

Gets an owner element of the attribute.

# Property Owner

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Owner

Gets an owner element of the attribute.

```
IElement Owner { get; }
```

## Property Value

IElement

The owner element of the attribute. If the owner does not exist, this property is `null`.

## Exceptions

ObjectDisposedException

The IAttribute has already been disposed.

# Interface IAttributes

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

A dictionary that contains IElement attributes. Modifying this dictionary will lead to modifying the element attributes.

```
public interface IAttributes : IAutoDisposable, IDictionary<string, string>,
ICollection<KeyValuePair<string, string>>, IEnumerable<KeyValuePair<string,
string>>, IEnumerable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed ,
IDictionary<string, string>.Add(string, string) , IDictionary<string, string>.ContainsKey(string) ,
IDictionary<string, string>.Remove(string) , IDictionary<string, string>.TryGetValue(string, out string) ,
IDictionary<string, string>.this[string] , IDictionary<string, string>.Keys ,
IDictionary<string, string>.Values ,
ICollection<KeyValuePair<string, string>>.Add(KeyValuePair<string, string>) ,
ICollection<KeyValuePair<string, string>>.Clear() ,
ICollection<KeyValuePair<string, string>>.Contains(KeyValuePair<string, string>) ,
ICollection<KeyValuePair<string, string>>.CopyTo(KeyValuePair<string, string>[], int) ,
ICollection<KeyValuePair<string, string>>.Remove(KeyValuePair<string, string>) ,
ICollection<KeyValuePair<string, string>>.Count ,
ICollection<KeyValuePair<string, string>>.IsReadOnly ,
IEnumerable<KeyValuePair<string, string>>.GetEnumerator() , IEnumerable.GetEnumerator()

## Methods

AsReadOnly()

Gets a read-only dictionary that contains a snapshot of the attributes of the current element.

# Method AsReadOnly

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## AsReadOnly()

Gets a read-only dictionary that contains a snapshot of the attributes of the current element.

```
IReadOnlyDictionary<string, string> AsReadOnly()
```

## Returns

IReadOnlyDictionary<string, string>

The dictionary that contains a snapshot of the attributes of the current element. The returned dictionary will be empty if the element doesn't have any attributes.

## Exceptions

ObjectDisposedException

The IAttributes has already been disposed.

# Interface IDocument

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents DOM HTML document of the web page.

```
public interface IDocument : INode, IEventTarget, ISearchContext, IAutoDisposable
```

**Inherited Members**

INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

BaseUri

Gets the `<base>` element's href attribute if one is present.

DocumentElement

Gets the document HTML element that usually represents HTML tag.

FocusedElement

Gets the currently focused element in the document.

## Methods

CreateElement(string)

Creates and returns a new DOM element with the specified tag name.

CreateEvent(EventType, EventParameters)

Creates a DOM event instance of the specified type.

CreateKeyDownEvent(KeyEventParameters)

Creates and returns a new `keyDown` IKeyEvent object with the given

```
parameters
```

.

CreateKeyPressEvent(KeyEventParameters)

Creates and returns a new `keyPress` IKeyEvent object with the given

```
parameters
```

.

CreateKeyUpEvent(KeyEventParameters)

Creates and returns a new `keyUp` IKeyEvent object with the given

```
parameters
```

.

CreateMouseEvent(EventType, MouseEventParameters)

Creates and returns a new IMouseEvent object with the given

```
eventType
```

and

```
parameters
```

.

CreateTextNode(string)

Returns a new Text DOM node with TextNode type.

# Property BaseUri

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## BaseUri

Gets the `<base>` element's href attribute if one is present.

```
string BaseUri { get; }
```

## Property Value

string

The `<base>` element's href attribute if one is present. If the currently loaded document doesn't have the `<base>` element, the method returns URL of the currently loaded document including "about:blank" when nothing is loaded.

# Property DocumentElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## DocumentElement

Gets the document HTML element that usually represents HTML tag.

```
IElement DocumentElement { get; }
```

## Property Value

IElement

The document HTML element that usually represents HTML tag. Can be null if the document element is not available (e.g. the web page of this document was disposed).

# Property FocusedElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## FocusedElement

Gets the currently focused element in the document.

```
IElement FocusedElement { get; }
```

## Property Value

IElement

The currently focused element in the document. Can be null if there is no focused element in this document.

## Exceptions

ObjectDisposedException

The IDocument has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method CreateElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## CreateElement(string)

Creates and returns a new DOM element with the specified tag name.

```
IElement CreateElement(string tagName)
```

## Parameters

`tagName` string

the tag name (e.g. "A", "P", "DIV") of the new DOM element.

## Returns

IElement

the new IElement or null if the `tagName` is incorrect.

## Exceptions

ArgumentException

The `tagName` is null, empty, or contains only blank characters.

# Method CreateEvent

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## CreateEvent(EventType, EventParameters)

Creates a DOM event instance of the specified type.

```
IEvent CreateEvent(EventType eventType, EventParameters parameters)
```

## Parameters

`eventType` EventType

the DOM event type.

`parameters` EventParameters

the parameters of the DOM event

## Returns

IEvent

DOM Event object

## Exceptions

ArgumentNullException

when eventType or parameters values are null.

# Method CreateKeyDownEvent

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## CreateKeyDownEvent(KeyEventParameters)

Creates and returns a new `keyDown` IKeyEvent object with the given

```
parameters
```

.

```
IKeyEvent CreateKeyDownEvent(KeyEventParameters parameters)
```

## Parameters

`parameters` KeyEventParameters

a KeyEventParametersobject representing key event properties. Cannot be null.

## Returns

IKeyEvent

the new IKeyEvent object.

## Exceptions

ArgumentException

when parameters value is null.

# Method CreateKeyPressEvent

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## CreateKeyPressEvent(KeyEventParameters)

Creates and returns a new `keyPress` IKeyEvent object with the given

```
parameters
```

.

```
IKeyEvent CreateKeyPressEvent(KeyEventParameters parameters)
```

## Parameters

`parameters` KeyEventParameters

a KeyEventParametersobject representing key event properties. Cannot be null.

## Returns

IKeyEvent

the new IKeyEvent object.

## Exceptions

ArgumentException

when parameters value is null.

# Method CreateKeyUpEvent

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## CreateKeyUpEvent(KeyEventParameters)

Creates and returns a new `keyUp` IKeyEvent object with the given

```
parameters
```

.

```
IKeyEvent CreateKeyUpEvent(KeyEventParameters parameters)
```

## Parameters

`parameters` KeyEventParameters

a KeyEventParametersobject representing key event properties. Cannot be null.

## Returns

IKeyEvent

the new IKeyEvent object.

## Exceptions

ArgumentException

when parameters value is null.

# Method CreateMouseEvent

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## CreateMouseEvent(EventType, MouseEventParameters)

Creates and returns a new IMouseEvent object with the given

```
eventType
```

and

```
parameters
```

.

```
IMouseEvent CreateMouseEvent(EventType eventType, MouseEventParameters parameters)
```

## Parameters

`eventType` EventType

an EventTypeobject representing a type of a valid DOM event supported by engine. Cannot be null.

`parameters` MouseEventParameters

a MouseEventParameters object representing mouse event properties (e.g. bubbles, cancellable). Cannot be null.

## Returns

IMouseEvent

the new IMouseEvent object.

## Exceptions

ArgumentException

when eventType or parameters values are null.

# Method CreateTextNode

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## CreateTextNode(string)

Returns a new Text DOM node with TextNode type.

```
INode CreateTextNode(string text = null)
```

## Parameters

`text` string

the string, which will be used to initialize node value.

## Returns

INode

the new Text DOM node with TextNode type.

# Interface IElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

This interface is implemented by all INode implementations to support DOM event model.

```
public interface IElement : INode, IEventTarget, ISearchContext, IAutoDisposable
```

**Inherited Members**

INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

AttributeNodes

Gets a list that contains attribute nodes of the element. Each list entry represents an attribute node.

Attributes

Gets a dictionary that contains attributes of the current element. Modifying this dictionary will lead to modifying the element attributes.

BoundingClientRect

Gets the bounds of the element and its position relative to the top-left of the viewport of the current document.

InnerHtml

Gets or sets the inner HTML of the current element.

InnerText

Gets or sets the inner text of the current element.

OuterHtml
: Gets or sets the outer HTML of the current element.

## Methods

Blur()
: Removes focus from the current element.

Focus()
: Gives focus to the current element (if it can be focused).

ScrollIntoView(AlignTo)
: Scrolls the element's parent container such that the element on which this method is called is visible to the user.

# Property AttributeNodes

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## AttributeNodes

Gets a list that contains attribute nodes of the element. Each list entry represents an attribute node.

```
IReadOnlyList<IAttribute> AttributeNodes { get; }
```

## Property Value

IReadOnlyList<IAttribute>

A list that contains attribute nodes of the element. The list will be empty if the element does not have attributes.

## Exceptions

ObjectDisposedException

The IElement has already been disposed.

# Property Attributes

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Attributes

Gets a dictionary that contains attributes of the current element. Modifying this dictionary will lead to modifying the element attributes.

```
IAttributes Attributes { get; }
```

## Property Value

IAttributes

A dictionary that contains attributes of the current element. Modifying this dictionary will lead to modifying the element attributes.

# Property BoundingClientRect

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## BoundingClientRect

Gets the bounds of the element and its position relative to the top-left of the viewport of the current document.

```
Rectangle BoundingClientRect { get; }
```

## Property Value

Rectangle

The bounds of the element and its position relative to the top-left of the viewport of the current document.

## Remarks

The amount of scrolling that has been done of the viewport area (or any other scrollable element) is taken into account when computing the bounding rectangle. This means that the rectangle's boundary edges (top, left, bottom, and right) change their values every time the scrolling position changes (because their values are relative to the viewport and not absolute). If you need the bounding rectangle relative to the top-left corner of the document, just add the current scrolling position to the top and left properties (these can be obtained using `window.scrollX` and `window.scrollY`) to get a bounding rectangle which is independent from the current scrolling position.

## Exceptions

ObjectDisposedException

The IElement has already been disposed.

# Property InnerHtml

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## InnerHtml

Gets or sets the inner HTML of the current element.

```
string InnerHtml { get; set; }
```

## Property Value

string

The inner HTML of the current element. If the current element doesn't have inner HTML or the element doesn't exist anymore, this will be an empty string.

## Exceptions

DomException

The value was not set.

ObjectDisposedException

The IElement has already been disposed.

# Property InnerText

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## InnerText

Gets or sets the inner text of the current element.

```
string InnerText { get; set; }
```

## Property Value

string

The inner text of the current element. If the current element doesn't have inner text or the element doesn't exist anymore, this will be an empty string.

## Exceptions

DomException

The value was not set.

ObjectDisposedException

The IElement has already been disposed.

# Property OuterHtml

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## OuterHtml

Gets or sets the outer HTML of the current element.

```
string OuterHtml { get; set; }
```

## Property Value

string

The string that represents an HTML serialization of this element and its descendants.

## Remarks

While the element will be replaced in the document, the variable whose `OuterHtml` property was set will still hold a reference to the original element. See https://developer.mozilla.org/en-US/docs/Web/API/Element/outerHTML MDN web docs.

## Exceptions

ArgumentException

The value is set to null.

DomException

The value was not set.

# Method Blur

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Blur()

Removes focus from the current element.

```
void Blur()
```

## Exceptions

ObjectDisposedException

The IElement has already been disposed.

# Method Focus

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Focus()

Gives focus to the current element (if it can be focused).

```
void Focus()
```

## Exceptions

ObjectDisposedException

The IElement has already been disposed.

# Method ScrollIntoView

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## ScrollIntoView(AlignTo)

Scrolls the element's parent container such that the element on which this method is called is visible to the user.

```
void ScrollIntoView(AlignTo alignTo)
```

## Parameters

`alignTo` AlignTo

Describes how the element will be aligned to the visible area of the scrollable ancestor

## Remarks

Note that the element may not be scrolled completely to the top or bottom depending on the layout of other elements.

## Exceptions

ObjectDisposedException

The IElement has already been disposed.

# Interface IFormControlElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents the form control element.

```
public interface IFormControlElement : IElement, INode, IEventTarget,
ISearchContext, IAutoDisposable
```

**Inherited Members**

IElement.AttributeNodes , IElement.Attributes , IElement.BoundingClientRect , IElement.InnerHtml , IElement.InnerText , IElement.OuterHtml , IElement.Blur() , IElement.Focus() , IElement.ScrollIntoView(AlignTo) , INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Form

Gets the `<form>` element containing this control.

IsEnabled

Indicates whether form control is enabled or disabled.

Value

Gets or sets the form control value.

# Property Form

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Form

Gets the `<form>` element containing this control.

```
IFormElement Form { get; }
```

## Property Value

IFormElement

The `<form>` element containing this control. Can be `null` if this control is not within the context of a form.

## Exceptions

ObjectDisposedException

The IFormControlElement has already been disposed.

# Property IsEnabled

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsEnabled

Indicates whether form control is enabled or disabled.

```
bool IsEnabled { get; }
```

## Property Value

bool

`true` if the control is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IFormControlElement has already been disposed.

# Property Value

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Value

Gets or sets the form control value.

```
string Value { get; set; }
```

## Property Value

string

The form control value.

## Exceptions

ObjectDisposedException

The IFormControlElement has already been disposed.

ArgumentException

The `value` is null.

# Interface IFormElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents DOM HTML Form element.

```
public interface IFormElement : IElement, INode, IEventTarget,
ISearchContext, IAutoDisposable
```

**Inherited Members**

IElement.AttributeNodes , IElement.Attributes , IElement.BoundingClientRect , IElement.InnerHtml , IElement.InnerText , IElement.OuterHtml , IElement.Blur() , IElement.Focus() , IElement.ScrollIntoView(AlignTo) , INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Action

Gets the 'action' attribute value.

Controls

Gets the collection of all control elements in the form.

Method

Gets the 'method' attribute value.

Name

Gets the 'name' attribute value.

# Methods

Reset()

Restores the form elements' default values. It performs the same action as a reset button.

Submit()

Submits the form. It performs the same action as a submit button.

# Property Action

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Action

Gets the 'action' attribute value.

```
string Action { get; }
```

## Property Value

string

The 'action' attribute value.

## Exceptions

ObjectDisposedException

The IFormElement has already been disposed.

# Property Controls

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Controls

Gets the collection of all control elements in the form.

```
IEnumerable<IFormControlElement> Controls { get; }
```

## Property Value

IEnumerable<IFormControlElement>

The collection of all control elements in the form.

## Exceptions

ObjectDisposedException

The IFormElement has already been disposed.

# Property Method

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Method

Gets the 'method' attribute value.

```
string Method { get; }
```

## Property Value

string

The 'method' attribute value.

## Exceptions

ObjectDisposedException

The IFormElement has already been disposed.

# Property Name

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Name

Gets the 'name' attribute value.

```
string Name { get; }
```

## Property Value

string

The 'name' attribute value.

## Exceptions

ObjectDisposedException

The IFormElement has already been disposed.

# Method Reset

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Reset()

Restores the form elements' default values. It performs the same action as a reset button.

```
void Reset()
```

## Exceptions

ObjectDisposedException

The IFormElement has already been disposed.

# Method Submit

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Submit()

Submits the form. It performs the same action as a submit button.

```
void Submit()
```

## Exceptions

ObjectDisposedException

The IFormElement has already been disposed.

# Interface IFrameElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents an HTML `<frame>` or `<iframe>` element.

```
public interface IFrameElement : IElement, INode, IEventTarget,
ISearchContext, IAutoDisposable
```

**Inherited Members**

IElement.AttributeNodes , IElement.Attributes , IElement.BoundingClientRect , IElement.InnerHtml , IElement.InnerText , IElement.OuterHtml , IElement.Blur() , IElement.Focus() , IElement.ScrollIntoView(AlignTo) , INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

ContentDocument

Gets the document of the current `<frame>` or `<iframe>` element.

# Property ContentDocument

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## ContentDocument

Gets the document of the current `<frame>` or `<iframe>` element.

```
IDocument ContentDocument { get; }
```

## Property Value

IDocument

The document of the current `<frame>` or `<iframe>` element if it exists, otherwise `null`.

## Exceptions

ObjectDisposedException

The IFrameElement has already been disposed.

# Interface IImageElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents DOM HTML image element `<img>`.

```
public interface IImageElement : IElement, INode, IEventTarget,
ISearchContext, IAutoDisposable
```

**Inherited Members**

IElement.AttributeNodes , IElement.Attributes , IElement.BoundingClientRect , IElement.InnerHtml , IElement.InnerText , IElement.OuterHtml , IElement.Blur() , IElement.Focus() , IElement.ScrollIntoView(AlignTo) , INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Contents

Gets the image displayed by the element.

# Property Contents

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Contents

Gets the image displayed by the element.

```
Bitmap Contents { get; }
```

## Property Value

Bitmap

The Bitmap with the raw pixels of the image displayed by the element.

## Remarks

The size of the bitmap depends on the size of the original image. It is not affected by the size specified for the element.

## Exceptions

ObjectDisposedException

The IImageElement has already been disposed.

# Interface IInputElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents the DOM element with `<input>` tag.

```
public interface IInputElement : IFormControlElement, IElement, INode, IEventTarget,
ISearchContext, IAutoDisposable
```

**Inherited Members**

IFormControlElement.Form , IFormControlElement.IsEnabled , IFormControlElement.Value , IElement.AttributeNodes , IElement.Attributes , IElement.BoundingClientRect , IElement.InnerHtml , IElement.InnerText , IElement.OuterHtml , IElement.Blur() , IElement.Focus() , IElement.ScrollIntoView(AlignTo) , INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Checked

Gets or sets the checked attribute of the input DOM element with the type `'checkbox'` or `'radio'`.

File

Gets or sets an absolute or relative path to a file if the current input DOM element has the type attribute with the 'file' value.

Files

Gets or sets the collection of absolute or relative paths to files if the current input DOM element has the type attribute with the 'file' value.

IsCheckBox

Indicates whether the DOM element's type attribute has the 'checkbox' value.

IsEmailField

Indicates whether the DOM element's type attribute has the 'email' value.

IsFile

Indicates whether the DOM element's type attribute has the 'file' value.

IsMultipleFile

Indicates whether the DOM element has both the type attribute with the 'file' value, and the multiple attribute, for example: `<input type="file" multiple>`

IsPasswordField

Indicates whether the DOM element's type attribute has the 'password' value.

IsRadioButton

Indicates whether the DOM element's type attribute has the 'radio' value.

IsText

Indicates if this input element is a text field and the type attribute value of the input HTML element is 'number'.

IsTextField

Indicates if the DOM element's type attribute has the `'text'` value.

# Property Checked

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Checked

Gets or sets the checked attribute of the input DOM element with the type `'checkbox'` or `'radio'`.

```
bool Checked { get; set; }
```

## Property Value

bool

`true` if the `'checkbox'` or `'radio'` element has non-empty value of the `'checked'` attribute.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Property File

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## File

Gets or sets an absolute or relative path to a file if the current input DOM element has the type attribute with the 'file' value.

```
string File { get; set; }
```

## Property Value

string

The absolute or relative path to a file if the current input DOM element has the type attribute with the `'file'` value. If this DOM element has the `multiple` attribute, this will be the first of file paths. If the element has no file selected, this will be an empty string.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

ArgumentException

The `value` is null, empty, or contains only blank characters.

# Property Files

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Files

Gets or sets the collection of absolute or relative paths to files if the current input DOM element has the type attribute with the 'file' value.

```
IEnumerable<string> Files { get; set; }
```

## Property Value

IEnumerable<string>

The collection of absolute or relative paths to files if the current input DOM element has the type attribute with the `'file'` value. If the element has no files selected, this will be an empty string.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

ArgumentException

The `value` is null, empty, or contains only blank characters.

# Property IsCheckBox

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsCheckBox

Indicates whether the DOM element's type attribute has the 'checkbox' value.

```
bool IsCheckBox { get; }
```

## Property Value

bool

`true` if the DOM element's type attribute has the 'checkbox' value. When the return value is true, the checked state of the DOM element can be retrieved or changed via the Checked property.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Property IsEmailField

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsEmailField

Indicates whether the DOM element's type attribute has the 'email' value.

```
bool IsEmailField { get; }
```

## Property Value

bool

`true` if the DOM element's type attribute has the 'email' value.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Property IsFile

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsFile

Indicates whether the DOM element's type attribute has the 'file' value.

```
bool IsFile { get; }
```

## Property Value

bool

`true` if the DOM element's type attribute has the 'file' value.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Property IsMultipleFile

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsMultipleFile

Indicates whether the DOM element has both the type attribute with the 'file' value, and the multiple attribute, for example: `<input type="file" multiple>`

```
bool IsMultipleFile { get; }
```

## Property Value

bool

`true` if the DOM element has both the type attribute with the 'file' value, and the multiple attribute.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Property IsPasswordField

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsPasswordField

Indicates whether the DOM element's type attribute has the 'password' value.

```
bool IsPasswordField { get; }
```

## Property Value

bool

`true` if the DOM element's type attribute has the 'password' value.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Property IsRadioButton

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsRadioButton

Indicates whether the DOM element's type attribute has the 'radio' value.

```
bool IsRadioButton { get; }
```

## Property Value

bool

`true` if the DOM element's type attribute has the 'radio' value.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Property IsText

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsText

Indicates if this input element is a text field and the type attribute value of the input HTML element is 'number'.

```
bool IsText { get; }
```

## Property Value

bool

`true` when IsTextField returns true and the type attribute value of the input HTML element is 'number'.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Property IsTextField

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## IsTextField

Indicates if the DOM element's type attribute has the `'text'` value.

```
bool IsTextField { get; }
```

## Property Value

bool

`true` if the DOM element's type attribute has the 'text' value.

## Exceptions

ObjectDisposedException

The IInputElement has already been disposed.

# Interface INode

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

This interface is implemented by all INode implementations to support DOM event model.

```
public interface INode : IEventTarget, ISearchContext, IAutoDisposable
```

**Inherited Members**

IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Children

Gets the collection of the child nodes. Modifying this collection will lead to the DOM tree modification.

Document

Gets the document instance containing this node.

Frame

Gets the frame containing this node.

NextSibling

Gets the next sibling node in the document tree.

NodeName

Gets the name of this node, depending on its NodeType.

NodeValue

Gets the value of this node, depending on its NodeType.

Parent

Gets the parent node.

PreviousSibling
Gets the previous sibling node in the document tree.

TextContent
Gets or sets the text content of the node and its descendants.

Type
Gets the type of this node.

XPath
Gets an XPath for the current Node.

## Methods

Click()
Simulates click on the current Node.

CompareDocumentPosition(INode)
Compares position of the current node against another node in a DOM tree.

Evaluate(string, XPathResultType)
Evaluates an XPath expression.

# Property Children

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Children

Gets the collection of the child nodes. Modifying this collection will lead to the DOM tree modification.

```
INodeCollection Children { get; }
```

## Property Value

INodeCollection

The collection of the child nodes.

# Property Document

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Document

Gets the document instance containing this node.

```
IDocument Document { get; }
```

## Property Value

IDocument

The document instance containing this node.

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Property Frame

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Frame

Gets the frame containing this node.

```
IFrame Frame { get; }
```

## Property Value

IFrame

The frame containing this node.

# Property NextSibling

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## NextSibling

Gets the next sibling node in the document tree.

```
INode NextSibling { get; }
```

## Property Value

INode

The next sibling node in the document tree if it exists, otherwise returns `null`.

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Property NodeName

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## NodeName

Gets the name of this node, depending on its NodeType.

```
string NodeName { get; }
```

## Property Value

string

The name of the node.

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Property NodeValue

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## NodeValue

Gets the value of this node, depending on its NodeType.

```
string NodeValue { get; set; }
```

## Property Value

string

The value of the node.

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Property Parent

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Parent

Gets the parent node.

```
INode Parent { get; }
```

## Property Value

INode

The parent node, if it exists, otherwise returns `null`

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Property PreviousSibling

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## PreviousSibling

Gets the previous sibling node in the document tree.

```
INode PreviousSibling { get; }
```

## Property Value

INode

The previous sibling node in the document tree if such node exists, otherwise returns `null`

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Property TextContent

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## TextContent

Gets or sets the text content of the node and its descendants.

```
string TextContent { get; set; }
```

## Property Value

string

## Remarks

Setting this property on a node removes all of its children and replaces them with a single text node with the given value.

The property value depends on the type of the node, whhich can be checked via the Type property.

The property value is an empty string if the element is a document, a document type, or a notation.

If the node is a CDATA section, a comment, a processing instruction, or a text node, this method returns the text inside this node (the NodeValue).

For other node types, this method returns the concatenation of the TextContent attribute value of every child node, excluding comments and processing instruction nodes. This is an empty string if the node has no children.

Differences from InnerText:

- The TextContent property gets the content of all elements, including <script> and <style > elements, and the mostly equivalent `InnerText` property does not.
- `InnerText` is also aware of style and will not return the text of hidden elements, whereas TextContent will.
- As `InnerText` is aware of CSS styling, it will trigger a reflow, whereas TextContent will not.

## Exceptions

ArgumentException

The `value` is null.

ObjectDisposedException

The INode has already been disposed.

# Property Type

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Type

Gets the type of this node.

```
NodeType Type { get; }
```

## Property Value

NodeType

The type of this node.

# Property XPath

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## XPath

Gets an XPath for the current Node.

```
string XPath { get; }
```

## Property Value

string

The XPath for the current Node or an empty string if such XPath is not available.

# Method Click

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Click()

Simulates click on the current Node.

```
void Click()
```

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Method CompareDocumentPosition

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## CompareDocumentPosition(INode)

Compares position of the current node against another node in a DOM tree.

```
DocumentPosition CompareDocumentPosition(INode otherNode)
```

## Parameters

`otherNode` INode

The node to be compared to the current node

## Returns

DocumentPosition

The document position of the node specified by the `otherNode` parameter relates to the position of the current node.

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Method Evaluate

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Evaluate(string, XPathResultType)

Evaluates an XPath expression.

```
IXPathResult Evaluate(string expression, XPathResultType type = XPathResultType.Any)
```

### Parameters

`expression` string

a string representing the XPath to be evaluated. Cannot be null or empty string.

`type` XPathResultType

type of result XPathResult to return. See XPathResultType

### Returns

IXPathResult

The IXPathResult object of the type specified in the `type` parameter. The return value will be always a valid IXPathResult object.

### Exceptions

ArgumentException

The `expression` is null, empty, or contains blank characters only.

ObjectDisposedException

The INode has already been disposed.

XPathException

The XPath evaluation error occurs.

# Interface INodeCollection

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents a collection of the DOM nodes.

```
public interface INodeCollection : IEnumerable<INode>, IEnumerable
```

**Inherited Members**

IEnumerable<INode>.GetEnumerator()

## Methods

Append(INode)

Appends a node as the last item of the collection. The new node could be an existing node in the document, or a new node. If the node is existing node, it will be moved to new location in the document.

AsReadOnly()

Returns the snapshot of the items in the collection as a read-only list.

Insert(INode, INode)

Inserts a new `node` before an existing `beforeNode`.

Remove(INode)

Removes a node from the collection.

Replace(INode, INode)

Replaces an existing child node with a new node. The new node could be an existing node in the document, or you can create a new node. If the `newNode` is an existing node, it will be moved to new location in the document.

# Method Append

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Append(INode)

Appends a node as the last item of the collection. The new node could be an existing node in the document, or a new node. If the node is existing node, it will be moved to new location in the document.

```
bool Append(INode node)
```

### Parameters

`node` INode

the node to append.

### Returns

bool

`true` if the node was successfully appended, `false` otherwise.

### Exceptions

ArgumentNullException

The `node` is null.

ObjectDisposedException

The INode has already been disposed.

# Method AsReadOnly

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## AsReadOnly()

Returns the snapshot of the items in the collection as a read-only list.

```
IReadOnlyList<INode> AsReadOnly()
```

## Returns

IReadOnlyList<INode>

the snapshot of the items in the collection as a read-only list.

## Exceptions

ObjectDisposedException

The INode has already been disposed.

# Method Insert

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Insert(INode, INode)

Inserts a new `node` before an existing `beforeNode`.

```
bool Insert(INode node, INode beforeNode)
```

## Parameters

`node` INode

The node to insert.

`beforeNode` INode

The node before which the node will be inserted.

## Returns

bool

`true` if node was successfully inserted, `false` otherwise.

## Remarks

The `node` could be an existing node in the document, or you can create and insert a new node. If the node is an existing node, it will be moved to new location in the document.

## Exceptions

ArgumentException

The `node` or `beforeNode` is null.

ObjectDisposedException

The INode has already been disposed.

# Method Remove

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Remove(INode)

Removes a node from the collection.

```
bool Remove(INode node)
```

## Parameters

`node` INode

The node to remove.

## Returns

bool

`true` if the node was successfully removed from the collection.

## Exceptions

ArgumentNullException

The `node` is null.

ObjectDisposedException

The INode has already been disposed.

# Method Replace

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Replace(INode, INode)

Replaces an existing child node with a new node. The new node could be an existing node in the document, or you can create a new node. If the `newNode` is an existing node, it will be moved to new location in the document.

```
bool Replace(INode newNode, INode oldNode)
```

## Parameters

`newNode` INode

the new node for replacement.

`oldNode` INode

the existing node, which will be replaced by `newNode`.

## Returns

bool

`true` if the `oldNode` was successfully replaced with the `newNode`.

## Remarks

The old node could be used for inserting/appending it into the document later.

## Exceptions

ArgumentException

The `newNode` or the `oldNode` is null.

ObjectDisposedException

The INode has already been disposed.

# Interface IOptionElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents the DOM HTML `<option>` element.

```
public interface IOptionElement : IElement, INode, IEventTarget,
ISearchContext, IAutoDisposable
```

**Inherited Members**

IElement.AttributeNodes , IElement.Attributes , IElement.BoundingClientRect , IElement.InnerHtml , IElement.InnerText , IElement.OuterHtml , IElement.Blur() , IElement.Focus() , IElement.ScrollIntoView(AlignTo) , INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Selected

Gets or sets the state of the option element.

# Property Selected

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Selected

Gets or sets the state of the option element.

```
bool Selected { get; set; }
```

## Property Value

bool

`true` if option element is selected, `false` otherwise.

## Exceptions

ObjectDisposedException

The IOptionElement has already been disposed.

# Interface ISearchContext

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

The base interface for search that is implemented by the DOM objects that provide search mechanisms.

```
public interface ISearchContext : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Methods

GetElementByClassName(string)

Finds the first IElement object in the current search context by the given `className`.

GetElementByCssSelector(string)

Finds the first IElement object in the current search context by the given CSS selector.

GetElementById(string)

Finds the first IElement object in the current search context by the given `id`.

GetElementByName(string)

Finds the first IElement object in the current search context by the given `name`.

GetElementByTagName(string)

Finds the first IElement object in the current search context by the given tag name.

GetElementsByClassName(string)

Finds all IElement objects in the current search context by the given `className`.

GetElementsByCssSelector(string)

Finds all IElement objects in the current search context by the given `cssSelector`.

GetElementsByName(string)

Finds all IElement objects in the current search context by the given `name`.

GetElementsByTagName(string)

Finds all IElement objects in the current search context by the given tag name.

# Method GetElementByClassName

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementByClassName(string)

Finds the first IElement object in the current search context by the given `className`.

```
IElement GetElementByClassName(string className)
```

## Parameters

`className` string

The class attribute of the HTML element.

## Returns

IElement

The first IElement object found in the current search context by the given `className` or `null` if no elements were found.

## Exceptions

ArgumentException

The `className` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Method GetElementByCssSelector

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementByCssSelector(string)

Finds the first IElement object in the current search context by the given CSS selector.

```
IElement GetElementByCssSelector(string cssSelector)
```

### Parameters

`cssSelector` string

The CSS selector.

### Returns

IElement

The first IElement object found in the current search context by the given CSS selector or `null` if no elements were found.

### Exceptions

ArgumentException

The `cssSelector` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Method GetElementById

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementById(string)

Finds the first IElement object in the current search context by the given `id`.

```
IElement GetElementById(string id)
```

## Parameters

`id` string

a string that represents the id attribute of the HTML element.

## Returns

IElement

The first IElement object found in the current search context by the given `id` or `null` if no elements were found.

## Exceptions

ArgumentException

The `id` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Method GetElementByName

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementByName(string)

Finds the first IElement object in the current search context by the given `name`.

```
IElement GetElementByName(string name)
```

## Parameters

`name` string

The `name` attribute of the HTML element.

## Returns

IElement

The first IElement object found in the current search context by the given `name` or `null` if no elements were found.

## Exceptions

ArgumentException

The `name` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Method GetElementByTagName

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementByTagName(string)

Finds the first IElement object in the current search context by the given tag name.

```
IElement GetElementByTagName(string tagName)
```

## Parameters

`tagName` string

The tag name of the HTML element.

## Returns

IElement

the first IElement object found in the current search context by the given tag name or `null` if no elements were found.

## Exceptions

ArgumentException

The `tagName` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Method GetElementsByClassName

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementsByClassName(string)

Finds all IElement objects in the current search context by the given `className`.

```
IEnumerable<IElement> GetElementsByClassName(string className)
```

## Parameters

`className` string

The `class` attribute of the HTML element.

## Returns

IEnumerable<IElement>

A collection of the IElement objects found in the current search context by the given `className` or an empty collection if no elements were found.

## Exceptions

ArgumentException

The `className` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Method GetElementsByCssSelector

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementsByCssSelector(string)

Finds all IElement objects in the current search context by the given `cssSelector`.

```
IEnumerable<IElement> GetElementsByCssSelector(string cssSelector)
```

## Parameters

`cssSelector` string

The CSS selector.

## Returns

IEnumerable<IElement>

A collection of the IElement objects found in the current search context by the given `cssSelector` or an empty collection if no elements were found.

## Exceptions

ArgumentException

The `cssSelector` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Method GetElementsByName

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementsByName(string)

Finds all IElement objects in the current search context by the given `name`.

```
IEnumerable<IElement> GetElementsByName(string name)
```

## Parameters

`name` string

The `name` attribute of the HTML element.

## Returns

IEnumerable<IElement>

a collection of the IElement objects found in the current search context by the given `name` or an empty collection if no elements were found.

## Exceptions

ArgumentException

The `name` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Method GetElementsByTagName

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## GetElementsByTagName(string)

Finds all IElement objects in the current search context by the given tag name.

```
IEnumerable<IElement> GetElementsByTagName(string tagName)
```

### Parameters

`tagName` string

The tag name of the HTML element.

### Returns

IEnumerable<IElement>

The collection of the IElement objects found in the current search context by the given tag name or an empty collection if no elements were found.

### Exceptions

ArgumentException

The `tagName` is null, empty or contains only blank characters.

ObjectDisposedException

The ISearchContext has already been disposed.

# Interface ISelectElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents DOM HTML `<select>` element.

```
public interface ISelectElement : IFormControlElement, IElement, INode, IEventTarget,
ISearchContext, IAutoDisposable
```

**Inherited Members**

IFormControlElement.Form , IFormControlElement.IsEnabled , IFormControlElement.Value , IElement.AttributeNodes , IElement.Attributes , IElement.BoundingClientRect , IElement.InnerHtml , IElement.InnerText , IElement.OuterHtml , IElement.Blur() , IElement.Focus() , IElement.ScrollIntoView(AlignTo) , INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Multiple

Enables or disables selecting multiple options in the list.

Options

Gets the collection of `<option>` elements contained by this `<select>` element.

# Property Multiple

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Multiple

Enables or disables selecting multiple options in the list.

```
bool Multiple { get; set; }
```

## Property Value

bool

`true` if multiple options can be selected, `false` if only one option can be selected at a time.

## Exceptions

ObjectDisposedException

The ISelectElement has already been disposed.

# Property Options

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Options

Gets the collection of `<option>` elements contained by this `<select>` element.

```
IEnumerable<IOptionElement> Options { get; }
```

## Property Value

IEnumerable<IOptionElement>

The collection of `<option>` elements.

## Exceptions

ObjectDisposedException

The ISelectElement has already been disposed.

# Interface ITextAreaElement

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents DOM HTML `<textarea>` element.

```
public interface ITextAreaElement : IFormControlElement, IElement, INode, IEventTarget,
ISearchContext, IAutoDisposable
```

**Inherited Members**

IFormControlElement.Form , IFormControlElement.IsEnabled , IFormControlElement.Value , IElement.AttributeNodes , IElement.Attributes , IElement.BoundingClientRect , IElement.InnerHtml , IElement.InnerText , IElement.OuterHtml , IElement.Blur() , IElement.Focus() , IElement.ScrollIntoView(AlignTo) , INode.Children , INode.Document , INode.Frame , INode.NextSibling , INode.NodeName , INode.NodeValue , INode.Parent , INode.PreviousSibling , INode.TextContent , INode.Type , INode.XPath , INode.Click() , INode.CompareDocumentPosition(INode) , INode.Evaluate(string, XPathResultType) , IEventTarget.Events , IEventTarget.DispatchEvent(IEvent) , ISearchContext.GetElementByClassName(string) , ISearchContext.GetElementByCssSelector(string) , ISearchContext.GetElementById(string) , ISearchContext.GetElementByName(string) , ISearchContext.GetElementByTagName(string) , ISearchContext.GetElementsByClassName(string) , ISearchContext.GetElementsByCssSelector(string) , ISearchContext.GetElementsByName(string) , ISearchContext.GetElementsByTagName(string) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

# Enum NodeType

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Represents the node types that can be used to distinguish different kind of nodes (such as an HTML element, text, attribute) from each other.

```
public enum NodeType
```

## Fields

`AttributeNode = 2`

An attribute of the HTML element.

`CDataSectionNode = 4`

A `[CDATA]` section.

`CommentNode = 8`

The content between between the `<!--` and `-->` statements.

`DocumentFragmentNode = 11`

A segment of the document structure.

`DocumentNode = 9`

The document node.

`DocumentTypeNode = 10`

The document type node. For example, `<!DOCTYPE html>` for HTML5 documents.

`ElementNode = 1`

An element node, such as `<p>` or `<div>`.

`EntityNode = 6`

An XML `<!ENTITY ...>` node.

`EntityReferenceNode = 5`

An XML entity reference node.

`NotationNode = 12`

An XML `<!NOTATION ...>` node.

`ProcessingInstructionsNode = 7`

A `ProcessingInstruction` of an XML document such as `<?xml-stylesheet ... ?>` declaration.

`ShadowRootNode = 14`

A shadow root node.

`TextNode = 3`

The actual text of the HTML element or the attribute.

`Unknown = 0`

Unknown node type.

`XPathNamespaceNode = 13`

An XPath namespace node.

# Class PointInspection

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

Provides information about a DOM node at the specified point inside the loaded document.

```
public sealed class PointInspection
```

**Inheritance**

object ← PointInspection

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AbsoluteImageUrl

Gets the absolute URL of the image located at the point.

AbsoluteLinkUrl

Gets the absolute URL of the link DOM element at the specified point. Returns an empty string if there is no link at the point.

LocalPoint

Gets the coordinates of the point relative to the node.

Node

Gets the node at the specified point.

UrlNode

Gets the link node when a link is located at the point.

# Property AbsoluteImageUrl

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## AbsoluteImageUrl

Gets the absolute URL of the image located at the point.

```
public string AbsoluteImageUrl { get; }
```

## Property Value

string

The absolute URL of the image located at the point or an empty string if there is no image at the point.

# Property AbsoluteLinkUrl

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## AbsoluteLinkUrl

Gets the absolute URL of the link DOM element at the specified point. Returns an empty string if there is no link at the point.

```
public string AbsoluteLinkUrl { get; }
```

## Property Value

string

The absolute URL of the link DOM element at the specified point or an empty string if there is no link at the point.

# Property LocalPoint

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## LocalPoint

Gets the coordinates of the point relative to the node.

```
public Point LocalPoint { get; }
```

## Property Value

Point

The coordinates of the point relative to the node.

# Property Node

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## Node

Gets the node at the specified point.

```
public INode Node { get; }
```

## Property Value

INode

The node at the specified point or null if there is no node at the point.

# Property UrlNode

Namespace: DotNetBrowser.Dom

Assembly: DotNetBrowser.dll

## UrlNode

Gets the link node when a link is located at the point.

```
public INode UrlNode { get; }
```

## Property Value

INode

The link node when a link is located at the point or null if there is no link node at the point.

# Namespace DotNetBrowser.Dom.Events

## Classes

DomEventArgs
: The common DOM event arguments.

Event
: Represents a DOM event that can be handled on the .NET side.

EventParameters
: The parameters of creating the DOM event.

EventParameters.Builder
: The EventParameters builder.

EventType
: The DOM event type. Wraps a string that specifies the name of the event without the "on" prefix.

KeyEventParameters
: Represents the DOM keyboard event parameters.

MouseEventParameters
: Represents the DOM mouse event parameters.

UiEventModifierParameters
: Represents the DOM UI event parameters with the key modifiers.

## Interfaces

IEvent
: Represents DOM `Event` object and provides access to the event object data.

IEventTarget
: This interface is implemented by all INode implementations to support DOM event model.

IEvents
: A collection of custom DOM events that can be listened and/or handled.

IKeyEvent
: Represents a keyboard event and provides an access to specific keyboard event data.

  This event occurs when a user performs some keyboard action, for example presses or releases a keyboard key.

IMouseEvent

Represents a mouse event and provides an access to specific mouse event data.

This event occurs when a user performs an action with a pointing device (such as a mouse), for example moves the mouse or clicks a mouse button.

ITouchEvent

Represents a touch event and provides an access to specific touch event data.

This event occurs when a user performs an action with a touch device (such as a touchscreen).

IUiKeyEventModifier

Represents a DOM UI event that can be fired with the key modifiers.

The KeyModifiers provides access to the data about pressed service keys, such as `Ctrl`, `Alt`, `Shift`, `Meta`, when a keyboard or mouse event occurs.

IWheelEvent

A wheel event that provides access to the wheel event data.

This event occurs when a user rotates a wheel on a pointing device (such as a mouse).

## Enums

DeltaMode

The type of the delta units.

DomKeyCode

The DOM key codes represent physical keys on the keyboard (as opposed to the character generated by pressing the key). In other words, these values are not altered by keyboard layout or the state of the modifier keys.

DomMouseButton

A DOM mouse button.

EventPhase

Represents the DOM event phases.

# Enum DeltaMode

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The type of the delta units.

```
public enum DeltaMode
```

## Fields

`Line = 1`

The delta units are specified in lines.

`Page = 2`

The delta units are specified in pages.

`Pixel = 0`

The delta units are specified in pixels.

# Class DomEventArgs

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The common DOM event arguments.

```
public sealed class DomEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← DomEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Event

Gets the DOM event that is currently handled.

# Property Event

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Event

Gets the DOM event that is currently handled.

```
public IEvent Event { get; }
```

## Property Value

IEvent

The DOM event that is currently handled.

# Enum DomKeyCode

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The DOM key codes represent physical keys on the keyboard (as opposed to the character generated by pressing the key). In other words, these values are not altered by keyboard layout or the state of the modifier keys.

```
public enum DomKeyCode
```

## Fields

`Abort = 151`

The "ABORT" key code.

`Again = 131`

The "AGAIN" key code.

`AltLeft = 166`

The "ALT_LEFT" key code.

`AltRight = 170`

The "ALT_RIGHT" key code.

`ArrowDown = 93`

The "ARROW_DOWN" key code.

`ArrowLeft = 92`

The "ARROW_LEFT" key code.

`ArrowRight = 91`

The "ARROW_RIGHT" key code.

`ArrowUp = 94`

The "ARROW_UP" key code.

`Backquote = 65`

The "BACKQUOTE" key code.

`Backslash = 62`

The "BACKSLASH" key code.

`Backspace = 55`

The "BACKSPACE" key code.

`BassBoost = 199`

The "BASS_BOOST" key code.

`BracketLeft = 60`

The "BRACKET_LEFT" key code.

`BracketRight = 61`

The "BRACKET_RIGHT" key code.

`BrightnessAuto = 179`

The "BRIGHTNESS_AUTO" key code.

`BrightnessDown = 175`

The "BRIGHTNESS_DOWN" key code.

`BrightnessMaximum = 178`

The "BRIGHTNESS_MAXIMUM" key code.

`BrightnessMinimium = 177`

The "BRIGHTNESS_MINIMIUM" key code.

`BrightnessToggle = 176`

The "BRIGHTNESS_TOGGLE" key code.

`BrightnessUp = 174`

The "BRIGHTNESS_UP" key code.

`BrowserBack = 225`

The "BROWSER_BACK" key code.

`BrowserFavorites = 229`

The "BROWSER_FAVORITES" key code.

`BrowserForward = 226`

The "BROWSER_FORWARD" key code.

`BrowserHome = 224`

The "BROWSER_HOME" key code.

`BrowserRefresh = 228`

The "BROWSER_REFRESH" key code.

`BrowserSearch = 223`

The "BROWSER_SEARCH" key code.

`BrowserStop = 227`

The "BROWSER_STOP" key code.

`CapsLock = 69`

The "CAPS_LOCK" key code.

`ChannelDown = 187`

The "CHANNEL_DOWN" key code.

`ChannelUp = 186`

The "CHANNEL_UP" key code.

`Close = 220`

The "CLOSE" key code.

`ClosedCaptionToggle = 173`

The "CLOSED_CAPTION_TOGGLE" key code.

`Comma = 66`

The "COMMA" key code.

`ContextMenu = 113`

The "CONTEXT_MENU" key code.

`ControlLeft = 164`

The "CONTROL_LEFT" key code.

`ControlRight = 168`

The "CONTROL_RIGHT" key code.

`Convert = 144`

The "CONVERT" key code.

`Copy = 134`

The "COPY" key code.

`Cut = 133`

The "CUT" key code.

`Del = 88`

The "DEL" key code.

`Digit0 = 52`

The "DIGIT0" key code.

`Digit1 = 43`

The "DIGIT1" key code.

`Digit2 = 44`

The "DIGIT2" key code.

`Digit3 = 45`

The "DIGIT3" key code.

`Digit4 = 46`

The "DIGIT4" key code.

`Digit5 = 47`

The "DIGIT5" key code.

`Digit6 = 48`

The "DIGIT6" key code.

`Digit7 = 49`

The "DIGIT7" key code.

`Digit8 = 50`

The "DIGIT8" key code.

`Digit9 = 51`

The "DIGIT9" key code.

`DisplayToggleIntExt = 12`

The "DISPLAY_TOGGLE_INT_EXT" key code.

`Eject = 196`

The "EJECT" key code.

`End = 89`

The "END" key code.

`Enter = 53`

The "ENTER" key code.

`Equal = 59`

The "EQUAL" key code.

`Escape = 54`

The "ESCAPE" key code.

`Exit = 185`

The "EXIT" key code.

`F1 = 70`

The "F1" key code.

`F10 = 79`

The "F10" key code.

`F11 = 80`

The "F11" key code.

`F12 = 81`

The "F12" key code.

`F13 = 116`

The "F13" key code.

`F14 = 117`

The "F14" key code.

`F15 = 118`

The "F15" key code.

`F16 = 119`

The "F16" key code.

`F17 = 120`

The "F17" key code.

`F18 = 121`

The "F18" key code.

`F19 = 122`

The "F19" key code.

`F2 = 71`

The "F2" key code.

`F20 = 123`

The "F20" key code.

`F21 = 124`

The "F21" key code.

`F22 = 125`

The "F22" key code.

`F23 = 126`

The "F23" key code.

`F24 = 127`

The "F24" key code.

`F3 = 72`

The "F3" key code.

`F4 = 73`

The "F4" key code.

`F5 = 74`

The "F5" key code.

`F6 = 75`

The "F6" key code.

`F7 = 76`

The "F7" key code.

`F8 = 77`

The "F8" key code.

`F9 = 78`

The "F9" key code.

`Find = 136`

The "FIND" key code.

`Fn = 4`

The "FN" key code.

`FnLock = 5`

The "FN_LOCK" key code.

`Help = 129`

The "HELP" key code.

`Home = 86`

The "HOME" key code.

`Hyper = 2`

The "HYPER" key code.

`Info = 172`

The "INFO" key code.

`Insert = 85`

The "INSERT" key code.

`IntlBackslash = 112`

The "INTL_BACKSLASH" key code.

`IntlRo = 141`

The "INTL_RO" key code.

`IntlYen = 143`

The "INTL_YEN" key code.

`KanaMode = 142`

The "KANA_MODE" key code.

`KbdIllumDown = 181`

The "KBD_ILLUM_DOWN" key code.

`KbdIllumUp = 180`

The "KBD_ILLUM_UP" key code.

`KeyboardLayoutSelect = 237`

The "KEYBOARD_LAYOUT_SELECT" key code.

`Lang1 = 146`

The "LANG1" key code.

`Lang2 = 147`

The "LANG2" key code.

`Lang3 = 148`

The "LANG3" key code.

`Lang4 = 149`

The "LANG4" key code.

`Lang5 = 150`

The "LANG5" key code.

`LaunchApp1 = 207`

The "LAUNCH_APP1" key code.

`LaunchApp2 = 206`

The "LAUNCH_APP2" key code.

`LaunchAssistant = 217`

The "LAUNCH_ASSISTANT" key code.

`LaunchAudioBrowser = 218`

The "LAUNCH_AUDIO_BROWSER" key code.

`LaunchCalendar = 205`

The "LAUNCH_CALENDAR" key code.

`LaunchContacts = 204`

The "LAUNCH_CONTACTS" key code.

`LaunchControlPanel = 211`

The "LAUNCH_CONTROL_PANEL" key code.

`LaunchDocuments = 213`

The "LAUNCH_DOCUMENTS" key code.

`LaunchInternetBrowser = 208`

The "LAUNCH_INTERNET_BROWSER" key code.

`LaunchKeyboardLayout = 215`

The "LAUNCH_KEYBOARD_LAYOUT" key code.

`LaunchMail = 203`

The "LAUNCH_MAIL" key code.

`LaunchPhone = 183`

The "LAUNCH_PHONE" key code.

`LaunchScreenSaver = 216`

The "LAUNCH_SCREEN_SAVER" key code.

`LaunchSpreadsheet = 202`

The "LAUNCH_SPREADSHEET" key code.

`LaunchWordProcessor = 201`

The "LAUNCH_WORD_PROCESSOR" key code.

`LockScreen = 210`

The "LOCK_SCREEN" key code.

`LogOff = 209`

The "LOG_OFF" key code.

`MailForward = 235`

The "MAIL_FORWARD" key code.

`MailReply = 234`

The "MAIL_REPLY" key code.

`MailSend = 236`

The "MAIL_SEND" key code.

`MediaFastForward = 191`

The "MEDIA_FAST_FORWARD" key code.

`MediaLast = 182`

The "MEDIA_LAST" key code.

`MediaPause = 189`

The "MEDIA_PAUSE" key code.

`MediaPlay = 188`

The "MEDIA_PLAY" key code.

`MediaPlayPause = 197`

The "MEDIA_PLAY_PAUSE" key code.

`MediaRecord = 190`

The "MEDIA_RECORD" key code.

`MediaRewind = 192`

The "MEDIA_REWIND" key code.

`MediaSelect = 200`

The "MEDIA_SELECT" key code.

`MediaStop = 195`

The "MEDIA_STOP" key code.

`MediaTrackNext = 193`

The "MEDIA_TRACK_NEXT" key code.

`MediaTrackPrevious = 194`

The "MEDIA_TRACK_PREVIOUS" key code.

`MetaLeft = 167`

The "META_LEFT" key code.

`MetaRight = 171`

The "META_RIGHT" key code.

`Minus = 58`

The "MINUS" key code.

`New = 219`

The "NEW" key code.

`NonConvert = 145`

The "NON_CONVERT" key code.

`None = 1`

No key code.

`NumLock = 95`

The "NUM_LOCK" key code.

`Numpad0 = 110`

The "NUMPAD0" key code.

`Numpad1 = 101`

The "NUMPAD1" key code.

`Numpad2 = 102`

The "NUMPAD2" key code.

`Numpad3 = 103`

The "NUMPAD3" key code.

`Numpad4 = 104`

The "NUMPAD4" key code.

`Numpad5 = 105`

The "NUMPAD5" key code.

`Numpad6 = 106`

The "NUMPAD6" key code.

`Numpad7 = 107`

The "NUMPAD7" key code.

`Numpad8 = 108`

The "NUMPAD8" key code.

`Numpad9 = 109`

The "NUMPAD9" key code.

`NumpadAdd = 99`

The "NUMPAD_ADD" key code.

`NumpadBackspace = 155`

The "NUMPAD_BACKSPACE" key code.

`NumpadClear = 162`

The "NUMPAD_CLEAR" key code.

`NumpadClearEntry = 163`

The "NUMPAD_CLEAR_ENTRY" key code.

`NumpadComma = 140`

The "NUMPAD_COMMA" key code.

`NumpadDecimal = 111`

The "NUMPAD_DECIMAL" key code.

`NumpadDivide = 96`

The "NUMPAD_DIVIDE" key code.

`NumpadEnter = 100`

The "NUMPAD_ENTER" key code.

`NumpadEqual = 115`

The "NUMPAD_EQUAL" key code.

`NumpadMemoryAdd = 159`

The "NUMPAD_MEMORY_ADD" key code.

`NumpadMemoryClear = 158`

The "NUMPAD_MEMORY_CLEAR" key code.

`NumpadMemoryRecall = 157`

The "NUMPAD_MEMORY_RECALL" key code.

`NumpadMemoryStore = 156`

The "NUMPAD_MEMORY_STORE" key code.

`NumpadMemorySubtract = 160`

The "NUMPAD_MEMORY_SUBTRACT" key code.

`NumpadMultiply = 97`

The "NUMPAD_MULTIPLY" key code.

`NumpadParenLeft = 153`

The "NUMPAD_PAREN_LEFT" key code.

`NumpadParenRight = 154`

The "NUMPAD_PAREN_RIGHT" key code.

`NumpadSignChange = 161`

The "NUMPAD_SIGN_CHANGE" key code.

`NumpadSubtract = 98`

The "NUMPAD_SUBTRACT" key code.

`Open = 128`

The "OPEN" key code.

`PageDown = 90`

The "PAGE_DOWN" key code.

`PageUp = 87`

The "PAGE_UP" key code.

`Paste = 135`

The "PASTE" key code.

`Pause = 84`

The "PAUSE" key code.

`Period = 67`

The "PERIOD" key code.

`Power = 114`

The "POWER" key code.

`Print = 222`

The "PRINT" key code.

`PrintScreen = 82`

The "PRINT_SCREEN" key code.

`PrivacyScreenToggle = 9`

The "PRIVACY_SCREEN_TOGGLE" key code.

`ProgramGuide = 184`

The "PROGRAM_GUIDE" key code.

`Props = 152`

The "PROPS" key code.

`Quote = 64`

The "QUOTE" key code.

`Redo = 233`

The "REDO" key code.

`Resume = 7`

The "RESUME" key code.

`Save = 221`

The "SAVE" key code.

`ScrollLock = 83`

The "SCROLL_LOCK" key code.

`Select = 130`

The "SELECT" key code.

`SelectTask = 212`

The "SELECT_TASK" key code.

`Semicolon = 63`

The "SEMICOLON" key code.

`ShiftLeft = 165`

The "SHIFT_LEFT" key code.

`ShiftRight = 169`

The "SHIFT_RIGHT" key code.

`ShowAllWindows = 238`

The "SHOW_ALL_WINDOWS" key code.

`Slash = 68`

The "SLASH" key code.

`Sleep = 10`

The "SLEEP" key code.

`Space = 57`

The "SPACE" key code.

`SpeechInputToggle = 198`

The "SPEECH_INPUT_TOGGLE" key code.

`SpellCheck = 214`

The "SPELL_CHECK" key code.

`Super = 3`

The "SUPER" key code.

`Suspend = 6`

The "SUSPEND" key code.

`Tab = 56`

The "TAB" key code.

`Turbo = 8`

The "TURBO" key code.

`Undo = 132`

The "UNDO" key code.

`Unknown = 0`

The key code is unidentified.

`UsA = 17`

The "US_A" key code.

`UsB = 18`

The "US_B" key code.

`UsC = 19`

The "US_C" key code.

`UsD = 20`

The "US_D" key code.

`UsE = 21`

The "US_E" key code.

`UsF = 22`

The "US_F" key code.

`UsG = 23`

The "US_G" key code.

`UsH = 24`

The "US_H" key code.

`UsI = 25`

The "US_I" key code.

`UsJ = 26`

The "US_J" key code.

`UsK = 27`

The "US_K" key code.

`UsL = 28`

The "US_L" key code.

`UsM = 29`

The "US_M" key code.

`UsN = 30`

The "US_N" key code.

`UsO = 31`

The "US_O" key code.

`UsP = 32`

The "US_P" key code.

`UsQ = 33`

The "US_Q" key code.

`UsR = 34`

The "US_R" key code.

`UsS = 35`

The "US_S" key code.

`UsT = 36`

The "US_T" key code.

`UsU = 37`

The "US_U" key code.

`UsV = 38`

The "US_V" key code.

`UsW = 39`

The "US_W" key code.

`UsX = 40`

The "US_X" key code.

`UsY = 41`

The "US_Y" key code.

`UsZ = 42`

The "US_Z" key code.

`UsbErrorRollOver = 14`

The "USB_ERROR_ROLL_OVER" key code.

`UsbErrorUndefined = 16`

The "USB_ERROR_UNDEFINED" key code.

`UsbPostFail = 15`

The "USB_POST_FAIL" key code.

`UsbReserved = 13`

The "USB_RESERVED" key code.

`VolumeDown = 139`

The "VOLUME_DOWN" key code.

`VolumeMute = 137`

The "VOLUME_MUTE" key code.

`VolumeUp = 138`

The "VOLUME_UP" key code.

`WakeUp = 11`

The "WAKE_UP" key code.

`ZoomIn = 230`

The "ZOOM_IN" key code.

`ZoomOut = 231`

The "ZOOM_OUT" key code.

`ZoomToggle = 232`

The "ZOOM_TOGGLE" key code.

# Enum DomMouseButton

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

A DOM mouse button.

```
public enum DomMouseButton
```

## Fields

`Back = 3`

A fourth button. Typically the *Browser Back* button.

`Forward = 4`

A fifth button. Typically the *Browser Forward* button.

`Main = 0`

A main mouse button. Usually the left button or the un-initialized state.

`Middle = 1`

An auxiliary button. Usually the wheel button or the middle button (if present).

`Right = 2`

A secondary button. Usually the right button.

## See Also

IMouseEvent

# Class Event

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents a DOM event that can be handled on the .NET side.

```
public sealed class Event
```

**Inheritance**

object ← Event

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

EventType

Gets the DOM event type.

UseCapture

Indicates if capturing is used for the DOM event.

## Events

EventReceived

Occurs when the corresponding DOM event is received.

## Operators

operator +(Event, EventHandler<DomEventArgs>)

The overloaded operator that can be used for simplifying registering event handlers for the DOM event.

operator -(Event, EventHandler<DomEventArgs>)

The overloaded operator that can be used for simplifying registering event handlers for the DOM event.

# Property EventType

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## EventType

Gets the DOM event type.

```
public EventType EventType { get; }
```

## Property Value

EventType

The DOM event type.

# Property UseCapture

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## UseCapture

Indicates if capturing is used for the DOM event.

```
public bool UseCapture { get; }
```

## Property Value

bool

`true` if capturing is used, `false` otherwise.

## Remarks

If capturing is used, all events of the specified type will be dispatched to the registered listeners before being dispatched to any event target beneath them in the tree.

Events which are bubbling upward through the tree will not trigger the listeners in this case.

# Event EventReceived

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the corresponding DOM event is received.

```
public event EventHandler<DomEventArgs> EventReceived
```

## Returns

EventHandler<DomEventArgs>

Occurs when the corresponding DOM event is received.

# Operator operator +

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## operator +(Event, EventHandler<DomEventArgs>)

The overloaded operator that can be used for simplifying registering event handlers for the DOM event.

```
public static Event operator +(Event evt, EventHandler<DomEventArgs> handler)
```

### Parameters

`evt` Event

The custom DOM event.

`handler` EventHandler<DomEventArgs>

the event handler to register for this DOM event.

### Returns

Event

The updated Event instance.

### Exceptions

ObjectDisposedException

The IEventTarget has already been disposed.

# Operator operator -

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## operator -(Event, EventHandler<DomEventArgs>)

The overloaded operator that can be used for simplifying registering event handlers for the DOM event.

```
public static Event operator -(Event node, EventHandler<DomEventArgs> handler)
```

### Parameters

`node` Event

The custom DOM event.

`handler` EventHandler<DomEventArgs>

The event handler to unregister from this DOM event.

### Returns

Event

The updated Event instance.

### Exceptions

ObjectDisposedException

The IEventTarget has already been disposed.

# Class EventParameters

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The parameters of creating the DOM event.

```
public class EventParameters
```

**Inheritance**

object ← EventParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Bubbles

Indicates whether the dispatched event is a bubbling event.

Cancelable

Indicates whether the dispatched event can have its default action prevented.

# Property Bubbles

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Bubbles

Indicates whether the dispatched event is a bubbling event.

```
public bool Bubbles { get; }
```

## Property Value

bool

`true` if the dispatched event is a bubbling event, `false` otherwise.

# Property Cancelable

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Cancelable

Indicates whether the dispatched event can have its default action prevented.

```
public bool Cancelable { get; }
```

## Property Value

bool

`true` if the dispatched event can have its default action prevented, `false` otherwise.

# Class EventParameters.Builder

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The EventParameters builder.

```
public class EventParameters.Builder
```

**Inheritance**

object ← EventParameters.Builder

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Bubbles

Gets or sets whether the dispatched event is a bubbling event.

Cancelable

Gets or sets whether the dispatched event can have its default action prevented.

## Methods

Build()

Builds and returns a read-only EventParameters object.

# Property Bubbles

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Bubbles

Gets or sets whether the dispatched event is a bubbling event.

```
public bool Bubbles { get; set; }
```

## Property Value

bool

`true` if the dispatched event is a bubbling event, `false` otherwise.

# Property Cancelable

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Cancelable

Gets or sets whether the dispatched event can have its default action prevented.

```
public bool Cancelable { get; set; }
```

## Property Value

bool

`true` if the dispatched event can have its default action prevented, `false` otherwise.

# Method Build

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Build()

Builds and returns a read-only EventParameters object.

```
public EventParameters Build()
```

## Returns

EventParameters

a read-only EventParameters object.

# Enum EventPhase

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents the DOM event phases.

```
public enum EventPhase
```

## Fields

`AtTarget = 2`

The event is being evaluated at the event target.

`BubblingPhase = 3`

The event is moving from the event target up to the root.

`CapturingPhase = 1`

The event is moving from the document root down the event target.

`Unspecified = 0`

The event phase is not specified.

# Class EventType

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The DOM event type. Wraps a string that specifies the name of the event without the "on" prefix.

```
public class EventType
```

**Inheritance**

object ← EventType

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

EventType(string)

Constructs an EventType instance from the given string.

## Fields

Abort

The "abort" event type.

Blur

The "blur" event type.

Change

The "change" event type.

Click

The "click" event type.

DoubleClick

The "dblclick" event type.

Error
The "error" event type.

Focus
The "focus" event type.

KeyDown
The "keydown" event type.

KeyPress
The "keypress" event type.

KeyUp
The "keyup" event type.

Load
The "load" event type.

MouseDown
The "mousedown" event type.

MouseMove
The "mousemove" event type.

MouseOut
The "mouseout" event type.

MouseOver
The "mouseover" event type.

MouseUp
The "mouseup" event type.

Reset
The "reset" event type.

Resize
The "resize" event type.

Scroll
The "scroll" event type.

Select
The "select" event type.

Submit
The "submit" event type.

TouchCancel
The "touchcancel" event type.

TouchEnd
The "touchend" event type.

TouchMove
The "touchmove" event type.

TouchStart
The "touchstart" event type.

Unload
The "unload" event type.

Wheel
The "wheel" event type.

## Properties

Value
The DOM event type as string value.

## Methods

Equals(object)

GetHashCode()

## Operators

operator ==(EventType, EventType)
The overloaded operator that simplifies comparing the EventType objects.

operator !=(EventType, EventType)
The overloaded operator that simplifies comparing the EventType objects.

# Constructor EventType

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## EventType(string)

Constructs an EventType instance from the given string.

```
public EventType(string value)
```

## Parameters

`value` string

A string that specifies the name of the event without the "on" prefix.

# Field Abort

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "abort" event type.

```
public static readonly EventType Abort
```

## Returns

EventType

The "abort" event type.

# Field Blur

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "blur" event type.

```
public static readonly EventType Blur
```

## Returns

EventType

The "blur" event type.

# Field Change

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "change" event type.

```
public static readonly EventType Change
```

## Returns

EventType

The "change" event type.

# Field Click

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "click" event type.

```
public static readonly EventType Click
```

## Returns

EventType

The "click" event type.

# Field DoubleClick

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "dblclick" event type.

```
public static readonly EventType DoubleClick
```

## Returns

EventType

The "dblclick" event type.

# Field Error

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "error" event type.

```
public static readonly EventType Error
```

## Returns

EventType

The "error" event type.

# Field Focus

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "focus" event type.

```
public static readonly EventType Focus
```

## Returns

EventType

The "focus" event type.

# Field KeyDown

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "keydown" event type.

```
public static readonly EventType KeyDown
```

## Returns

EventType

The "keydown" event type.

# Field KeyPress

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "keypress" event type.

```
public static readonly EventType KeyPress
```

## Returns

EventType

The "keypress" event type.

# Field KeyUp

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "keyup" event type.

```
public static readonly EventType KeyUp
```

## Returns

EventType

The "keyup" event type.

# Field Load

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "load" event type.

```
public static readonly EventType Load
```

## Returns

EventType

The "load" event type.

# Field MouseDown

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "mousedown" event type.

```
public static readonly EventType MouseDown
```

## Returns

EventType

The "mousedown" event type.

# Field MouseMove

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "mousemove" event type.

```
public static readonly EventType MouseMove
```

## Returns

EventType

The "mousemove" event type.

# Field MouseOut

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "mouseout" event type.

```
public static readonly EventType MouseOut
```

## Returns

EventType

The "mouseout" event type.

# Field MouseOver

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "mouseover" event type.

```
public static readonly EventType MouseOver
```

## Returns

EventType

The "mouseover" event type.

# Field MouseUp

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "mouseup" event type.

```
public static readonly EventType MouseUp
```

## Returns

EventType

The "mouseup" event type.

# Field Reset

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "reset" event type.

```
public static readonly EventType Reset
```

## Returns

EventType

The "reset" event type.

# Field Resize

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "resize" event type.

```
public static readonly EventType Resize
```

## Returns

EventType

The "resize" event type.

# Field Scroll

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "scroll" event type.

```
public static readonly EventType Scroll
```

## Returns

EventType

The "scroll" event type.

# Field Select

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "select" event type.

```
public static readonly EventType Select
```

## Returns

EventType

The "select" event type.

# Field Submit

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "submit" event type.

```
public static readonly EventType Submit
```

## Returns

EventType

The "submit" event type.

# Field TouchCancel

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "touchcancel" event type.

```
public static readonly EventType TouchCancel
```

## Returns

EventType

The "touchcancel" event type.

# Field TouchEnd

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "touchend" event type.

```
public static readonly EventType TouchEnd
```

## Returns

EventType

The "touchend" event type.

# Field TouchMove

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "touchmove" event type.

```
public static readonly EventType TouchMove
```

## Returns

EventType

The "touchmove" event type.

# Field TouchStart

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "touchstart" event type.

```
public static readonly EventType TouchStart
```

## Returns

EventType

The "touchstart" event type.

# Field Unload

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "unload" event type.

```
public static readonly EventType Unload
```

## Returns

EventType

The "unload" event type.

# Field Wheel

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

The "wheel" event type.

```
public static readonly EventType Wheel
```

## Returns

EventType

The "wheel" event type.

# Property Value

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Value

The DOM event type as string value.

```
public string Value { get; }
```

## Property Value

string

# Method Equals

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Operator operator !=

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## operator !=(EventType, EventType)

The overloaded operator that simplifies comparing the EventType objects.

```
public static bool operator !=(EventType eventType1, EventType eventType2)
```

## Parameters

`eventType1` EventType

The EventType to compare.

`eventType2` EventType

The EventType to compare.

## Returns

bool

true if the objects are considered to be not equal.

# Operator operator ==

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## operator ==(EventType, EventType)

The overloaded operator that simplifies comparing the EventType objects.

```
public static bool operator ==(EventType eventType1, EventType eventType2)
```

### Parameters

`eventType1` EventType

The EventType to compare.

`eventType2` EventType

The EventType to compare.

### Returns

bool

true if the objects are considered to be equal.

# Interface IEvent

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents DOM `Event` object and provides access to the event object data.

```
public interface IEvent : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Bubbles

Indicates whether an event is a bubbling event.

Cancelable

Indicates whether an event can have its default action prevented.

CurrentTarget

Gets the IEventTarget whose event listeners are currently being processed.

EventPhase

Gets the phase of event flow.

Target

Gets the IEventTarget to which the event was originally dispatched.

Type

Gets the type of this DOM event.

## Methods

PreventDefault()

Cancels the event if it is cancelable, without stopping further propagation of the event.

StopPropagation()

Prevents further propagation of the current event.

# Property Bubbles

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Bubbles

Indicates whether an event is a bubbling event.

```
bool Bubbles { get; }
```

## Property Value

bool

If the event can bubble the property value is `true`, in other case the value is `false`.

# Property Cancelable

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Cancelable

Indicates whether an event can have its default action prevented.

```
bool Cancelable { get; }
```

## Property Value

bool

If the default action can be prevented, the property value is `true`, in other case the value is `false`.

# Property CurrentTarget

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## CurrentTarget

Gets the IEventTarget whose event listeners are currently being processed.

```
IEventTarget CurrentTarget { get; }
```

## Property Value

IEventTarget

The IEventTarget whose event listeners are currently being processed.

# Property EventPhase

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## EventPhase

Gets the phase of event flow.

```
EventPhase EventPhase { get; }
```

## Property Value

EventPhase

The phase of event flow that is currently being evaluated.

# Property Target

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Target

Gets the IEventTarget to which the event was originally dispatched.

```
IEventTarget Target { get; }
```

## Property Value

IEventTarget

The IEventTarget to which the event was originally dispatched.

# Property Type

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Type

Gets the type of this DOM event.

```
EventType Type { get; }
```

## Property Value

EventType

The type of this DOM event.

# Method PreventDefault

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## PreventDefault()

Cancels the event if it is cancelable, without stopping further propagation of the event.

```
void PreventDefault()
```

# Method StopPropagation

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## StopPropagation()

Prevents further propagation of the current event.

```
void StopPropagation()
```

# Interface IEventTarget

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

This interface is implemented by all INode implementations to support DOM event model.

```
public interface IEventTarget : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Events

Gets the DOM events that can be listened.

## Methods

DispatchEvent(IEvent)

Dispatches (sends) a particular DOM event to the current target.

# Property Events

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Events

Gets the DOM events that can be listened.

```
IEvents Events { get; }
```

## Property Value

IEvents

The DOM events that can be listened.

# Method DispatchEvent

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## DispatchEvent(IEvent)

Dispatches (sends) a particular DOM event to the current target.

```
bool DispatchEvent(IEvent evt)
```

## Parameters

`evt` IEvent

The DOM event to dispatch

## Returns

bool

`false` if event is cancelable and at least one of the event handlers which handled this event called cancelable action; `true` otherwise.

## Exceptions

ObjectDisposedException

The IEventTarget has already been disposed.

# Interface IEvents

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

A collection of custom DOM events that can be listened and/or handled.

```
public interface IEvents
```

## Properties

this[EventType, bool]
: Gets or sets the custom DOM event to listen.

## Events

Abort
: Occurs when the loading of a resource has been aborted.

Blur
: Occurs when an element loses focus.

Change
: Occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>).

Click
: Occurs when the user clicks on an element.

DoubleClick
: Occurs when the user double-clicks on an element.

Error
: Occurs when an error occurs while loading an external file.

Focus
: Occurs when an element gets focus.

KeyDown
Occurs when the user presses a key.

KeyPress
Occurs when the user presses a key that produces a character value.

KeyUp
Occurs when the user releases a key.

Load
Occurs when an object has loaded.

MouseDown
Occurs when the user presses a mouse button over an element.

MouseMove
Occurs when the pointer is moving while it is over an element.

MouseOut
Occurs when a user moves the mouse pointer out of an element, or out of one of its children

MouseOver
Occurs when the pointer is moved onto an element, or onto one of its children.

MouseUp
Occurs when a user releases a mouse button over an element.

Reset
Occurs when a form is reset.

Resize
Occurs when the document view is resized.

Scroll
Occurs when an element's scrollbar is being scrolled.

Select
Occurs after the user selects some text (for <input> and <textarea>).

Submit
Occurs when a form is submitted.

TouchCancel
Occurs when one or more touch points have been disrupted in an implementation-specific manner.

TouchEnd
: Occurs when one or more touch points are removed from the touch surface.

TouchMove
: Occurs when one or more touch points are moved along the touch surface.

TouchStart
: Occurs when one or more touch points are placed on the touch surface.

Unload
: Occurs once a page has unloaded (for <body>).

Wheel
: Occurs when user scrolls a mouse wheel over an element.

# Property this

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## this[EventType, bool]

Gets or sets the custom DOM event to listen.

```
Event this[EventType eventType, bool useCapture = false] { get; set; }
```

## Parameters

`eventType` EventType

The event type to listen.

`useCapture` bool

if true, indicates that the user wishes to initiate capture. After initiating capture, all events of the specified type will be dispatched to the registered listeners before being dispatched to any event target beneath them in the tree. Events which are bubbling upward through the tree will not trigger the listeners in this case.

## Property Value

Event

The object that can be used to listen to the custom DOM event.

# Event Abort

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the loading of a resource has been aborted.

```
event EventHandler<DomEventArgs> Abort
```

## Returns

EventHandler<DomEventArgs>

Occurs when the loading of a resource has been aborted.

# Event Blur

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when an element loses focus.

```
event EventHandler<DomEventArgs> Blur
```

## Returns

EventHandler<DomEventArgs>

Occurs when an element loses focus.

# Event Change

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>).

```
event EventHandler<DomEventArgs> Change
```

## Returns

EventHandler<DomEventArgs>

Occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>).

# Event Click

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the user clicks on an element.

```
event EventHandler<DomEventArgs> Click
```

## Returns

EventHandler<DomEventArgs>

Occurs when the user clicks on an element.

# Event DoubleClick

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the user double-clicks on an element.

```
event EventHandler<DomEventArgs> DoubleClick
```

## Returns

EventHandler<DomEventArgs>

Occurs when the user double-clicks on an element.

# Event Error

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when an error occurs while loading an external file.

```
event EventHandler<DomEventArgs> Error
```

## Returns

EventHandler<DomEventArgs>

Occurs when an error occurs while loading an external file.

# Event Focus

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when an element gets focus.

```
event EventHandler<DomEventArgs> Focus
```

## Returns

EventHandler<DomEventArgs>

Occurs when an element gets focus.

# Event KeyDown

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the user presses a key.

```
event EventHandler<DomEventArgs> KeyDown
```

## Returns

EventHandler<DomEventArgs>

Occurs when the user presses a key.

# Event KeyPress

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the user presses a key that produces a character value.

```
event EventHandler<DomEventArgs> KeyPress
```

## Returns

EventHandler<DomEventArgs>

Occurs when the user presses a key that produces a character value.

# Event KeyUp

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the user releases a key.

```
event EventHandler<DomEventArgs> KeyUp
```

## Returns

EventHandler<DomEventArgs>

Occurs when the user releases a key.

# Event Load

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when an object has loaded.

```
event EventHandler<DomEventArgs> Load
```

## Returns

EventHandler<DomEventArgs>

Occurs when an object has loaded.

# Event MouseDown

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the user presses a mouse button over an element.

```
event EventHandler<DomEventArgs> MouseDown
```

## Returns

EventHandler<DomEventArgs>

Occurs when the user presses a mouse button over an element.

# Event MouseMove

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the pointer is moving while it is over an element.

```
event EventHandler<DomEventArgs> MouseMove
```

## Returns

EventHandler<DomEventArgs>

Occurs when the pointer is moving while it is over an element.

# Event MouseOut

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when a user moves the mouse pointer out of an element, or out of one of its children

```
event EventHandler<DomEventArgs> MouseOut
```

## Returns

EventHandler<DomEventArgs>

Occurs when a user moves the mouse pointer out of an element, or out of one of its children

# Event MouseOver

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the pointer is moved onto an element, or onto one of its children.

```
event EventHandler<DomEventArgs> MouseOver
```

## Returns

EventHandler<DomEventArgs>

Occurs when the pointer is moved onto an element, or onto one of its children.

# Event MouseUp

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when a user releases a mouse button over an element.

```
event EventHandler<DomEventArgs> MouseUp
```

## Returns

EventHandler<DomEventArgs>

Occurs when a user releases a mouse button over an element.

# Event Reset

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when a form is reset.

```
event EventHandler<DomEventArgs> Reset
```

## Returns

EventHandler<DomEventArgs>

Occurs when a form is reset.

# Event Resize

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when the document view is resized.

```
event EventHandler<DomEventArgs> Resize
```

## Returns

EventHandler<DomEventArgs>

Occurs when the document view is resized.

# Event Scroll

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when an element's scrollbar is being scrolled.

```
event EventHandler<DomEventArgs> Scroll
```

## Returns

EventHandler<DomEventArgs>

Occurs when an element's scrollbar is being scrolled.

# Event Select

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs after the user selects some text (for <input> and <textarea>).

```
event EventHandler<DomEventArgs> Select
```

## Returns

EventHandler<DomEventArgs>

Occurs after the user selects some text (for <input> and <textarea>).

# Event Submit

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when a form is submitted.

```
event EventHandler<DomEventArgs> Submit
```

## Returns

EventHandler<DomEventArgs>

Occurs when a form is submitted.

# Event TouchCancel

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when one or more touch points have been disrupted in an implementation-specific manner.

```
event EventHandler<DomEventArgs> TouchCancel
```

## Returns

EventHandler<DomEventArgs>

Occurs when one or more touch points have been disrupted in an implementation-specific manner.

# Event TouchEnd

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when one or more touch points are removed from the touch surface.

```
event EventHandler<DomEventArgs> TouchEnd
```

## Returns

EventHandler<DomEventArgs>

Occurs when one or more touch points are removed from the touch surface.

# Event TouchMove

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when one or more touch points are moved along the touch surface.

```
event EventHandler<DomEventArgs> TouchMove
```

## Returns

EventHandler<DomEventArgs>

Occurs when one or more touch points are moved along the touch surface.

# Event TouchStart

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when one or more touch points are placed on the touch surface.

```
event EventHandler<DomEventArgs> TouchStart
```

## Returns

EventHandler<DomEventArgs>

Occurs when one or more touch points are placed on the touch surface.

# Event Unload

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs once a page has unloaded (for <body>).

```
event EventHandler<DomEventArgs> Unload
```

## Returns

EventHandler<DomEventArgs>

Occurs once a page has unloaded (for <body>).

# Event Wheel

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Occurs when user scrolls a mouse wheel over an element.

```
event EventHandler<DomEventArgs> Wheel
```

## Returns

EventHandler<DomEventArgs>

Occurs when user scrolls a mouse wheel over an element.

# Interface IKeyEvent

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents a keyboard event and provides an access to specific keyboard event data.

This event occurs when a user performs some keyboard action, for example presses or releases a keyboard key.

```
public interface IKeyEvent : IUiKeyEventModifier, IEvent, IAutoDisposable
```

**Inherited Members**
IUiKeyEventModifier.KeyModifiers , IEvent.Bubbles , IEvent.Cancelable , IEvent.CurrentTarget , IEvent.EventPhase , IEvent.Target , IEvent.Type , IEvent.PreventDefault() , IEvent.StopPropagation() , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Character
: Gets the character generated by pressing the key.

KeyCode
: Gets the key code that represents a physical key on the keyboard.

# Property Character

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Character

Gets the character generated by pressing the key.

```
string Character { get; }
```

## Property Value

string

The character generated by pressing the key.

# Property KeyCode

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## KeyCode

Gets the key code that represents a physical key on the keyboard.

```
DomKeyCode KeyCode { get; }
```

## Property Value

DomKeyCode

The key code. If the key can't be identified, returns Unknown.

# Interface IMouseEvent

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents a mouse event and provides an access to specific mouse event data.

This event occurs when a user performs an action with a pointing device (such as a mouse), for example moves the mouse or clicks a mouse button.

```
public interface IMouseEvent : IUiKeyEventModifier, IEvent, IAutoDisposable
```

**Inherited Members**
IUiKeyEventModifier.KeyModifiers , IEvent.Bubbles , IEvent.Cancelable , IEvent.CurrentTarget , IEvent.EventPhase , IEvent.Target , IEvent.Type , IEvent.PreventDefault() , IEvent.StopPropagation() , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Button
: Gets the button which was pressed on the mouse to trigger the event.

ClickCount
: Gets the click count.

ClientLocation
: Gets the location of the mouse cursor in the local (DOM content) coordinate system at the time the event occurred.

OffsetLocation
: Gets the location of the mouse cursor in the component's coordinate system at the time the event occurred.

ScreenLocation
: Gets the location of the mouse cursor in the screen's coordinate system at the time the event occurred.

# Property Button

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Button

Gets the button which was pressed on the mouse to trigger the event.

```
DomMouseButton Button { get; }
```

## Property Value

DomMouseButton

The mouse button which was pressed to trigger the event.

# Property ClickCount

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## ClickCount

Gets the click count.

```
uint ClickCount { get; }
```

## Property Value

uint

the count of consecutive clicks that happened in a short amount of time for mouse events, such as `click`, `mouseDown`, `mouseUp`. For all other UI events returns 0.

# Property ClientLocation

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## ClientLocation

Gets the location of the mouse cursor in the local (DOM content) coordinate system at the time the event occurred.

```
Point ClientLocation { get; }
```

## Property Value

Point

The location of the mouse cursor.

# Property OffsetLocation

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## OffsetLocation

Gets the location of the mouse cursor in the component's coordinate system at the time the event occurred.

```
Point OffsetLocation { get; }
```

## Property Value

Point

The location of the mouse cursor.

## Remarks

For example, clicking in the top-left corner of the client area will always result in that the `X` field of the result equals 0, regardless of whether the page is scrolled horizontally.

This API is available since DotNetBrowser 2.1.

# Property ScreenLocation

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## ScreenLocation

Gets the location of the mouse cursor in the screen's coordinate system at the time the event occurred.

```
Point ScreenLocation { get; }
```

## Property Value

Point

The location of the mouse cursor.

# Interface ITouchEvent

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents a touch event and provides an access to specific touch event data.

This event occurs when a user performs an action with a touch device (such as a touchscreen).

```
public interface ITouchEvent : IUiKeyEventModifier, IEvent, IAutoDisposable
```

**Inherited Members**
IUiKeyEventModifier.KeyModifiers , IEvent.Bubbles , IEvent.Cancelable , IEvent.CurrentTarget , IEvent.EventPhase , IEvent.Target , IEvent.Type , IEvent.PreventDefault() , IEvent.StopPropagation() , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

ChangedTouchPoints
Gets the list of touch points that have changed since the last touch event.

TargetTouchPoints
Gets the list of touch points that are specific to the target element.

TouchPoints
Gets the list of touch points that are currently on the screen.

# Property ChangedTouchPoints

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## ChangedTouchPoints

Gets the list of touch points that have changed since the last touch event.

```
IReadOnlyList<ITouchPoint> ChangedTouchPoints { get; }
```

## Property Value

IReadOnlyList<ITouchPoint>

The list of touch points that have changed since the last touch event.

# Property TargetTouchPoints

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## TargetTouchPoints

Gets the list of touch points that are specific to the target element.

```
IReadOnlyList<ITouchPoint> TargetTouchPoints { get; }
```

## Property Value

IReadOnlyList<ITouchPoint>

The list of touch points that are specific to the target element.

# Property TouchPoints

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## TouchPoints

Gets the list of touch points that are currently on the screen.

```
IReadOnlyList<ITouchPoint> TouchPoints { get; }
```

## Property Value

IReadOnlyList<ITouchPoint>

The list of touch points that are currently on the screen.

# Interface IUiKeyEventModifier

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents a DOM UI event that can be fired with the key modifiers.

The KeyModifiers provides access to the data about pressed service keys, such as `Ctrl`, `Alt`, `Shift`, `Meta`, when a keyboard or mouse event occurs.

```
public interface IUiKeyEventModifier : IEvent, IAutoDisposable
```

**Inherited Members**

IEvent.Bubbles , IEvent.Cancelable , IEvent.CurrentTarget , IEvent.EventPhase , IEvent.Target , IEvent.Type , IEvent.PreventDefault() , IEvent.StopPropagation() , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

KeyModifiers

Gets the key modifiers that are applied to the event.

# Property KeyModifiers

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## KeyModifiers

Gets the key modifiers that are applied to the event.

```
IKeyModifiers KeyModifiers { get; }
```

## Property Value

IKeyModifiers

The key modifiers that are applied to the event.

# Interface IWheelEvent

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

A wheel event that provides access to the wheel event data.

This event occurs when a user rotates a wheel on a pointing device (such as a mouse).

```
public interface IWheelEvent : IMouseEvent, IUiKeyEventModifier, IEvent, IAutoDisposable
```

**Inherited Members**

IMouseEvent.Button , IMouseEvent.ClickCount , IMouseEvent.ClientLocation , IMouseEvent.OffsetLocation , IMouseEvent.ScreenLocation , IUiKeyEventModifier.KeyModifiers , IEvent.Bubbles , IEvent.Cancelable , IEvent.CurrentTarget , IEvent.EventPhase , IEvent.Target , IEvent.Type , IEvent.PreventDefault() , IEvent.StopPropagation() , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

DeltaMode

Gets the delta units type.

DeltaX

Gets the amount of units to scroll horizontally.

DeltaY

Gets the amount of units to scroll vertically.

# Property DeltaMode

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## DeltaMode

Gets the delta units type.

```
DeltaMode DeltaMode { get; }
```

## Property Value

DeltaMode

The delta units type used, e.g. pixels, lines, or pages.

# Property DeltaX

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## DeltaX

Gets the amount of units to scroll horizontally.

```
double DeltaX { get; }
```

## Property Value

double

The amount of units to scroll horizontally.

# Property DeltaY

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## DeltaY

Gets the amount of units to scroll vertically.

```
double DeltaY { get; }
```

## Property Value

double

The amount of units to scroll vertically.

# Class KeyEventParameters

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents the DOM keyboard event parameters.

```
public class KeyEventParameters
```

**Inheritance**

object ← KeyEventParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Character

Gets or sets the character generated by pressing the key.

KeyCode

Gets or sets the key code of the pressed key.

UiEventModifierParameters

Gets or sets the DOM UI event parameters with the key modifiers.

# Property Character

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Character

Gets or sets the character generated by pressing the key.

```
public string Character { get; set; }
```

## Property Value

string

The character generated by pressing the key.

# Property KeyCode

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## KeyCode

Gets or sets the key code of the pressed key.

```
public DomKeyCode KeyCode { get; set; }
```

## Property Value

DomKeyCode

The key code of the pressed key.

# Property UiEventModifierParameters

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## UiEventModifierParameters

Gets or sets the DOM UI event parameters with the key modifiers.

```
public UiEventModifierParameters UiEventModifierParameters { get; set; }
```

## Property Value

UiEventModifierParameters

The DOM UI event parameters with the key modifiers.

# Class MouseEventParameters

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents the DOM mouse event parameters.

```
public class MouseEventParameters
```

**Inheritance**

object ← MouseEventParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Button

Gets or sets the number of the mouse button that was pressed to trigger the event.

ClickCount

Gets or sets the click count.

ClientLocation

Gets or sets the location of the mouse cursor in the component's coordinate system at the time the event occurred.

ScreenLocation

Gets or sets the location of the mouse cursor in the screen's coordinate system at the time the event occurred.

UiEventModifierParameters

Gets or sets the DOM UI event parameters with the key modifiers.

# Property Button

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## Button

Gets or sets the number of the mouse button that was pressed to trigger the event.

```
public uint Button { get; set; }
```

## Property Value

uint

The number of the mouse button that was pressed to trigger the event:

- 0 - Main button pressed, usually the left button or the uninitialized state
- 1 - Auxiliary button pressed, usually the wheel button or the middle button (if present)
- 2 - Secondary button pressed, usually the right button
- 3 - Fourth button, typically the Browser Back button
- 4 - Fifth button, typically the Browser Forward button

# Property ClickCount

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## ClickCount

Gets or sets the click count.

```
public uint ClickCount { get; set; }
```

## Property Value

uint

the count of consecutive clicks that happened in a short amount of time for mouse events, such as `click`, `mouseDown`, `mouseUp`. For all other UI events it is equal to 0.

# Property ClientLocation

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## ClientLocation

Gets or sets the location of the mouse cursor in the component's coordinate system at the time the event occurred.

```
public Point ClientLocation { get; set; }
```

## Property Value

Point

the location of the mouse cursor in the component's coordinate system at the time the event occurred. For example, clicking in the top-left corner of the client area will always result in that the `ClientLocation.X` value equals 0, regardless of whether the page is scrolled horizontally.

# Property ScreenLocation

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## ScreenLocation

Gets or sets the location of the mouse cursor in the screen's coordinate system at the time the event occurred.

```
public Point ScreenLocation { get; set; }
```

## Property Value

Point

The location of the mouse cursor in the screen's coordinate system at the time the event occurred.

# Property UiEventModifierParameters

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## UiEventModifierParameters

Gets or sets the DOM UI event parameters with the key modifiers.

```
public UiEventModifierParameters UiEventModifierParameters { get; set; }
```

## Property Value

UiEventModifierParameters

The DOM UI event parameters with the key modifiers.

# Class UiEventModifierParameters

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

Represents the DOM UI event parameters with the key modifiers.

```
public class UiEventModifierParameters
```

**Inheritance**

object ← UiEventModifierParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

EventParameters

Gets or sets the DOM event parameters applicable to the UI events.

KeyModifiers

Gets or sets the key modifiers.

# Property EventParameters

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## EventParameters

Gets or sets the DOM event parameters applicable to the UI events.

```
public EventParameters EventParameters { get; set; }
```

## Property Value

EventParameters

The DOM event parameters applicable to the UI events.

# Property KeyModifiers

Namespace: DotNetBrowser.Dom.Events

Assembly: DotNetBrowser.dll

## KeyModifiers

Gets or sets the key modifiers.

```
public KeyModifiers KeyModifiers { get; set; }
```

## Property Value

KeyModifiers

The key modifiers.

# Namespace DotNetBrowser.Dom.XPath

## Classes

XPathException
: This exception is thrown if XPath evaluation error occurs.

## Interfaces

IXPathResult
: Represents the result of the XPath expression evaluation.

## Enums

XPathResultType
: Represents the XPath result types.

# Interface IXPathResult

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

Represents the result of the XPath expression evaluation.

```
public interface IXPathResult
```

## Properties

Bool
: Gets the boolean value of the evaluation result.

Iterator
: Gets the iterator which allows requesting the items from the Chromium engine during the iterating.

Node
: Gets the DOM node returned as the result of evaluation.

NodesSnapshot
: Gets the snapshot collection of the DOM nodes. If evaluation result type is not ordered or unordered node iterator then returns null.

Numeric
: Gets the numeric value of the evaluation result.

String
: Gets the string value of the evaluation result.

Type
: Gets the result type.

# Property Bool

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

## Bool

Gets the boolean value of the evaluation result.

```
bool? Bool { get; }
```

## Property Value

bool?

The bool value of the evaluation result. If evaluation result type is not boolean, then returns null.

# Property Iterator

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

## Iterator

Gets the iterator which allows requesting the items from the Chromium engine during the iterating.

```
IEnumerable<INode> Iterator { get; }
```

## Property Value

IEnumerable<INode>

The iterator which allows requesting the items from the Chromium engine during the iterating. If evaluation result type is not an ordered or unordered node iterator, then returns null.

# Property Node

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

## Node

Gets the DOM node returned as the result of evaluation.

```
INode Node { get; }
```

## Property Value

INode

The DOM node returned as the result of evaluation. If evaluation result type is not first ordered or any unordered node, then returns null.

# Property NodesSnapshot

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

## NodesSnapshot

Gets the snapshot collection of the DOM nodes. If evaluation result type is not ordered or unordered node iterator then returns null.

```
IEnumerable<INode> NodesSnapshot { get; }
```

## Property Value

IEnumerable<INode>

The snapshot collection of the DOM nodes. If evaluation result type is not an ordered or unordered node iterator, then returns null.

# Property Numeric

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

## Numeric

Gets the numeric value of the evaluation result.

```
double? Numeric { get; }
```

## Property Value

double?

The numeric value of the evaluation result. If evaluation result type is not numeric, then returns null.

# Property String

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

## String

Gets the string value of the evaluation result.

```
string String { get; }
```

## Property Value

string

The string value of the evaluation result. If evaluation result type is not a string, then returns null.

# Property Type

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

## Type

Gets the result type.

```
XPathResultType Type { get; }
```

## Property Value

XPathResultType

The type of this IXPathResult.

# Class XPathException

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

This exception is thrown if XPath evaluation error occurs.

```
public sealed class XPathException : Exception, ISerializable
```

### Inheritance

object ← Exception ← XPathException

### Implements

ISerializable

### Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Constructors

XPathException(string)

Initializes a new instance of the XPathException class with a specified error message.

# Constructor XPathException

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

## XPathException(string)

Initializes a new instance of the XPathException class with a specified error message.

```
public XPathException(string message)
```

## Parameters

`message` string

The error message.

# Enum XPathResultType

Namespace: DotNetBrowser.Dom.XPath

Assembly: DotNetBrowser.dll

Represents the XPath result types.

```
public enum XPathResultType
```

## Fields

`Any = 1`

The result type is not set. In this case the type is a natural type of the evaluation result.

`AnyUnorderedNode = 9`

The result represents a single node matching the XPath expression. The node is not necessarily the first node in the document that matches the expression.

`Boolean = 4`

The result contains a single boolean value.

`FirstOrderedNode = 10`

The result represents the first node in the document matching the XPath expression.

`Number = 2`

The result contains a single number.

`OrderedNodeIterator = 6`

The result represents a set of all nodes matching the XPath expression. The nodes in the result are in the same order that they appear in the document.

`OrderedNodeSnapshot = 8`

The result represents the snapshots of all nodes matching the XPath expression. The nodes in the result are in the same order that they appear in the document.

`String = 3`

The result contains a single string.

`UnorderedNodeIterator = 5`

The result represents a set of all nodes matching the XPath expression. The order of nodes in the result is not defined.

`UnorderedNodeSnapshot = 7`

The result represents the snapshots of all nodes matching the XPath expression. The order of the nodes in the result is not defined.

`Unspecified = 0`

The result type is not set. Possibly, the evaluation error occurred.

# Namespace DotNetBrowser.Downloads

## Classes

DownloadInfo
: The general information about the download.

## Interfaces

IDownload
: Represents a download.

IDownloads
: A service that can be used to work with the downloads.

## Enums

DownloadState
: The download state.

# Class DownloadInfo

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

The general information about the download.

```
public sealed class DownloadInfo
```

**Inheritance**

object ← DownloadInfo

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

MimeType

Gets the MIME type of this download.

SuggestedFileName

Gets the suggested file name.

Url

Gets the URL of this download.

# Property MimeType

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## MimeType

Gets the MIME type of this download.

```
public MimeType MimeType { get; }
```

## Property Value

MimeType

The MIME type of this download.

# Property SuggestedFileName

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## SuggestedFileName

Gets the suggested file name.

```
public string SuggestedFileName { get; }
```

## Property Value

string

The suggested file name.

# Property Url

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Url

Gets the URL of this download.

```
public string Url { get; }
```

## Property Value

string

The URL of this download.

# Enum DownloadState

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

The download state.

```
public enum DownloadState
```

## Fields

`Cancelled = 3`

The download has been cancelled.

`Complete = 2`

The download is completely finished.

`InProgress = 1`

The download is in progress.

`Interrupted = 4`

The download has been interrupted.

# Interface IDownload

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

Represents a download.

```
public interface IDownload : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Browser
Gets the browser that initiated this download.

Extension
Gets the extension that initiated this download.

Info
Gets the additional information about this download.

IsPaused
Indicates whether this download is paused.

State
Gets the current download state.

## Methods

Cancel()
Cancels this download. Will have no effect when the download is already cancelled.

Pause()
Pauses this download. Will have no effect when the download is already paused.

Resume()

Resumes this download. Will have no effect when the download is not paused.

## Events

Canceled

Occurs when the download was canceled.

Finished

Occurs when the download was finished.

Interrupted

Occurs when the download was interrupted.

Paused

Occurs when the download was paused.

Updated

Occurs when the download state was updated.

# Property Browser

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Browser

Gets the browser that initiated this download.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The browser that initiated this download. Can be `null` if no IBrowser is associated with this download.

# Property Extension

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Extension

Gets the extension that initiated this download.

```
IExtension Extension { get; }
```

## Property Value

IExtension

The extension that initiated this download. Can be `null` if no IExtension is associated with this download.

# Property Info

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Info

Gets the additional information about this download.

```
DownloadInfo Info { get; }
```

## Property Value

DownloadInfo

The additional information about this download.

# Property IsPaused

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## IsPaused

Indicates whether this download is paused.

```
bool IsPaused { get; }
```

## Property Value

bool

`true` if the download is paused, `false` otherwise.

## Exceptions

ObjectDisposedException

The IDownload has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property State

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## State

Gets the current download state.

```
DownloadState State { get; }
```

## Property Value

DownloadState

The current download state.

## Exceptions

ObjectDisposedException

The IDownload has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method Cancel

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Cancel()

Cancels this download. Will have no effect when the download is already cancelled.

```
void Cancel()
```

## Exceptions

ObjectDisposedException

The IDownload has already been disposed.

# Method Pause

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Pause()

Pauses this download. Will have no effect when the download is already paused.

```
void Pause()
```

## Exceptions

ObjectDisposedException

The IDownload has already been disposed.

# Method Resume

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Resume()

Resumes this download. Will have no effect when the download is not paused.

```
void Resume()
```

## Exceptions

ObjectDisposedException

The IDownload has already been disposed.

# Event Canceled

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

Occurs when the download was canceled.

```
event EventHandler<CanceledEventArgs> Canceled
```

## Returns

EventHandler<CanceledEventArgs>

Occurs when the download was canceled.

# Event Finished

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

Occurs when the download was finished.

```
event EventHandler<FinishedEventArgs> Finished
```

## Returns

EventHandler<FinishedEventArgs>

Occurs when the download was finished.

# Event Interrupted

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

Occurs when the download was interrupted.

```
event EventHandler<InterruptedEventArgs> Interrupted
```

## Returns

EventHandler<InterruptedEventArgs>

Occurs when the download was interrupted.

# Event Paused

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

Occurs when the download was paused.

```
event EventHandler<PausedEventArgs> Paused
```

## Returns

EventHandler<PausedEventArgs>

Occurs when the download was paused.

# Event Updated

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

Occurs when the download state was updated.

```
event EventHandler<UpdatedEventArgs> Updated
```

## Returns

EventHandler<UpdatedEventArgs>

Occurs when the download state was updated.

# Interface IDownloads

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

A service that can be used to work with the downloads.

```
public interface IDownloads
```

## Properties

Engine

Gets the IEngine instance associated with this object.

Items

Gets an immutable collection of all download items including already completed downloads during this session and the active downloads.

Profile

Gets the IProfile instance associated with this object.

# Property Engine

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Items

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Items

Gets an immutable collection of all download items including already completed downloads during this session and the active downloads.

```
IEnumerable<IDownload> Items { get; }
```

## Property Value

IEnumerable<IDownload>

The collection of all download items including already completed downloads during this session and the active downloads.

# Property Profile

Namespace: DotNetBrowser.Downloads

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Namespace DotNetBrowser.Downloads.Events

## Classes

CanceledEventArgs
: Event arguments for the Canceled event.

DownloadEventArgs
: The base class for IDownload events.

FinishedEventArgs
: Event arguments for the Finished event.

InterruptedEventArgs
: Event arguments for the Interrupted event.

PausedEventArgs
: Event arguments for the Paused event.

UpdatedEventArgs
: Event arguments for the Updated event.

## Enums

DownloadInterruptionReason
: Possible download interruption reason.

# Class CanceledEventArgs

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

Event arguments for the Canceled event.

```
public sealed class CanceledEventArgs : DownloadEventArgs
```

### Inheritance

object ← EventArgs ← DownloadEventArgs ← CanceledEventArgs

### Inherited Members

DownloadEventArgs.Download , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class DownloadEventArgs

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

The base class for IDownload events.

```
public class DownloadEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← DownloadEventArgs

**Derived**

CanceledEventArgs, FinishedEventArgs, InterruptedEventArgs, PausedEventArgs, UpdatedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Download

Gets the download instance which is the source of this event.

# Property Download

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

## Download

Gets the download instance which is the source of this event.

```
public IDownload Download { get; }
```

## Property Value

IDownload

The download instance which is the source of this event.

# Enum DownloadInterruptionReason

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

Possible download interruption reason.

```
public enum DownloadInterruptionReason
```

## Fields

`Crash = 50`

The browser crashed. Internal use only: resume pending downloads if possible.

`FileAccessDenied = 3`

The file cannot be accessed due to security restrictions. "Access Denied".

`FileBlocked = 11`

The file was blocked due to local policy. "Blocked"

`FileFailed = 2`

Generic file operation failure. "File Error".

`FileHashMismatch = 14`

The partial file didn't match the expected hash.

`FileNameTooLong = 5`

The directory or file name is too long. "Path Too Long".

`FileNoSpace = 4`

There is not enough room on the drive. "Disk Full".

`FileSameAsSource = 15`

The source and the target of the download were the same

`FileSecurityCheckFailed = 12`

An attempt to check the safety of the download failed due to unexpected reasons. See http:///crbug.com/153212.

`FileTooLarge = 6`

The file is too large for the file system to handle. "File Too Large".

`FileTooShort = 13`

An attempt was made to seek past the end of a file in opening a file (as part of resuming a previously interrupted download).

`FileTransientError = 10`

The file was in use. Too many files are opened at once. We have run out of memory. "Temporary Problem".

`FileVirusInfected = 7`

The file contains a virus. "Virus".

`NetworkDisconnected = 22`

The network connection has been lost. "Connection Lost".

`NetworkFailed = 20`

Generic network failure. "Network Error".

`NetworkInvalidRequest = 24`

The network request was invalid. This may be due to the original URL or a redirected URL:

- Having an unsupported scheme.
- Being an invalid URL.
- Being disallowed by policy.

`NetworkServerDown = 23`

The server has gone down. "Server Down".

`NetworkTimeout = 21`

The network operation timed out. "Operation Timed Out".

`ServerBadContent = 33`

The server does not have the requested data. "Unable to get file".

`ServerCertProblem = 35`

Server certificate problem.

`ServerContentLengthMismatch = 38`

The server sent fewer bytes than the content-length header. It may indicate that the connection was closed prematurely, or the Content-Length header was invalid. The download is only interrupted if strong validators are present. Otherwise, it is treated as finished.

`ServerFailed = 30`

The server indicates that the operation has failed (generic). "Server Error".

`ServerForbidden = 36`

Server access forbidden.

`ServerNoRange = 31`

The server does not support range requests. Internal use only: must restart from the beginning.

`ServerUnauthorized = 34`

Server didn't authorize access to resource.

`ServerUnreachable = 37`

Unexpected server response. This might indicate that the responding server may not be the intended server.

`Unknown = 1`

There is no interruption reason.

`UserCanceled = 40`

The user canceled the download. "Canceled".

`UserShutdown = 41`

The user shut down the browser. Internal use only: resume pending downloads if possible.

# Class FinishedEventArgs

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

Event arguments for the Finished event.

```
public sealed class FinishedEventArgs : DownloadEventArgs
```

### Inheritance

object ← EventArgs ← DownloadEventArgs ← FinishedEventArgs

### Inherited Members

DownloadEventArgs.Download , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class InterruptedEventArgs

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

Event arguments for the Interrupted event.

```
public sealed class InterruptedEventArgs : DownloadEventArgs
```

**Inheritance**

object ← EventArgs ← DownloadEventArgs ← InterruptedEventArgs

**Inherited Members**

DownloadEventArgs.Download , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

InterruptionReason

Gets the possible download interruption reason.

# Property InterruptionReason

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

## InterruptionReason

Gets the possible download interruption reason.

```
public DownloadInterruptionReason InterruptionReason { get; }
```

## Property Value

DownloadInterruptionReason

The download interruption reason.

# Class PausedEventArgs

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

Event arguments for the Paused event.

```
public sealed class PausedEventArgs : DownloadEventArgs
```

### Inheritance

object ← EventArgs ← DownloadEventArgs ← PausedEventArgs

### Inherited Members

DownloadEventArgs.Download , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class UpdatedEventArgs

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

Event arguments for the Updated event.

```
public class UpdatedEventArgs : DownloadEventArgs
```

**Inheritance**

object ← EventArgs ← DownloadEventArgs ← UpdatedEventArgs

**Inherited Members**

DownloadEventArgs.Download , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CurrentSpeed

Gets the estimated download speed.

Progress

Gets the rough percent complete.

ReceivedBytes

Gets the number or received (downloaded) bytes.

TotalBytes

Gets the total size of the file.

# Property CurrentSpeed

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

## CurrentSpeed

Gets the estimated download speed.

```
public long CurrentSpeed { get; }
```

## Property Value

long

The estimated speed in bytes per second.

# Property Progress

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

## Progress

Gets the rough percent complete.

```
public float Progress { get; }
```

## Property Value

float

Rough percent complete. If the download size is not known, this value will be equal to -1.

# Property ReceivedBytes

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

## ReceivedBytes

Gets the number or received (downloaded) bytes.

```
public long ReceivedBytes { get; }
```

## Property Value

long

The number or received (downloaded) bytes. The received bytes are always less than total bytes.

# Property TotalBytes

Namespace: DotNetBrowser.Downloads.Events

Assembly: DotNetBrowser.dll

## TotalBytes

Gets the total size of the file.

```
public long TotalBytes { get; }
```

## Property Value

long

The total size of the file in bytes.

# Namespace DotNetBrowser.Downloads.Handlers

## Classes

StartDownloadParameters

The parameters of the StartDownloadHandler.

StartDownloadResponse

A response to the StartDownloadHandler.

# Class StartDownloadParameters

Namespace: DotNetBrowser.Downloads.Handlers

Assembly: DotNetBrowser.dll

The parameters of the StartDownloadHandler.

```
public sealed class StartDownloadParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← StartDownloadParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Download

Gets the download which is about to be started.

# Property Download

Namespace: DotNetBrowser.Downloads.Handlers

Assembly: DotNetBrowser.dll

## Download

Gets the download which is about to be started.

```
public IDownload Download { get; }
```

## Property Value

IDownload

The download which is about to be started.

# Class StartDownloadResponse

Namespace: DotNetBrowser.Downloads.Handlers

Assembly: DotNetBrowser.dll

A response to the StartDownloadHandler.

```
public sealed class StartDownloadResponse
```

**Inheritance**

object ← StartDownloadResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a StartDownloadResponse that notifies the engine that the file download should be canceled.

DownloadTo(string)

Creates a StartDownloadResponse that notifies the engine that the file should be downloaded into a specific location.

# Method Cancel

Namespace: DotNetBrowser.Downloads.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a StartDownloadResponse that notifies the engine that the file download should be canceled.

```
public static StartDownloadResponse Cancel()
```

## Returns

StartDownloadResponse

The StartDownloadResponse instance that can be used as a return value in StartDownloadHandler implementation.

# Method DownloadTo

Namespace: DotNetBrowser.Downloads.Handlers

Assembly: DotNetBrowser.dll

## DownloadTo(string)

Creates a StartDownloadResponse that notifies the engine that the file should be downloaded into a specific location.

```
public static StartDownloadResponse DownloadTo(string filePath)
```

### Parameters

`filePath` string

The absolute path to store the downloaded file.

### Returns

StartDownloadResponse

The StartDownloadResponse instance that can be used as a return value in StartDownloadHandler implementation.

### Exceptions

ArgumentException

The `filePath` is null, empty, or contains only blank characters.

# Namespace DotNetBrowser.Engine

## Classes

BinariesExtractionOptions
: The options that are used to configure the Chromium binaries extraction process.

ChromiumBinariesMissingException
: Thrown when the engine initialization fails due to missing compatible Chromium binaries.

ConnectionClosedException
: The exception thrown when the connection to the Chromium engine appears to be closed.

EngineFactory
: Factory class that is used to create IEngine instances.

EngineInitializationException
: The base class for those exceptions that can be thrown during IEngine initialization.

EngineOptions
: The options that are used to configure IEngine instances.

EngineOptions.Builder
: A builder class to construct engine options.

InvalidLicenseException
: Thrown when the given license is invalid.

LicenseException
: The base class for those exceptions that can be thrown during license checking.

MissingDependencyException
: Thrown when Chromium fails to find the required system libraries on Linux.

NoLicenseException
: Thrown when no license found.

PasswordStore
: Defines password store types that are used to specify which encryption storage backend to use to encrypt cookies on Linux.

## Interfaces

IEngine

Provides access to the Chromium engine functionality.

## Enums

BinariesVerificationLevel

The level of the Chromium binaries verification.

ProprietaryFeatures

The list of supported proprietary features.

RenderingMode

The supported rendering modes.

Theme

The Chromium theme, which affects the appearanceof web pages and their contents, as well as Chromium dialogs, such as Print Preview and DevTools.

# Class BinariesExtractionOptions

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The options that are used to configure the Chromium binaries extraction process.

```
public sealed class BinariesExtractionOptions
```

**Inheritance**

object ← BinariesExtractionOptions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

BinariesExtractionOptions(string)

Creates and initializes an instance of BinariesExtractionOptions.

## Properties

AdditionalBinariesSearchPath

Gets the additional custom path to search for an assembly containing compatible Chromium binaries.

CheckLinuxDependencies

Indicates whether it is necessary to check if Linux environment has all required shared libraries.

Chromium64BitPreferred

Indicates whether 64-bit Chromium binaries are preferred when possible.

VerificationLevel

Gets or sets the binaries verification level.

# Constructor BinariesExtractionOptions

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## BinariesExtractionOptions(string)

Creates and initializes an instance of BinariesExtractionOptions.

```
public BinariesExtractionOptions(string additionalBinariesSearchPath = null)
```

## Parameters

`additionalBinariesSearchPath` string

The additional custom path to search for an assembly containing compatible Chromium binaries.

# Property AdditionalBinariesSearchPath

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## AdditionalBinariesSearchPath

Gets the additional custom path to search for an assembly containing compatible Chromium binaries.

```
public string AdditionalBinariesSearchPath { get; }
```

## Property Value

string

The additional custom path to search for an assembly containing compatible Chromium binaries.

# Property CheckLinuxDependencies

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## CheckLinuxDependencies

Indicates whether it is necessary to check if Linux environment has all required shared libraries.

```
public bool CheckLinuxDependencies { get; set; }
```

## Property Value

bool

`true` if it is necessary to check if Linux environment has all required shared libraries, `false` otherwise.

## Remarks

By default, the check is performed.

If `false`, this step is skipped. It's useful for clients that are confident in their environments.

The `ldd` command is used to check presence of the libraries required by Chromium executable.

# Property Chromium64BitPreferred

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Chromium64BitPreferred

Indicates whether 64-bit Chromium binaries are preferred when possible.

```
public bool Chromium64BitPreferred { get; set; }
```

## Property Value

bool

`true` if the 64-bit Chromium binaries are preferred, `false` otherwise.

## Remarks

By default, the bitness of the Chromium process will match the bitness of the .NET process.

However, when this option is set to true, the .NET process is 32-bit, and the operating system is 64-bit, DotNetBrowser will try using the 64-bit Chromium binaries.

# Property VerificationLevel

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## VerificationLevel

Gets or sets the binaries verification level.

```
public BinariesVerificationLevel VerificationLevel { get; set; }
```

## Property Value

BinariesVerificationLevel

## Remarks

By default, the binaries verification level is set to Normal.

# Enum BinariesVerificationLevel

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The level of the Chromium binaries verification.

```
public enum BinariesVerificationLevel
```

## Fields

`Minimal = 1`

Check files presence only.

`Normal = 2`

Check files presence and size.

`Off = 0`

Do not perform binaries verification.

`Strong = 3`

Check files presence and hash.

# Class ChromiumBinariesMissingException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Thrown when the engine initialization fails due to missing compatible Chromium binaries.

```
public class ChromiumBinariesMissingException : EngineInitializationException,
ISerializable
```

**Inheritance**

object ← Exception ← EngineInitializationException ← ChromiumBinariesMissingException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

ChromiumBinariesMissingException(string)

Initializes a new instance with the specified `message`.

# Constructor ChromiumBinariesMissingException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## ChromiumBinariesMissingException(string)

Initializes a new instance with the specified `message`.

```
public ChromiumBinariesMissingException(string message)
```

## Parameters

`message` string

The detailed message.

# Class ConnectionClosedException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The exception thrown when the connection to the Chromium engine appears to be closed.

```
public class ConnectionClosedException : InvalidOperationException, ISerializable
```

### Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← ConnectionClosedException

### Implements

ISerializable

### Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

# Class EngineFactory

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.Core.dll

Factory class that is used to create IEngine instances.

```
public static class EngineFactory
```

**Inheritance**

object ← EngineFactory

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Create()

Creates and returns an IEngine instance.

Create(EngineOptions)

Creates and returns an IEngine instance.

Create(RenderingMode)

Creates and returns an IEngine instance.

CreateAsync()

Asynchronously creates and returns an IEngine instance.

CreateAsync(EngineOptions)

Asynchronously creates and returns an IEngine instance.

CreateAsync(RenderingMode)

Asynchronously creates and returns an IEngine instance.

# Method Create

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.Core.dll

## Create(EngineOptions)

Creates and returns an IEngine instance.

```
public static IEngine Create(EngineOptions options)
```

## Parameters

`options` EngineOptions

The EngineOptions instance that will be used for initializing the engine

## Returns

IEngine

A completely initialized Chromium engine.

## Examples

```
string dataDir = TestUtil.GenerateCustomFolderPath();
Debug.WriteLine($"Data directory: {dataDir}");
Directory.CreateDirectory(dataDir);

EngineOptions engineOptions = new EngineOptions.Builder
    {
        UserDataDirectory = dataDir
    }
   .Build();
IEngine engine = EngineFactory.Create(engineOptions);
```

## Exceptions

EngineInitializationException

The IEngine initialization has failed for some reason.

### Create(RenderingMode)

Creates and returns an IEngine instance.

```
public static IEngine Create(RenderingMode renderingMode)
```

## Parameters

`renderingMode` RenderingMode

The rendering mode that will be used for initializing the engine.

## Returns

IEngine

A completely initialized Chromium engine.

## Remarks

Calling this method is equivalent to calling the Create(EngineOptions) method with the EngineOptions built by calling `new EngineOptions.Builder().Build()` with the rendering mode specified.

## Exceptions

EngineInitializationException

The IEngine initialization has failed for some reason.

### Create()

Creates and returns an IEngine instance.

```
public static IEngine Create()
```

## Returns

IEngine

A completely initialized Chromium engine.

## Remarks

Calling this method is equivalent to calling the Create(EngineOptions) method with the EngineOptions built by calling `new EngineOptions.Builder().Build()`.

## Exceptions

EngineInitializationException

The IEngine initialization has failed for some reason.

# Method CreateAsync

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.Core.dll

## CreateAsync()

Asynchronously creates and returns an IEngine instance.

```
public static Task<IEngine> CreateAsync()
```

### Returns

Task<IEngine>

A task which returns an initialized Chromium engine on completion. The task can throw EngineInitializationException if the IEngine initialization has failed for some reason.

### Remarks

Calling this method is equivalent to calling the CreateAsync(EngineOptions) method with the EngineOptions built by calling `new EngineOptions.Builder().Build()`.

## CreateAsync(EngineOptions)

Asynchronously creates and returns an IEngine instance.

```
public static Task<IEngine> CreateAsync(EngineOptions options)
```

### Parameters

`options` EngineOptions

The EngineOptions instance that will be used for initializing the engine

## Returns

Task<IEngine>

A task which returns an initialized Chromium engine on completion. The task can throw EngineInitializationException if the IEngine initialization has failed for some reason.

## Examples

```
string dataDir = TestUtil.GenerateCustomFolderPath();
Debug.WriteLine($"Data directory: {dataDir}");
Directory.CreateDirectory(dataDir);

EngineOptions engineOptions = new EngineOptions.Builder
    {
        UserDataDirectory = dataDir
    }
   .Build();
IEngine engine = await EngineFactory.CreateAsync(engineOptions);
```

# CreateAsync(RenderingMode)

Asynchronously creates and returns an IEngine instance.

```
public static Task<IEngine> CreateAsync(RenderingMode renderingMode)
```

## Parameters

`renderingMode` RenderingMode

The RenderingMode that will be used for initializing the engine

## Returns

Task<IEngine>

A task, which returns an initialized Chromium engine on completion. The task can throw EngineInitializationException if the IEngine initialization has failed for some reason.

## Remarks

Calling this method is equivalent to calling the CreateAsync(EngineOptions) method with the Engine Options built by calling `new EngineOptions.Builder().Build()` with the rendering mode specified.

# Class EngineInitializationException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The base class for those exceptions that can be thrown during IEngine initialization.

```
public class EngineInitializationException : Exception, ISerializable
```

**Inheritance**

object ← Exception ← EngineInitializationException

**Implements**

ISerializable

**Derived**

ChromiumBinariesMissingException, LicenseException, MissingDependencyException

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

EngineInitializationException(long)

Initializes a new instance of the EngineInitializationException with the specified Chromium `exitCode`.

EngineInitializationException(string)

Initializes a new instance of the EngineInitializationException with the specified `message`.

# Constructor EngineInitializationException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## EngineInitializationException(string)

Initializes a new instance of the EngineInitializationException with the specified `message`.

```
public EngineInitializationException(string message)
```

### Parameters

`message` string

The detail message.

## EngineInitializationException(long)

Initializes a new instance of the EngineInitializationException with the specified Chromium `exitCode`.

```
public EngineInitializationException(long exitCode)
```

### Parameters

`exitCode` long

The Chromium exit code.

# Class EngineOptions

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The options that are used to configure IEngine instances.

```
public class EngineOptions
```

**Inheritance**

object ← EngineOptions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

BinariesExtractionOptions

Gets the options that are used to configure the Chromium binaries extraction process.

ChromiumDirectory

Gets the absolute path to the directory where the Chromium binaries are located.

ChromiumSwitches

Gets an immutable collection of the Chromium switches that will be passed to the Chromium Main process.

CrashDumpDirectory

Gets the path that will be used to store the automatically generated Chromium crash dumps.

DiskCacheSize

Gets the disk cache size.

GoogleApiKey

Gets the Google API key.

GoogleDefaultClientId

Gets the Google client ID.

GoogleDefaultClientSecret

Gets the Google client secret.

IsAutoplayEnabled

Indicates whether video playback will automatically start after page loading.

IsDnsOverHttpsDisabled

Indicates whether the DNS over HTTPS (DoH) protocol is disabled.

IsFileAccessFromFilesAllowed

Indicates whether file access from files is allowed.

IsGpuDisabled

Indicates whether GPU is disabled.

IsIncognitoEnabled

Indicates whether the incognito mode for the IEngine instance is enabled.

IsMediaRoutingEnabled

Indicates whether media routing is enabled for the IEngine instance.

By default, it is disabled.

IsNativeKeyboardInputEnabled

Indicates whether the keyboard events processing using native system API is enabled.

When not applied, the .NET keyboard listeners is used to collect key events, pre-handle them and dispatch to Chromium.

When applied, the OS-specific API is used to collect key events from .NET with subsequent dispatch to Chromium.

IsSandboxDisabled

Indicates whether the sandbox is disabled.

IsTouchMenuDisabled

Indicates whether touch popup menu is disabled.

IsWebSecurityDisabled

Indicates whether the same-origin policy is disabled.

Language

Gets the Chromium language that is used on the default error pages and message dialogs.

LicenseKey

Gets the license key.

PasswordStore
Gets the password store type which specifies the storage backend should be used to encrypt cookies on Linux.

ProprietaryFeatures
Gets the proprietary features enabled for the IEngine instance.

RemoteDebuggingPort
Gets the remote debugging port.

RenderingMode
Gets the rendering mode indicating how the content of the web pages will be rendered.

Schemes
Gets the readonly dictionary of the schemes that will be intercepted.

UserAgent
Gets the custom user agent string that is used to override the default user agent for the IEngine.

UserDataDirectory
Gets an absolute path to the directory where the user data is stored.

## Methods

IsProprietaryFeatureEnabled(ProprietaryFeatures)
Checks whether the given proprietary features are enabled.

# Property BinariesExtractionOptions

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## BinariesExtractionOptions

Gets the options that are used to configure the Chromium binaries extraction process.

```
public BinariesExtractionOptions BinariesExtractionOptions { get; }
```

## Property Value

BinariesExtractionOptions

The options that are used to configure the Chromium binaries extraction process.

# Property ChromiumDirectory

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## ChromiumDirectory

Gets the absolute path to the directory where the Chromium binaries are located.

```
public string ChromiumDirectory { get; }
```

## Property Value

string

The absolute path to the directory where the Chromium binaries are located or an empty string if it was not set.

## Remarks

If the directory contains the Chromium binaries, the library will check them and make sure that they are compatible with the current library version. If the directory does not contain the Chromium binaries, or the binaries aren't compatible with the current library version, the library will search for the binaries inside the assemblies from the application references and working directory and extract them into the given directory programmatically. It might take some time to extract the Chromium binaries from a JAR archive. The time depends on the hardware (CPU and SSD/HDD) performance.

# Property ChromiumSwitches

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## ChromiumSwitches

Gets an immutable collection of the Chromium switches that will be passed to the Chromium Main process.

```
public IEnumerable<string> ChromiumSwitches { get; }
```

## Property Value

IEnumerable<string>

An immutable collection of the Chromium switches that will be passed to the Chromium Main process.

## Remarks

The library does not support all the possible Chromium switches, so there is no guarantee that the passed switches will work. The switches from this list will overwrite the corresponding options configured through the EngineOptions.Builder.

For example, if you configure the remote debugging port using theRemoteDebuggingPort property and then add the `--remote-debugging-port` switch using the ChromiumSwitches property, then the port from the switch will be used.

# Property CrashDumpDirectory

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## CrashDumpDirectory

Gets the path that will be used to store the automatically generated Chromium crash dumps.

```
public string CrashDumpDirectory { get; }
```

## Property Value

string

The path that will be used to store the automatically generated Chromium crash dumps.

# Property DiskCacheSize

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## DiskCacheSize

Gets the disk cache size.

```
public long DiskCacheSize { get; }
```

## Property Value

long

The disk cache size in bytes.

## Remarks

If the disk cache size was not set, then it will be calculated automatically. The size depends on the available disk space in the volume where the disk cache is located.

# Property GoogleApiKey

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## GoogleApiKey

Gets the Google API key.

```
public string GoogleApiKey { get; }
```

## Property Value

string

The Google API key.

## Remarks

Some Chromium features such as Geolocation, Spelling, Speech, etc. use Google APIs, and to access those APIs, an API Key, OAuth 2.0 client ID, and client secret are required. Setting up API keys is optional. If you do not do it, the specific APIs using Google services won't work.

# Property GoogleDefaultClientId

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## GoogleDefaultClientId

Gets the Google client ID.

```
public string GoogleDefaultClientId { get; }
```

## Property Value

string

The Google client ID.

## Remarks

Some Chromium features such as Geolocation, Spelling, Speech, etc. use Google APIs, and to access those APIs, an API Key, OAuth 2.0 client ID, and client secret are required. Setting up API keys is optional. If you do not do it, the specific APIs using Google services won't work.

# Property GoogleDefaultClientSecret

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## GoogleDefaultClientSecret

Gets the Google client secret.

```
public string GoogleDefaultClientSecret { get; }
```

## Property Value

string

The Google client secret.

## Remarks

Some Chromium features such as Geolocation, Spelling, Speech, etc. use Google APIs, and to access those APIs, an API Key, OAuth 2.0 client ID, and client secret are required. Setting up API keys is optional. If you do not do it, the specific APIs using Google services won't work.

# Property IsAutoplayEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsAutoplayEnabled

Indicates whether video playback will automatically start after page loading.

```
public bool IsAutoplayEnabled { get; }
```

## Property Value

bool

`true` if autoplay is enabled, `false` otherwise.

# Property IsDnsOverHttpsDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsDnsOverHttpsDisabled

Indicates whether the DNS over HTTPS (DoH) protocol is disabled.

```
public bool IsDnsOverHttpsDisabled { get; }
```

## Property Value

bool

`true` if DoH is disabled, `false` otherwise.

# Property IsFileAccessFromFilesAllowed

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsFileAccessFromFilesAllowed

Indicates whether file access from files is allowed.

```
public bool IsFileAccessFromFilesAllowed { get; }
```

## Property Value

bool

`true` if the file access from files is allowed, `false` otherwise.

# Property IsGpuDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsGpuDisabled

Indicates whether GPU is disabled.

```
public bool IsGpuDisabled { get; }
```

## Property Value

bool

`true` if GPU is disabled, `false` otherwise.

# Property IsIncognitoEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsIncognitoEnabled

Indicates whether the incognito mode for the IEngine instance is enabled.

```
public bool IsIncognitoEnabled { get; }
```

## Property Value

bool

`true` if the incognito mode is enabled, `false` otherwise.

## Remarks

In the incognito mode the user data such as browsing history, cookies, site data, and the information entered in forms are stored in memory and released once you close the `Engine`.

# Property IsMediaRoutingEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsMediaRoutingEnabled

Indicates whether media routing is enabled for the IEngine instance.

By default, it is disabled.

```
public bool IsMediaRoutingEnabled { get; }
```

## Property Value

bool

`true` if media routing is enabled, `false` otherwise.

# Property IsNativeKeyboardInputEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsNativeKeyboardInputEnabled

Indicates whether the keyboard events processing using native system API is enabled.

When not applied, the .NET keyboard listeners is used to collect key events, pre-handle them and dispatch to Chromium.

When applied, the OS-specific API is used to collect key events from .NET with subsequent dispatch to Chromium.

```
public bool IsNativeKeyboardInputEnabled { get; }
```

## Property Value

bool

`true` if native keyboard input is enabled, `false` otherwise.

# Property IsSandboxDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsSandboxDisabled

Indicates whether the sandbox is disabled.

```
public bool IsSandboxDisabled { get; }
```

## Property Value

bool

`true` if the sandbox is disabled, `false` otherwise.

# Property IsTouchMenuDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsTouchMenuDisabled

Indicates whether touch popup menu is disabled.

```
public bool IsTouchMenuDisabled { get; }
```

## Property Value

bool

`true` if the touch popup menu is disabled, `false` otherwise.

# Property IsWebSecurityDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsWebSecurityDisabled

Indicates whether the same-origin policy is disabled.

```
public bool IsWebSecurityDisabled { get; }
```

## Property Value

bool

`true` if the same-origin policy is disabled, `false` otherwise.

# Property Language

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Language

Gets the Chromium language that is used on the default error pages and message dialogs.

```
public Language Language { get; }
```

## Property Value

Language

The Chromium language.

## Remarks

By default, the language is dynamically configured according to the default locale of the application. If the list of supported languages does not contain the language obtained from the application locale, then the US English language is used.

# Property LicenseKey

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## LicenseKey

Gets the license key.

```
public string LicenseKey { get; }
```

## Property Value

string

The license key.

## Remarks

The license key can be set via adding the "dotnetbrowser.license" file as embedded resource or individually for every IEngine using the LicenseKey property.

# Property PasswordStore

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## PasswordStore

Gets the password store type which specifies the storage backend should be used to encrypt cookies on Linux.

```
public PasswordStore PasswordStore { get; }
```

## Property Value

PasswordStore

The password store type. Returns null if the password store type was not set.

## Remarks

If the password store type was not set, then the encryption storage backend is chosen by Chromium engine automatically.

# Property ProprietaryFeatures

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## ProprietaryFeatures

Gets the proprietary features enabled for the IEngine instance.

```
public ProprietaryFeatures ProprietaryFeatures { get; }
```

## Property Value

ProprietaryFeatures

The proprietary features enabled for the IEngine instance.

# Property RemoteDebuggingPort

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## RemoteDebuggingPort

Gets the remote debugging port.

```
public uint RemoteDebuggingPort { get; }
```

## Property Value

uint

The remote debugging port.

## Remarks

This option enables remote debugging over HTTP at the specific port. When remote debugging is enabled, you can navigate to the `http://localhost:<port>` address from another `Browser` instance or from the Google Chrome browser and debug the loaded web pages using Chrome DevTools. If you use Google Chrome browser for remote debugging, its version must be equal to the Chromium engine version that is used in DotNetBrowser.

# Property RenderingMode

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## RenderingMode

Gets the rendering mode indicating how the content of the web pages will be rendered.

```
public RenderingMode RenderingMode { get; }
```

## Property Value

RenderingMode

The rendering mode.

# Property Schemes

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Schemes

Gets the readonly dictionary of the schemes that will be intercepted.

```
public IReadOnlyDictionary<Scheme, IHandler<InterceptRequestParameters,
InterceptRequestResponse>> Schemes { get; }
```

## Property Value

IReadOnlyDictionary<Scheme, IHandler<InterceptRequestParameters, InterceptRequestResponse>>

The readonly dictionary of the schemes that will be intercepted.

# Property UserAgent

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## UserAgent

Gets the custom user agent string that is used to override the default user agent for the IEngine.

```
public string UserAgent { get; }
```

## Property Value

string

The custom user agent string.

# Property UserDataDirectory

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## UserDataDirectory

Gets an absolute path to the directory where the user data is stored.

```
public string UserDataDirectory { get; }
```

## Property Value

string

An absolute path to the directory where the user data is stored.

## Remarks

The user data directory stores the data such as cache, cookies, history, GPU cache, local storage, visited links, web data, spell checking dictionary files, etc.

**Important**: the user data directory cannot be used at the same time by multiple IEngine instances running in a single or different applications.

# Method IsProprietaryFeatureEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IsProprietaryFeatureEnabled(ProprietaryFeatures)

Checks whether the given proprietary features are enabled.

```
public bool IsProprietaryFeatureEnabled(ProprietaryFeatures proprietaryFeatures)
```

## Parameters

`proprietaryFeatures` ProprietaryFeatures

the features to check.

## Returns

bool

true if the feature is enabled, false otherwise.

# Class EngineOptions.Builder

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

A builder class to construct engine options.

```
public class EngineOptions.Builder
```

**Inheritance**

object ← EngineOptions.Builder

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Remarks

Each of the properties modifies the state of the builder. Builders are not thread-safe and should not be used concurrently from multiple threads without external synchronization.

## Properties

AutoplayEnabled

Enables or disables automatic video playback after page loading.

BinariesExtractionOptions

Gets or sets the options that are used to configure the Chromium binaries extraction process.

ChromiumDirectory

Gets or sets the absolute path to the directory where Chromium binaries are located or the empty string if it was not set.

ChromiumSwitches

Gets a set of the Chromium switches that will be passed to the Chromium Main process.

CrashDumpDirectory

Gets or sets the path that will be used to store automatically generated Chromium crash dumps.

DiskCacheSize

Gets or sets the disk cache size.

DnsOverHttpsDisabled

Disables the DNS over HTTPS (DoH) protocol.

FileAccessFromFilesAllowed

Allows or disallows the file access from files.

GoogleApiKey

Gets or sets the Google API key.

GoogleDefaultClientId

Gets or sets the Google client ID.

GoogleDefaultClientSecret

Gets or sets the Google client secret.

GpuDisabled

Disables or enables GPU.

IncognitoEnabled

Enables or disables the incognito mode for the IEngine instance.

Language

Gets or sets the Chromium language that is used on the default error pages and message dialogs.

LicenseKey

Gets or sets the license key.

MediaRoutingEnabled

Enables or disables the media routing.

By default, it is disabled.

NativeKeyboardInputEnabled

Enables keyboard events processing using native system API in .NET.

When not applied, the .NET keyboard listeners is used to collect key events, pre-handle them and dispatch to Chromium.

When applied, the OS-specific API is used to collect key events from .NET with subsequent dispatch to Chromium.

PasswordStore

Gets or sets the password store type which specifies the storage backend should be used to encrypt cookies on Linux.

ProprietaryFeatures
Gets or sets the proprietary features enabled for the IEngine instance. By default, all the proprietary features are disabled.

RemoteDebuggingPort
Gets or sets the remote debugging port.

RenderingMode
Gets or sets the rendering mode indicating how the content of the web pages will be rendered.

SandboxDisabled
Enables or disables the sandbox.

Schemes
Gets the dictionary of the schemes that will be intercepted.

TouchMenuDisabled
Enables or disables the touch popup menu.

UserAgent
Gets or sets the custom user agent string that is used to override the default user agent for the IEngine.

UserDataDirectory
Gets or sets an absolute path to the directory where the user data is stored.

WebSecurityDisabled
Enables or disables the same-origin policy.

## Methods

Build()
Creates a EngineOptions instance.

# Property AutoplayEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## AutoplayEnabled

Enables or disables automatic video playback after page loading.

```
public bool AutoplayEnabled { get; set; }
```

## Property Value

bool

`true` if autoplay is enabled, `false` otherwise.

# Property BinariesExtractionOptions

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## BinariesExtractionOptions

Gets or sets the options that are used to configure the Chromium binaries extraction process.

```
public BinariesExtractionOptions BinariesExtractionOptions { get; set; }
```

## Property Value

BinariesExtractionOptions

The options that are used to configure the Chromium binaries extraction process.

# Property ChromiumDirectory

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## ChromiumDirectory

Gets or sets the absolute path to the directory where Chromium binaries are located or the empty string if it was not set.

```
public string ChromiumDirectory { get; set; }
```

## Property Value

string

> The absolute path to the directory where Chromium binaries are located or the empty string if it was not set.

## Remarks

If the directory contains the Chromium binaries, the library will check them and make sure that they are compatible with the current library version. If the directory does not contain the Chromium binaries, or the binaries aren't compatible with the current library version, the library will search for the binaries inside the assemblies from the application references and working directory and extract them into the given directory programmatically. It might take some time to extract the Chromium binaries from a JAR archive. The time depends on the hardware (CPU and SSD/HDD) performance.

# Property ChromiumSwitches

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## ChromiumSwitches

Gets a set of the Chromium switches that will be passed to the Chromium Main process.

```
public HashSet<string> ChromiumSwitches { get; }
```

## Property Value

HashSet<string>

A set of the Chromium switches that will be passed to the Chromium Main process.

## Remarks

The library does not support all the possible Chromium switches, so there is no guarantee that the passed switches will work. The switches from this list will overwrite the corresponding options configured through the EngineOptions.Builder.

For example, if you configure the remote debugging port using theRemoteDebuggingPort property and then add the `--remote-debugging-port` switch using the ChromiumSwitches property, then the port from the switch will be used.

# Property CrashDumpDirectory

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## CrashDumpDirectory

Gets or sets the path that will be used to store automatically generated Chromium crash dumps.

```
public string CrashDumpDirectory { get; set; }
```

## Property Value

string

The path that will be used to store automatically generated Chromium crash dumps.

## Remarks

If this property is set to null, the default path will be used. If this property is set to an empty string, crash dump generation will be disabled.

# Property DiskCacheSize

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## DiskCacheSize

Gets or sets the disk cache size.

```
public long DiskCacheSize { get; set; }
```

## Property Value

long

The disk cache size in bytes.

## Remarks

If the disk cache size was not set, then it will be calculated automatically. The size depends on the available disk space in the volume where the disk cache is located.

# Property DnsOverHttpsDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## DnsOverHttpsDisabled

Disables the DNS over HTTPS (DoH) protocol.

```
public bool DnsOverHttpsDisabled { get; set; }
```

## Property Value

bool

`true` if DoH is disabled, `false` otherwise.

## Remarks

By default, DoH is enabled.

# Property FileAccessFromFilesAllowed

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## FileAccessFromFilesAllowed

Allows or disallows the file access from files.

```
public bool FileAccessFromFilesAllowed { get; set; }
```

## Property Value

bool

`true` if the file access is allowed, `false` otherwise.

# Property GoogleApiKey

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## GoogleApiKey

Gets or sets the Google API key.

```
public string GoogleApiKey { get; set; }
```

## Property Value

string

The Google API key.

## Remarks

Some Chromium features such as Geolocation, Spelling, Speech, etc. use Google APIs, and to access those APIs, an API Key, OAuth 2.0 client ID, and client secret are required. Setting up API keys is optional. If you do not do it, the specific APIs using Google services won't work.

# Property GoogleDefaultClientId

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## GoogleDefaultClientId

Gets or sets the Google client ID.

```
public string GoogleDefaultClientId { get; set; }
```

## Property Value

string

The Google client ID.

## Remarks

Some Chromium features such as Geolocation, Spelling, Speech, etc. use Google APIs, and to access those APIs, an API Key, OAuth 2.0 client ID, and client secret are required. Setting up API keys is optional. If you do not do it, the specific APIs using Google services won't work.

# Property GoogleDefaultClientSecret

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## GoogleDefaultClientSecret

Gets or sets the Google client secret.

```
public string GoogleDefaultClientSecret { get; set; }
```

## Property Value

string

The Google client secret.

## Remarks

Some Chromium features such as Geolocation, Spelling, Speech, etc. use Google APIs, and to access those APIs, an API Key, OAuth 2.0 client ID, and client secret are required. Setting up API keys is optional. If you do not do it, the specific APIs using Google services won't work.

# Property GpuDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## GpuDisabled

Disables or enables GPU.

```
public bool GpuDisabled { get; set; }
```

## Property Value

bool

`true` if the GPU is disabled, `false` otherwise.

# Property IncognitoEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## IncognitoEnabled

Enables or disables the incognito mode for the IEngine instance.

```
public bool IncognitoEnabled { get; set; }
```

## Property Value

bool

`true` if the incognito mode is enabled, `false` otherwise.

## Remarks

In the incognito mode the user data such as browsing history, cookies, site data, and the information entered in forms are stored in memory and released once you close the `Engine`.

# Property Language

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Language

Gets or sets the Chromium language that is used on the default error pages and message dialogs.

```
public Language Language { get; set; }
```

## Property Value

Language

The Chromium language.

## Remarks

By default, the language is dynamically configured according to the default locale of the application. If the list of supported languages does not contain the language obtained from the application locale, then the US English language is used. This property allows you to override the default behavior and configure the Chromium engine with the given language.

# Property LicenseKey

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## LicenseKey

Gets or sets the license key.

```
public string LicenseKey { get; set; }
```

## Property Value

string

The license key.

## Remarks

The license key can be set via adding the "dotnetbrowser.license" file as embedded resource or individually for every IEngine using the LicenseKey property.

# Property MediaRoutingEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## MediaRoutingEnabled

Enables or disables the media routing.

By default, it is disabled.

```
public bool MediaRoutingEnabled { get; set; }
```

## Property Value

bool

`true` if the media routing is enabled, `false` otherwise.

# Property NativeKeyboardInputEnabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## NativeKeyboardInputEnabled

Enables keyboard events processing using native system API in .NET.

When not applied, the .NET keyboard listeners is used to collect key events, pre-handle them and dispatch to Chromium.

When applied, the OS-specific API is used to collect key events from .NET with subsequent dispatch to Chromium.

```
public bool NativeKeyboardInputEnabled { get; set; }
```

## Property Value

bool

`true` if native keyboard input is enabled, `false` otherwise.

# Property PasswordStore

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## PasswordStore

Gets or sets the password store type which specifies the storage backend should be used to encrypt cookies on Linux.

```
public PasswordStore PasswordStore { get; set; }
```

## Property Value

PasswordStore

The password store type. Returns null if the password store type was not set.

## Remarks

If the password store type was not set, then the encryption storage backend is chosen by Chromium engine automatically.

# Property ProprietaryFeatures

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## ProprietaryFeatures

Gets or sets the proprietary features enabled for the IEngine instance. By default, all the proprietary features are disabled.

```
public ProprietaryFeatures ProprietaryFeatures { get; set; }
```

## Property Value

ProprietaryFeatures

The proprietary features enabled for the IEngine instance.

# Property RemoteDebuggingPort

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## RemoteDebuggingPort

Gets or sets the remote debugging port.

```
public uint RemoteDebuggingPort { get; set; }
```

## Property Value

uint

The remote debugging port.

## Remarks

This option enables remote debugging over HTTP at the specific port. When remote debugging is enabled, you can navigate to the `http://localhost:<port>` address from another `Browser` instance or from the Google Chrome browser and debug the loaded web pages using Chrome DevTools. If you use Google Chrome browser for remote debugging, its version must be equal to the Chromium engine version that is used in DotNetBrowser.

# Property RenderingMode

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## RenderingMode

Gets or sets the rendering mode indicating how the content of the web pages will be rendered.

```
public RenderingMode RenderingMode { get; set; }
```

## Property Value

RenderingMode

The rendering mode.

# Property SandboxDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## SandboxDisabled

Enables or disables the sandbox.

```
public bool SandboxDisabled { get; set; }
```

## Property Value

bool

`true` if the sandbox is disabled, `false` otherwise.

# Property Schemes

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Schemes

Gets the dictionary of the schemes that will be intercepted.

```
public IDictionary<Scheme, IHandler<InterceptRequestParameters, InterceptRequestResponse>>
Schemes { get; }
```

## Property Value

IDictionary<Scheme, IHandler<InterceptRequestParameters, InterceptRequestResponse>>

The dictionary of the schemes that will be intercepted.

# Property TouchMenuDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## TouchMenuDisabled

Enables or disables the touch popup menu.

```
public bool TouchMenuDisabled { get; set; }
```

## Property Value

bool

`true` if the touch popup menu is disabled, `false` otherwise.

# Property UserAgent

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## UserAgent

Gets or sets the custom user agent string that is used to override the default user agent for the IEngine.

```
public string UserAgent { get; set; }
```

## Property Value

string

The custom user agent string.

## Remarks

This property has the same effect as adding the `--user-agent` switch.

# Property UserDataDirectory

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## UserDataDirectory

Gets or sets an absolute path to the directory where the user data is stored.

```
public string UserDataDirectory { get; set; }
```

## Property Value

string

An absolute path to the directory where the user data is stored.

## Remarks

The user data directory stores the data such as cache, cookies, history, GPU cache, local storage, visited links, web data, spell checking dictionary files, etc.

**Important**: the user data directory cannot be used at the same time by multiple IEngine instances running in a single or different applications.

# Property WebSecurityDisabled

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## WebSecurityDisabled

Enables or disables the same-origin policy.

```
public bool WebSecurityDisabled { get; set; }
```

## Property Value

bool

`true` if the same-origin policy is disabled, `false` otherwise.

# Method Build

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Build()

Creates a EngineOptions instance.

```
public EngineOptions Build()
```

## Returns

EngineOptions

new EngineOptions instance initialized according to the current builder state.

# Interface IEngine

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Provides access to the Chromium engine functionality.

```
public interface IEngine : IDisposable, IAutoDisposable<EngineDisposedEventArgs>,
IAutoDisposable
```

**Inherited Members**
IDisposable.Dispose() , IAutoDisposable<EngineDisposedEventArgs>.Disposed , IAutoDisposable.IsDisposed

## Remarks

To perform operations with the engine, a license key is required. The license key represents a string that can be set via the `dotnetbrowser.license` file or individually for every `IEngine` using the LicenseKey property. If you set the license key via configuring the license file, then please make sure that you set it before creating an `IEngine` instance.

The Chromium engine is running in a separate native process. Communication between the native and .NET process is done through the Inter-Process Communication (IPC) layer that allows transferring data between two processes on a local machine.

The native process allocates memory and system resources that must be released. So, when the engine is no longer needed, it must be disposed through the Dispose() method to shutdown the native process and free all the allocated memory and system resources. For example:

```
IEngine engine = EngineFactory.Create(engineOptions);
//...
engine.Dispose();
```

Any attempt to use an already disposed engine or any of its services will lead to the InvalidOperationException.

To get notifications that the `IEngine` instance has been disposed subscribe to the following event:

```
engine.Disposed += (s, e) => {
// The engine has been disposed or unexpectedly crashed.
});
```

To get notifications that the `IEngine` instance has been unexpectedly crashed use:

```
engine.Disposed += (s, e) => {
    long exitCode = e.ExitCode;
    // The engine has been crashed if this exit code is non-zero.
});
```

```
string dataDir = TestUtil.GenerateCustomFolderPath();
    Debug.WriteLine($"Data directory: {dataDir}");
    Directory.CreateDirectory(dataDir);
EngineOptions engineOptions = new EngineOptions.Builder
    {
        UserDataDirectory = dataDir
    }
   .Build();
IEngine engine = EngineFactory.Create(engineOptions);</code></pre>
```

## Properties

MediaDevices
: Gets the service that allows managing media stream devices.

Options
: Gets the options that were used to initialize this IEngine.

Profiles
: Gets the live collection of the Chromium profiles available in this engine.

Theme
: Gets or sets the current Chromium theme.

## Methods

CreateBrowser()

Creates a new IBrowser instance with the initial "about:blank" web page.

# Property MediaDevices

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## MediaDevices

Gets the service that allows managing media stream devices.

```
IMediaDevices MediaDevices { get; }
```

## Property Value

IMediaDevices

The service that allows managing media stream devices.

# Property Options

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Options

Gets the options that were used to initialize this IEngine.

```
EngineOptions Options { get; }
```

## Property Value

EngineOptions

The EngineOptions that were used to initialize this IEngine.

# Property Profiles

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Profiles

Gets the live collection of the Chromium profiles available in this engine.

```
IProfiles Profiles { get; }
```

## Property Value

IProfiles

The live collection of the Chromium profiles available in this engine.

# Property Theme

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## Theme

Gets or sets the current Chromium theme.

```
Theme Theme { get; set; }
```

## Property Value

Theme

The theme that is currently used by Chromium: system, light, or dark.

## Exceptions

ObjectDisposedException

The IEngine has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method CreateBrowser

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## CreateBrowser()

Creates a new IBrowser instance with the initial "about:blank" web page.

```
IBrowser CreateBrowser()
```

## Returns

IBrowser

The new IBrowser instance.

## Remarks

This browser instance will be bound to the default Chromium profile.

# Class InvalidLicenseException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Thrown when the given license is invalid.

```
public sealed class InvalidLicenseException : LicenseException, ISerializable
```

**Inheritance**

object ← Exception ← EngineInitializationException ← LicenseException ← InvalidLicenseException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Constructors

InvalidLicenseException(string, string)

# Constructor InvalidLicenseException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## InvalidLicenseException(string, string)

```
public InvalidLicenseException(string message, string helpLink = null)
```

### Parameters

`message` string

`helpLink` string

# Class LicenseException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The base class for those exceptions that can be thrown during license checking.

```
public class LicenseException : EngineInitializationException, ISerializable
```

**Inheritance**

object ← Exception ← EngineInitializationException ← LicenseException

**Implements**

ISerializable

**Derived**

InvalidLicenseException, NoLicenseException

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

LicenseException(string)

Initializes a new instance of the EngineInitializationException with the specified `message`.

# Constructor LicenseException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## LicenseException(string)

Initializes a new instance of the EngineInitializationException with the specified `message`.

```
public LicenseException(string message)
```

## Parameters

`message` string

The detail message.

# Class MissingDependencyException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Thrown when Chromium fails to find the required system libraries on Linux.

```
public sealed class MissingDependencyException : EngineInitializationException,
ISerializable
```

**Inheritance**

object ← Exception ← EngineInitializationException ← MissingDependencyException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Remarks

When this exception is thrown, the following actions should be performed:
- make sure that the current Linux distribution and version meets the https://teamdev.com/dotnetbrowser/docs/guides/requirements.html system requirements
- install the required system libraries

# Class NoLicenseException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Thrown when no license found.

```
public class NoLicenseException : LicenseException, ISerializable
```

**Inheritance**

object ← Exception ← EngineInitializationException ← LicenseException ← NoLicenseException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

NoLicenseException()

# Constructor NoLicenseException

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## NoLicenseException()

```
public NoLicenseException()
```

# Class PasswordStore

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Defines password store types that are used to specify which encryption storage backend to use to encrypt cookies on Linux.

```
public sealed class PasswordStore : TypedEnum<string>
```

**Inheritance**

object ← TypedEnum<string> ← PasswordStore

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Auto

Choose encryption automatically.

Basic

Use basic encryption.

Default

Default password store encryption (Auto).

Gnome

Use GNOME encryption.

GnomeKeyring

Use GNOME keyring encryption.

GnomeLibsecret

Use GNOME libsecret encryption.

KWallet

Use KWallet encryption.

KWallet5

Use KWallet5 encryption.

## Operators

explicit operator PasswordStore(string)

Returns the PasswordStore instance by its string representation.

# Field Auto

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Choose encryption automatically.

```
public static readonly PasswordStore Auto
```

## Returns

PasswordStore

Choose encryption automatically.

# Field Basic

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Use basic encryption.

```
public static readonly PasswordStore Basic
```

## Returns

PasswordStore

Use basic encryption.

# Field Default

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Default password store encryption (Auto).

```
public static readonly PasswordStore Default
```

## Returns

PasswordStore

Default password store encryption (Auto).

# Field Gnome

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Use GNOME encryption.

```
public static readonly PasswordStore Gnome
```

## Returns

PasswordStore

Use GNOME encryption.

# Field GnomeKeyring

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Use GNOME keyring encryption.

```
public static readonly PasswordStore GnomeKeyring
```

## Returns

PasswordStore

Use GNOME keyring encryption.

# Field GnomeLibsecret

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Use GNOME libsecret encryption.

```
public static readonly PasswordStore GnomeLibsecret
```

## Returns

PasswordStore

Use GNOME libsecret encryption.

# Field KWallet

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Use KWallet encryption.

```
public static readonly PasswordStore KWallet
```

## Returns

PasswordStore

Use KWallet encryption.

# Field KWallet5

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

Use KWallet5 encryption.

```
public static readonly PasswordStore KWallet5
```

## Returns

PasswordStore

Use KWallet5 encryption.

# Operator explicit operator

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

## explicit operator PasswordStore(string)

Returns the PasswordStore instance by its string representation.

```
public static explicit operator PasswordStore(string value)
```

## Parameters

`value` string

The password store string representation

## Returns

PasswordStore

The PasswordStore instance that corresponds to the provided string representation.

# Enum ProprietaryFeatures

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The list of supported proprietary features.

```
[Flags]
public enum ProprietaryFeatures
```

## Fields

`Aac = 64`

Represents support of the AAC codec.

**Important**: AAC codec is a proprietary component. By enabling this codec you state that you are aware that AAC is a proprietary component and you should have a license in order to use it. For more information, you could contact patent holders: Via Licensing and MPEG LA. TeamDev shall not be responsible for your use of AAC codec.

`H264 = 1`

Represents support of the H.264 codec.

**Important**: H.264 codec is a proprietary component. By enabling this codec you state that you are aware that H.264 is a proprietary component and you should have a license in order to use it. For more information, you could contact patent holders: Via Licensing and MPEG LA. TeamDev shall not be responsible for your use of H.264 codec.

`Hevc = 16`

Represents support of the HEVC codec.

**Important**: HEVC codec is a proprietary component. By enabling this codec you state that you are aware that HEVC is a proprietary component and you should have a license in order to use it. For more information, you could contact patent holders: Via Licensing and MPEG LA. TeamDev shall not be responsible for your use of HEVC codec.

`None = 0`

No proprietary features enabled.

`Widevine = 128`

Represents support of the Widevine DRM decryption.

**Important**: Widevine is a Google proprietary component, governed by its own terms of use. For more information, see https://www.widevine.com/ . TeamDev shall not be responsible for your use of Widevine.

# Enum RenderingMode

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The supported rendering modes.

```
public enum RenderingMode
```

## Fields

`HardwareAccelerated = 1`

The engine renders content using GPU and displays it directly on a native surface (window).

`OffScreen = 2`

The engine renders content using GPU and copies the pixels to the process memory.

# Enum Theme

Namespace: DotNetBrowser.Engine

Assembly: DotNetBrowser.dll

The Chromium theme, which affects the appearanceof web pages and their contents, as well as Chromium dialogs, such as Print Preview and DevTools.

```
public enum Theme
```

## Fields

`Dark = 2`

The dark theme.

`Light = 1`

The light theme.

`System = 3`

The theme inherited from the operating system settings.

`Unspecified = 0`

Reserved value.

# Namespace DotNetBrowser.Engine.Events

## Classes

EngineDisposedEventArgs

The event arguments for the event indicating that the IEngine has been disposed.

# Class EngineDisposedEventArgs

Namespace: DotNetBrowser.Engine.Events

Assembly: DotNetBrowser.dll

The event arguments for the event indicating that the IEngine has been disposed.

```
public class EngineDisposedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← EngineDisposedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Engine

Gets the IEngine instance that has been disposed.

ExitCode

Gets the exit code of the main Chromium process that has been terminated.

# Property Engine

Namespace: DotNetBrowser.Engine.Events

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance that has been disposed.

```
public IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance that has been disposed.

# Property ExitCode

Namespace: DotNetBrowser.Engine.Events

Assembly: DotNetBrowser.dll

## ExitCode

Gets the exit code of the main Chromium process that has been terminated.

```
public long ExitCode { get; }
```

## Property Value

long

The exit code of the main Chromium process that has been terminated. If this code is non-zero, the process terminated unexpectedly due to the crash.

# Namespace DotNetBrowser.Extensions

## Classes

ExtensionInstallationException

Thrown when the extension installation has failed for some reason.

## Interfaces

IExtension

A Chromium extension.

Extensions can be installed via the IExtensions instance. It's possible to install an extension either from the CRX file or from the Chrome WebStore. Each extension is installed on a per-profile basis, and is not shared with other profiles.

IExtensionAction

The extension action is a clickable extension icon.

In Chrome, extension actions are located in the top-right corner of the toolbar. Clicking the icon usually results in showing an extension action popup. If the extension requests to show a popup, the Open ExtensionActionPopupHandler is invoked. The extension can also handle the click action without showing any popups, in which case the handler is not invoked.

IExtensions

A service that allows managing extensions.

## Enums

ExtensionActionType

The extension action type.

ExtensionPermission

The extension permission types.

# Enum ExtensionActionType

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

The extension action type.

```
public enum ExtensionActionType
```

## Fields

`Action = 3`

The chrome.action type available in Manifest V3.

`Browser = 2`

The browser action can be applied to all web pages within a browser.

`ExtensionActionTypeUnspecified = 0`

Reserved value.

`Page = 1`

The page action can be applied to some web pages, but is not applicable to all pages.

# Class ExtensionInstallationException

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

Thrown when the extension installation has failed for some reason.

```
public class ExtensionInstallationException : InvalidOperationException, ISerializable
```

### Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← ExtensionInstallationException

### Implements

ISerializable

### Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

ExtensionInstallationException(string)

# Constructor ExtensionInstallationException

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## ExtensionInstallationException(string)

```
public ExtensionInstallationException(string message)
```

## Parameters

`message` string

# Enum ExtensionPermission

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

The extension permission types.

```
public enum ExtensionPermission
```

## Fields

`ActiveTab = 2`

Gives temporary access to the active tab through a user gesture.

`Alarms = 3`

Gives access to the chrome.alarms API.

`Background = 4`

Makes Chrome start up early (as soon as the user logs into their computer—before the user launches Chrome), and shut down late (even after its last window is closed until the user explicitly quits Chrome). In DotNetBrowser this permission has no effect.

`Bookmarks = 5`

Gives access to the `chrome.bookmarks` API.

`BrowsingData = 6`

Gives access to the `chrome.browsingData` API.

`CertificateProvider = 7`

ChromeOS only. Gives access to the `chrome.certificateProvider` API.

`ClipboardRead = 8`

Required if the extension uses `document.execCommand('paste')`.

`ClipboardWrite = 9`

Required if the extension uses `document.execCommand('copy')` or `document.execCommand('cut')`.

`ContentSettings = 10`

Gives access to the `chrome.contentSettings` API.

`ContextMenus = 11`

Gives access to the `chrome.contextMenus` API.

`Cookies = 12`

Gives access to the `chrome.cookies` API.

`Debugger = 13`

Gives access to the `chrome.debugger` API.

`DeclarativeContent = 14`

Gives access to the `chrome.declarativeContent` API.

`DeclarativeNetRequest = 15`

Gives access to the `chrome.declarativeNetRequest` API.

`DeclarativeNetRequestFeedback = 17`

Gives access to the `chrome.declarativeNetRequestFeedback` API which returns information on declarative rules matched.

`DeclarativeNetRequestWithHostAccess = 16`

Gives access to the `chrome.declarativeNetRequest` API, but requires host permissions to the request URL and initiator to act on a request.

`DeclarativeWebRequest = 18`

Gives access to the `chrome.declarativeWebRequest` API. This API is deprecated. Check out the `chrome.declarativeNetRequest` API instead.

`DesktopCapture = 19`

Gives access to the `chrome.desktopCapture` API.

`DocumentScan = 20`

Gives access to the `chrome.documentScan` API.

`Downloads = 21`

Gives access to the `chrome.downloads` API.

`DownloadsOpen = 69`

The permission required to use the `chrome.downloads.open()` method.

`DownloadsUI = 70`

The permission required to use the `chrome.downloads.setUiOptions()` method.In DotNetBrowser this method has no effect.

`EnterpriseDeviceAttributes = 22`

Gives access to the `chrome.enterprise.deviceAttributes` API.

`EnterpriseHardwarePlatform = 23`

Gives access to the `chrome.enterprise.hardwarePlatform` API.

`EnterpriseNetworkingAttributes = 24`

Gives access to the `chrome.enterprise.networkingAttributes` API.

`EnterprisePlatformKeys = 25`

Gives access to the `chrome.enterprise.platformKeys` API.

`Experimental = 26`

Gives access to the `chrome.experimental.*` APIs.

`ExtensionPermissionUnspecified = 0`

Reserved value.

`FileBrowserHandler = 27`

ChromeOS only. Gives access to the `chrome.fileBrowserHandler` API.

`FileSystemProvider = 28`

ChromeOS only. Gives access to the `chrome.fileSystemProvider` API.

`FontSettings = 29`

Gives access to the `chrome.fontSettings` API.

`Gcm = 30`

Gives access to the `chrome.gcm` API.

`Geolocation = 31`

Allows the extension to use the geolocation API without prompting the user for permission.

`History = 32`

Gives access to the `chrome.history` API.

`Identity = 33`

Gives access to the `chrome.identity` API.

`Idle = 34`

Gives access to the `chrome.idle` API.

`LoginState = 35`

ChromeOS only. Gives access to the `chrome.loginState` API.

`Management = 36`

Gives access to the `chrome.management` API.

`NativeMessaging = 37`

Gives access to the native messaging API.

`Notifications = 38`

Gives access to the `chrome.notifications` API.

`Offscreen = 39`

Gives access to the `chrome.offscreen` API.

`PageCapture = 40`

Gives access to the `chrome.pageCapture` API.

`PlatformKeys = 41`

ChromeOS only. Gives access to the `chrome.platformKeys` API.

`Power = 42`

Gives access to the `chrome.power` API.

`PrinterProvider = 43`

Gives access to the `chrome.printerProvider` API.

`Printing = 44`

ChromeOS only. Gives access to the `chrome.printing` API.

`PrintingMetrics = 45`

ChromeOS only. Gives access to the `chrome.printingMetrics` API.

`Privacy = 46`

Gives access to the `chrome.privacy` API.

`Process = 47`

Gives access to the `chrome.processes` API.

`Proxy = 48`

Gives access to the `chrome.proxy` API.

`Scripting = 49`

Gives access to the `chrome.scripting` API.

`Search = 50`

Gives access to the `chrome.search` API.

`Sessions = 51`

Gives access to the `chrome.sessions` API.

`Storage = 52`

Gives access to the `chrome.storage` API.

`SystemCpu = 53`

Gives access to the `chrome.system.cpu` API.

`SystemDisplay = 54`

Gives access to the `chrome.system.display` API.

`SystemMemory = 55`

Gives access to the `chrome.system.memory` API.

`SystemStorage = 56`

Gives access to the `chrome.system.storage` API.

`TabCapture = 57`

Gives access to the `chrome.tabCapture` API.

`TabGroups = 58`

Gives access to the `chrome.tabGroups` API.

`Tabs = 59`

Gives access to privileged fields of the Tab objects used by several APIs including `chrome.tabs` and `chrome.windows`. In many circumstances your extension will not need to declare the "tabs" permission to make use of these APIs.

`TopSites = 60`

Gives access to the `chrome.topSites` API.

`Tts = 61`

Gives access to the `chrome.tts` API.

`TtsEngine = 62`

Gives access to the `chrome.ttsEngine` API.

`Unknown = 1`

An unknown permission type.

`UnlimitedStorage = 63`

Provides an unlimited quota for storing client-side data, such as databases and local storage files. Without this permission, the extension is limited to 10 MB of local storage.

`VpnProvider = 64`

ChromeOS only. Gives access to the `chrome.vpnProvider` API.

`Wallpaper = 65`

ChromeOS only. Gives access to the `chrome.wallpaper` API.

`WebAuthenticationProxy = 71`

Gives access to the `chrome.webAuthenticationProxy` API.

`WebNavigation = 66`

Gives access to the `chrome.webNavigation` API.

`WebRequest = 67`

Gives access to the `chrome.webRequest` API.

`WebRequestBlocking = 68`

Required if the extension uses the `chrome.webRequest` API in a blocking fashion.

# Interface IExtension

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

A Chromium extension.

Extensions can be installed via the IExtensions instance. It's possible to install an extension either from the CRX file or from the Chrome WebStore. Each extension is installed on a per-profile basis, and is not shared with other profiles.

```
public interface IExtension : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Remarks

There are certain limitations related to Chromium tabs, windows, and other UI elements. The `windows.Window` Extensions API object is mapped to a native window associated with the browser, not the .NET window it is embedded into. Thus, methods and properties of the `windows.Window` object, such as `Window.width`, `Window.height`, `Window.focused`, etc. are bound to that native window.

From Chromium's perspective, each browser is a window with one tab. Extensions that manipulate tabs, tab groups, or windows should be used with caution, as they might not work as expected.

Below is the list of Chromium Extension APIs that are not supported in DotNetBrowser:

- `chrome.tabs.discard`,
- `chrome.tabs.remove`,
- `chrome.tabs.duplicate`,
- `chrome.windows.remove`,
- `chrome.windows.update`,
- `chrome.window.create` with more than one URL in parameters,
- `chrome.sessions.restore`,
- `chrome.downloads.*`,
- `chrome.desktopCapture.chooseDesktopMedia`.

If an extension calls an unsupported method that returns a `Promise`, it will be rejected with an error. If a method accepts a callback, the `chrome.runtime.lastError` property will be set to an error.

## Properties

HasAction

Indicates whether an extension has an action.

Id

Gets the identifier of the extension.

Name

Gets the name of the extension.

OpenExtensionPopupHandler

Gets or sets a handler that is used when the extension is about to open a popup window.

Profile

Gets the profile instance associated with this extension.

## Methods

GetAction(IBrowser)

Gets the extension action for the given `browser`.

Uninstall()

Uninstalls this extension.

# Property HasAction

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## HasAction

Indicates whether an extension has an action.

```
bool HasAction { get; }
```

## Property Value

bool

## Exceptions

ObjectDisposedException

The IExtension has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Id

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Id

Gets the identifier of the extension.

```
string Id { get; }
```

## Property Value

string

The identifier of the extension.

## Exceptions

ObjectDisposedException

The IExtension has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Name

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Name

Gets the name of the extension.

```
string Name { get; }
```

## Property Value

string

The name of the extension.

## Exceptions

ObjectDisposedException

The IExtension has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property OpenExtensionPopupHandler

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## OpenExtensionPopupHandler

Gets or sets a handler that is used when the extension is about to open a popup window.

```
IHandler<OpenExtensionPopupParameters> OpenExtensionPopupHandler { get; set; }
```

## Property Value

IHandler<OpenExtensionPopupParameters>

## Remarks

The callback is invoked by the following extension API functions:

- `chrome.tabs.create`,
- `chrome.windows.create`,
- `chrome.runtime.openOptionsPage`,
- `setUninstallURL`, upon uninstalling an extension.

## Exceptions

ObjectDisposedException

The IExtension has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Profile

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Profile

Gets the profile instance associated with this extension.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The profile associated with this extension.

# Method GetAction

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## GetAction(IBrowser)

Gets the extension action for the given `browser`.

```
IExtensionAction GetAction(IBrowser browser)
```

## Parameters

`browser` IBrowser

The browser that will be associated with the action.

## Returns

IExtensionAction

The extension action for the passed browser or `null` if the action is not available.

## Exceptions

ObjectDisposedException

The IExtension has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method Uninstall

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Uninstall()

Uninstalls this extension.

```
Task<bool> Uninstall()
```

## Returns

Task<bool>

A task that completes when the extension is uninstalled. The result value is `true` if the extension was uninstalled successfully, `false` otherwise.

## Exceptions

ObjectDisposedException

The IExtension has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Interface IExtensionAction

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

The extension action is a clickable extension icon.

In Chrome, extension actions are located in the top-right corner of the toolbar. Clicking the icon usually results in showing an extension action popup. If the extension requests to show a popup, the Open ExtensionActionPopupHandler is invoked. The extension can also handle the click action without showing any popups, in which case the handler is not invoked.

```
public interface IExtensionAction : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Badge
Gets the extension action badge text for the given browser.

Browser
Gets the browser instance associated with this extension action.

Extension
Gets the extension associated with this extension action.

Icon
Gets the extension action icon for the given browser.

IsEnabled
Indicates whether the extension action is currently enabled.

Tooltip
Gets the extension action tooltip.

Type
Gets the extension action type.

## Methods

Click()

Simulates a click on the extension action icon for the given browser.

## Events

Updated

Occurs when the extension action has been updated.

# Property Badge

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Badge

Gets the extension action badge text for the given browser.

```
string Badge { get; }
```

## Property Value

string

The extension action badge text for the given browser.

## Exceptions

ObjectDisposedException

The IExtensionAction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Browser

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Browser

Gets the browser instance associated with this extension action.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The browser instance associated with this extension action.

# Property Extension

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Extension

Gets the extension associated with this extension action.

```
IExtension Extension { get; }
```

## Property Value

IExtension

The extension associated with this extension action.

# Property Icon

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Icon

Gets the extension action icon for the given browser.

```
Bitmap Icon { get; }
```

## Property Value

Bitmap

The extension action icon for the given browser.

## Exceptions

ObjectDisposedException

The IExtensionAction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property IsEnabled

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## IsEnabled

Indicates whether the extension action is currently enabled.

```
bool IsEnabled { get; }
```

## Property Value

bool

`true` if the extension action is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IExtensionAction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Tooltip

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Tooltip

Gets the extension action tooltip.

```
string Tooltip { get; }
```

## Property Value

string

The extension action tooltip.

## Exceptions

ObjectDisposedException

The IExtensionAction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Type

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Type

Gets the extension action type.

```
ExtensionActionType Type { get; }
```

## Property Value

ExtensionActionType

The extension action type.

## Exceptions

ObjectDisposedException

The IExtensionAction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method Click

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Click()

Simulates a click on the extension action icon for the given browser.

```
void Click()
```

## Remarks

Clicking a disabled action is no-op.

## Exceptions

ObjectDisposedException

The IExtensionAction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Event Updated

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

Occurs when the extension action has been updated.

```
event EventHandler<ExtensionActionUpdatedEventArgs> Updated
```

## Returns

EventHandler<ExtensionActionUpdatedEventArgs>

Occurs when the extension action has been updated.

## Remarks

You can use this event to monitor changes in the extension action badge, tooltip, or icon.

# Interface IExtensions

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

A service that allows managing extensions.

```
public interface IExtensions : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

All

Gets all extensions that are currently installed for the profile.

InstallExtensionHandler

Gets or sets a handler that is used when the user installs an extension from Chrome Web Store.

Profile

Gets the IProfile instance associated with this object.

UninstallExtensionHandler

Gets or sets a handler that is used when the extension is about to uninstall.

## Methods

Install(string)

Installs an extension from the local CRX file by the given `path`.

## Events

ExtensionInstalled

Occurs when an extension is installed.

ExtensionUninstalled

Occurs when an extension is uninstalled.

ExtensionUpdated

Occurs when an extension is updated.

# Property All

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## All

Gets all extensions that are currently installed for the profile.

```
IReadOnlyCollection<IExtension> All { get; }
```

## Property Value

IReadOnlyCollection<IExtension>

## Exceptions

ObjectDisposedException

The IExtensions has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property InstallExtensionHandler

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## InstallExtensionHandler

Gets or sets a handler that is used when the user installs an extension from Chrome Web Store.

```
IHandler<InstallExtensionParameters, InstallExtensionResponse> InstallExtensionHandler {
get; set; }
```

## Property Value

IHandler<InstallExtensionParameters, InstallExtensionResponse>

## Remarks

Use the Install value to install the extension.

Use the Cancel value to cancel the installation.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel will be used.

## Exceptions

ObjectDisposedException

The IExtensions has already been disposed.

# Property Profile

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Property UninstallExtensionHandler

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## UninstallExtensionHandler

Gets or sets a handler that is used when the extension is about to uninstall.

```
IHandler<UninstallExtensionParameters, UninstallExtensionResponse>
UninstallExtensionHandler { get; set; }
```

## Property Value

IHandler<UninstallExtensionParameters, UninstallExtensionResponse>

## Remarks

Use the Uninstall value to uninstall the extension.

Use the Cancel value to cancel the uninstalling.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Cancel will be used.

## Exceptions

ObjectDisposedException

The IExtensions has already been disposed.

# Method Install

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

## Install(string)

Installs an extension from the local CRX file by the given `path`.

```
Task<IExtension> Install(string path)
```

## Parameters

`path` string

The path to the extension CRX3 package.

## Returns

Task<IExtension>

A task that completes when the extension installation is completed. If the installation is successful, the result will contain an IExtension instance that corresponds to the installed extension. If the extension installation fails, the task will complete with an ExtensionInstallationException describing the error.

## Remarks

Extensions can be installed for an incognito profile.

Extensions installed for the default profile in the engine with IsIncognitoEnabled option set to `true` are also installed for the default profile. Extensions installed for the default profile in the engine with IsIncognitoEnabled option set to `false` are not enabled when the engine is launched with this option.

If the same extension is already installed, this method returns the already installed extension. If the extension version is newer than that is already installed, this method replaces the installed extension.

## Exceptions

ObjectDisposedException

The IExtensions has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Event ExtensionInstalled

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

Occurs when an extension is installed.

```
event EventHandler<ExtensionInstalledEventArgs> ExtensionInstalled
```

## Returns

EventHandler<ExtensionInstalledEventArgs>

Occurs when an extension is installed.

# Event ExtensionUninstalled

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

Occurs when an extension is uninstalled.

```
event EventHandler<ExtensionUninstalledEventArgs> ExtensionUninstalled
```

## Returns

EventHandler<ExtensionUninstalledEventArgs>

Occurs when an extension is uninstalled.

# Event ExtensionUpdated

Namespace: DotNetBrowser.Extensions

Assembly: DotNetBrowser.dll

Occurs when an extension is updated.

```
event EventHandler<ExtensionUpdatedEventArgs> ExtensionUpdated
```

## Returns

EventHandler<ExtensionUpdatedEventArgs>

Occurs when an extension is updated.

# Namespace DotNetBrowser.Extensions.Events

## Classes

ExtensionActionUpdatedEventArgs

Event arguments for the Updated event.

ExtensionEventArgs

The base class for the extension-related event arguments.

ExtensionInstalledEventArgs

Event arguments for the ExtensionInstalled event.

ExtensionUninstalledEventArgs

Event arguments for the ExtensionUninstalled event.

ExtensionUpdatedEventArgs

Event arguments for the ExtensionUpdated event.

# Class ExtensionActionUpdatedEventArgs

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

Event arguments for the Updated event.

```
public sealed class ExtensionActionUpdatedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← ExtensionActionUpdatedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Browser

Gets the browser for which the extension action has been updated.

ExtensionAction

Gets the extension action which is the source of this event.

# Property Browser

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

## Browser

Gets the browser for which the extension action has been updated.

```
public IBrowser Browser { get; }
```

## Property Value

IBrowser

# Property ExtensionAction

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

## ExtensionAction

Gets the extension action which is the source of this event.

```
public IExtensionAction ExtensionAction { get; }
```

## Property Value

IExtensionAction

The extension action which is the source of this event.

# Class ExtensionEventArgs

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

The base class for the extension-related event arguments.

```
public class ExtensionEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← ExtensionEventArgs

**Derived**

ExtensionInstalledEventArgs, ExtensionUpdatedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Extension

Gets the extension for which the event occurred.

# Property Extension

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

## Extension

Gets the extension for which the event occurred.

```
public IExtension Extension { get; }
```

## Property Value

IExtension

# Class ExtensionInstalledEventArgs

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

Event arguments for the ExtensionInstalled event.

```
public sealed class ExtensionInstalledEventArgs : ExtensionEventArgs
```

### Inheritance

object ← EventArgs ← ExtensionEventArgs ← ExtensionInstalledEventArgs

### Inherited Members

ExtensionEventArgs.Extension , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class ExtensionUninstalledEventArgs

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

Event arguments for the ExtensionUninstalled event.

```
public sealed class ExtensionUninstalledEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← ExtensionUninstalledEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ExtensionId

Gets the identifier of the uninstalled extension.

# Property ExtensionId

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

## ExtensionId

Gets the identifier of the uninstalled extension.

```
public string ExtensionId { get; }
```

## Property Value

string

# Class ExtensionUpdatedEventArgs

Namespace: DotNetBrowser.Extensions.Events

Assembly: DotNetBrowser.dll

Event arguments for the ExtensionUpdated event.

```
public sealed class ExtensionUpdatedEventArgs : ExtensionEventArgs
```

### Inheritance

object ← EventArgs ← ExtensionEventArgs ← ExtensionUpdatedEventArgs

### Inherited Members

ExtensionEventArgs.Extension , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Namespace DotNetBrowser.Extensions.Handlers

## Classes

InstallExtensionParameters

The parameters of the InstallExtensionHandler.

InstallExtensionResponse

The response to the InstallExtensionHandler.

OpenExtensionPopupParameters

The parameters of the OpenExtensionPopupHandler.

UninstallExtensionParameters

The parameters of the UninstallExtensionHandler.

UninstallExtensionResponse

The response to the UninstallExtensionHandler.

# Class InstallExtensionParameters

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The parameters of the InstallExtensionHandler.

```
public sealed class InstallExtensionParameters
```

**Inheritance**

object ← InstallExtensionParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ExtensionCrxFile

Gets the absolute path to the CRX file the extension is about to be installed from.

ExtensionId

Gets the ID of the extension to install.

ExtensionName

Gets the name of the extension to install.

ExtensionVersion

Gets the version of the extension to install.

Hosts

Gets the hosts that the extension requests access to.

Permissions

Gets the required extension permissions.

# Property ExtensionCrxFile

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## ExtensionCrxFile

Gets the absolute path to the CRX file the extension is about to be installed from.

```
public string ExtensionCrxFile { get; }
```

## Property Value

string

## Remarks

The file will be automatically deleted after the installation is completed or canceled. You can make a copy of the CRX file and store it for future use. For example, you can use the file to pack the extension into the application bundle and install it programmatically using Install(string)

# Property ExtensionId

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## ExtensionId

Gets the ID of the extension to install.

```
public string ExtensionId { get; }
```

## Property Value

string

# Property ExtensionName

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## ExtensionName

Gets the name of the extension to install.

```
public string ExtensionName { get; }
```

## Property Value

string

# Property ExtensionVersion

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## ExtensionVersion

Gets the version of the extension to install.

```
public string ExtensionVersion { get; }
```

## Property Value

string

# Property Hosts

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## Hosts

Gets the hosts that the extension requests access to.

```
public IReadOnlyList<string> Hosts { get; }
```

## Property Value

IReadOnlyList<string>

## Remarks

The values are represented in the form of matches pattern .

Please note that values are being processed by engine and might be not equal to the extension manifest values. For example, some slashes and wildcards could be added.

# Property Permissions

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## Permissions

Gets the required extension permissions.

```
public IReadOnlyList<ExtensionPermission> Permissions { get; }
```

## Property Value

IReadOnlyList<ExtensionPermission>

# Class InstallExtensionResponse

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The response to the InstallExtensionHandler.

```
public sealed class InstallExtensionResponse
```

**Inheritance**

object ← InstallExtensionResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Cancel

The InstallExtensionResponse that notifies the engine that the extension installation should be canceled.

Install

The InstallExtensionResponse that notifies the engine that the extension should be installed.

# Field Cancel

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The InstallExtensionResponse that notifies the engine that the extension installation should be canceled.

```
public static InstallExtensionResponse Cancel
```

## Returns

InstallExtensionResponse

The InstallExtensionResponse instance that can be used as a return value in InstallExtensionHandler implementation.

# Field Install

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The InstallExtensionResponse that notifies the engine that the extension should be installed.

```
public static InstallExtensionResponse Install
```

## Returns

InstallExtensionResponse

The InstallExtensionResponse instance that can be used as a return value in InstallExtensionHandler implementation.

# Class OpenExtensionPopupParameters

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The parameters of the OpenExtensionPopupHandler.

```
public sealed class OpenExtensionPopupParameters
```

**Inheritance**

object ← OpenExtensionPopupParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Extension

Gets the extension that requests opening the popup.

PopupBrowser

Gets the created popup browser.

# Property Extension

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## Extension

Gets the extension that requests opening the popup.

```
public IExtension Extension { get; }
```

## Property Value

IExtension

# Property PopupBrowser

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## PopupBrowser

Gets the created popup browser.

```
public IBrowser PopupBrowser { get; }
```

## Property Value

IBrowser

# Class UninstallExtensionParameters

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The parameters of the UninstallExtensionHandler.

```
public sealed class UninstallExtensionParameters
```

**Inheritance**

object ← UninstallExtensionParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Extension

Gets the extension that is about to be uninstalled.

# Property Extension

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

## Extension

Gets the extension that is about to be uninstalled.

```
public IExtension Extension { get; }
```

## Property Value

IExtension

# Class UninstallExtensionResponse

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The response to the UninstallExtensionHandler.

```
public sealed class UninstallExtensionResponse
```

**Inheritance**

object ← UninstallExtensionResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Cancel

The UninstallExtensionResponse that notifies the engine that the extension uninstalling should be canceled.

Uninstall

The UninstallExtensionResponse that notifies the engine that the extension should be uninstalled.

# Field Cancel

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The UninstallExtensionResponse that notifies the engine that the extension uninstalling should be canceled.

```
public static UninstallExtensionResponse Cancel
```

## Returns

UninstallExtensionResponse

The UninstallExtensionResponse instance that can be used as a return value in UninstallExtensionHandler implementation.

# Field Uninstall

Namespace: DotNetBrowser.Extensions.Handlers

Assembly: DotNetBrowser.dll

The UninstallExtensionResponse that notifies the engine that the extension should be uninstalled.

```
public static UninstallExtensionResponse Uninstall
```

## Returns

UninstallExtensionResponse

The UninstallExtensionResponse instance that can be used as a return value in UninstallExtensionHandler implementation.

# Namespace DotNetBrowser.Frames

## Classes

EditorCommand

Provides the supported commands that can be executed in a IFrame.

WebStorageException

Thrown by IWebStorage properties and methods to indicate that the requested operation failed. This is a superclass of the exceptions that can be thrown during working with IWebStorage.

WebStorageOverflowException

Thrown by IWebStorage indexer if the new value size exceeds the available space in the storage.

WebStorageSecurityException

Thrown by IWebStorage methods if the access to the web storage is forbidden for the current document.

## Interfaces

IFrame

Represents a frame in the browser. Each web page loaded in the browser has a main(top-level) frame. The frame itself may have child frames. When a web page is unloaded, its frame and all child frames are closed automatically.

IWebStorage

An HTML WebStorage.

Provides access to the session storage or local storage for a particular document on the loaded web page. Allows you to add, modify, or delete the stored items.

# Class EditorCommand

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

Provides the supported commands that can be executed in a IFrame.

```
public sealed class EditorCommand
```

**Inheritance**

object ← EditorCommand

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

BackColor(string)

Returns a command that allows setting the background color for selected text in a WYSIWYG editor.

Bold()

Returns a command that allows making the selected text bold in a WYSIWYG editor.

Copy()

Returns a command that allows copying the selected text in the frame.

Cut()

Returns a command that allows cutting the selected text in a text field, text area or WYSIWYG editor.

Delete()

Returns a command that allows deleting the selected text in a text field, text area or WYSIWYG editor.

DeleteBackward()

Returns a command that allows deleting character before the caret position in a text field, text area or WYSIWYG editor.

DeleteForward()

Returns a command that allows deleting character after the caret position in a text field, text area or WYSIWYG editor.

DeleteToBeginningOfLine()
Returns a command that allows deleting character from the caret position to the beginning of line in a text field, text area or WYSIWYG editor.

DeleteWordBackward()
Returns a command that allows deleting a word before the caret position in a text field, text area or WYSIWYG editor.

DeleteWordForward()
Returns a command that allows deleting a word after the caret position in a text field, text area or WYSIWYG editor.

FindText(string)
Returns a command that allows searching for the given `text` occurrence from the current caret position in the frame. If match found, this command will select it and scroll down to make it visible, if needed.

FontName(string)
Returns a command that allows setting the given `fontName` for the selected text in a WYSIWYG editor.

FontSize(int)
Returns a command that allows setting the given `fontSize` for the selected text in a WYSIWYG editor.

ForeColor(string)
Returns a command that allows setting the given foreground `color` for the selected text in a WYSIWYG editor.

IgnoreSpelling()
Returns a command that allows disabling the spelling mistakes highlighting in a text area or a WYSIWYG editor.

InsertHtml(string)
Returns a command that allows inserting the given `html` content in a WYSIWYG editor.

InsertImage(string)
Returns a command that allows inserting an image with the given `source` in a WYSIWYG editor. If the `source` points to an invalid location, then image won't be inserted.

InsertNewLine()
Returns a command that allows inserting a new line in a WYSIWYG editor.

InsertParagraph()
Inserts new paragraph in a WYSIWYG editor.

InsertTab()
Returns a command that allows inserting a tab character in a WYSIWYG editor.

InsertText(string)
Returns a command that allows inserting the given `text` content in a WYSIWYG editor or a text field.

Italic()
Returns a command that allows making the selected text italic in a WYSIWYG editor.

MovePageDown()
Returns a command that allows moving the caret position at one page down in a text area or a WYSIWYG editor.

MovePageUp()
Returns a command that allows moving the caret position at one page up in a text area or a WYSIWYG editor.

MoveToBeginningOfLine()
Returns a command that allows moving a word to the beginning of line in a text area or a WYSIWYG editor.

MoveToBeginningOfLineAndModifySelection()
Returns a command that allows moving a word to the beginning of line and modifying selection in a text area or a WYSIWYG editor.

MoveToEndOfLine()
Returns a command that allows moving a word to the end of line in a text area or a WYSIWYG editor.

MoveToEndOfLineAndModifySelection()
Returns a command that allows moving a word to the end of line and modifying selection in a text area or a WYSIWYG editor.

MoveWordLeft()
Returns a command that allows moving a word left in a text area or a WYSIWYG editor.

MoveWordLeftAndModifySelection()
Returns a command that allows moving a word left and modifying selection in a text area or a WYSIWYG editor.

MoveWordRight()
Returns a command that allows moving a word right in a text area or a WYSIWYG editor.

MoveWordRightAndModifySelection()
Returns a command that allows moving a word right and modifying selection in a text area or a WYSIWYG editor.

Paste()
Returns a command that allows pasting content of the clipboard in a text field, text area or a WYSIWYG editor.

Redo()
Returns a command that allows reversing the last Undo() action.

ScrollLineDown()
Returns a command that allows scrolling one line down.

ScrollLineUp()
Returns a command that allows scrolling one line up.

ScrollPageBackward()
Returns a command that allows scrolling a page backward.

ScrollPageForward()
Returns a command that allows scrolling a page forward.

ScrollToBeginningOfDocument()
Returns a command that allows scrolling content to the beginning of the document in the frame.

ScrollToEndOfDocument()
Returns a command that allows scrolling content to the end of the document in the frame.

SelectAll()
Returns a command that allows selecting all text in the frame or the currently focused text field, text area or a WYSIWYG editor.

SelectLine()
Returns a command that allows selecting a line at the caret position in a text area or a WYSIWYG editor.

SelectParagraph()
Returns a command that allows selecting the whole paragraph at the caret position in a text area or a WYSIWYG editor.

SelectSentence()
Returns a command that allows selecting the whole sentence at the caret position in a text area or a WYSIWYG editor.

SelectWord()
: Returns a command that allows selecting the word at the caret position in a text field, text area or a WYSIWYG editor.

ToggleBold()
: Returns a command that allows toggling bold style for the selected text in a WYSIWYG editor.

ToggleItalic()
: Returns a command that allows toggling italic style for the selected text in a WYSIWYG editor.

ToggleUnderline()
: Returns a command that allows toggling underline style for the selected text in a WYSIWYG editor.

Underline()
: Returns a command that allows underlying the selected text in a WYSIWYG editor.

Undo()
: Returns a command that allows reversing the last action.

Unselect()
: Returns a command that allows clearing the current selection in a frame.

# Method BackColor

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## BackColor(string)

Returns a command that allows setting the background color for selected text in a WYSIWYG editor.

```
public static EditorCommand BackColor(string color)
```

## Parameters

`color` string

the color to set.

## Returns

EditorCommand

The EditorCommand instance.

## Exceptions

ArgumentException

Thrown when the `color` is null, empty or contain only white space.

# Method Bold

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Bold()

Returns a command that allows making the selected text bold in a WYSIWYG editor.

```
public static EditorCommand Bold()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method Copy

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Copy()

Returns a command that allows copying the selected text in the frame.

```
public static EditorCommand Copy()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method Cut

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Cut()

Returns a command that allows cutting the selected text in a text field, text area or WYSIWYG editor.

```
public static EditorCommand Cut()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method Delete

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Delete()

Returns a command that allows deleting the selected text in a text field, text area or WYSIWYG editor.

```
public static EditorCommand Delete()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method DeleteBackward

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## DeleteBackward()

Returns a command that allows deleting character before the caret position in a text field, text area or WYSIWYG editor.

```
public static EditorCommand DeleteBackward()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method DeleteForward

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## DeleteForward()

Returns a command that allows deleting character after the caret position in a text field, text area or WYSIWYG editor.

```
public static EditorCommand DeleteForward()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method DeleteToBeginningOfLine

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## DeleteToBeginningOfLine()

Returns a command that allows deleting character from the caret position to the beginning of line in a text field, text area or WYSIWYG editor.

```
public static EditorCommand DeleteToBeginningOfLine()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method DeleteWordBackward

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## DeleteWordBackward()

Returns a command that allows deleting a word before the caret position in a text field, text area or WYSIWYG editor.

```
public static EditorCommand DeleteWordBackward()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method DeleteWordForward

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## DeleteWordForward()

Returns a command that allows deleting a word after the caret position in a text field, text area or WYSIWYG editor.

```
public static EditorCommand DeleteWordForward()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method FindText

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## FindText(string)

Returns a command that allows searching for the given `text` occurrence from the current caret position in the frame. If match found, this command will select it and scroll down to make it visible, if needed.

```
public static EditorCommand FindText(string text)
```

## Parameters

`text` string

the text to search.

## Returns

EditorCommand

The EditorCommand instance.

## Exceptions

ArgumentException

Thrown when the `text` is null, empty or contain only white space.

# Method FontName

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## FontName(string)

Returns a command that allows setting the given `fontName` for the selected text in a WYSIWYG editor.

```
public static EditorCommand FontName(string fontName)
```

## Parameters

`fontName` string

the font name to set.

## Returns

EditorCommand

The EditorCommand instance.

## Exceptions

ArgumentException

Thrown when the `fontName` is null, empty or contain only white space.

# Method FontSize

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## FontSize(int)

Returns a command that allows setting the given `fontSize` for the selected text in a WYSIWYG editor.

```
public static EditorCommand FontSize(int fontSize)
```

## Parameters

`fontSize` int

the font size to set.

## Returns

EditorCommand

The EditorCommand instance.

## Exceptions

ArgumentException

Thrown when the `fontSize` is less or equal to zero.

# Method ForeColor

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ForeColor(string)

Returns a command that allows setting the given foreground `color` for the selected text in a WYSIWYG editor.

```
public static EditorCommand ForeColor(string color)
```

### Parameters

`color` string

the color to set.

### Returns

EditorCommand

The EditorCommand instance.

### Exceptions

ArgumentException

Thrown when the `color` is null, empty or contain only white space.

# Method IgnoreSpelling

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## IgnoreSpelling()

Returns a command that allows disabling the spelling mistakes highlighting in a text area or a WYSIWYG editor.

```
public static EditorCommand IgnoreSpelling()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method InsertHtml

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## InsertHtml(string)

Returns a command that allows inserting the given `html` content in a WYSIWYG editor.

```
public static EditorCommand InsertHtml(string html)
```

### Parameters

`html` string

the html content to insert.

### Returns

EditorCommand

The EditorCommand instance.

### Exceptions

ArgumentException

Thrown when the `html` is null, empty or contain only white space.

# Method InsertImage

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## InsertImage(string)

Returns a command that allows inserting an image with the given `source` in a WYSIWYG editor. If the `source` points to an invalid location, then image won't be inserted.

```
public static EditorCommand InsertImage(string source)
```

### Parameters

`source` string

the value of the 'src' attribute of the IMG tag

### Returns

EditorCommand

The EditorCommand instance.

### Exceptions

ArgumentException

Thrown when the `source` is null, empty or contain only white space.

# Method InsertNewLine

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## InsertNewLine()

Returns a command that allows inserting a new line in a WYSIWYG editor.

```
public static EditorCommand InsertNewLine()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method InsertParagraph

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## InsertParagraph()

Inserts new paragraph in a WYSIWYG editor.

```
public static EditorCommand InsertParagraph()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method InsertTab

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## InsertTab()

Returns a command that allows inserting a tab character in a WYSIWYG editor.

```
public static EditorCommand InsertTab()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method InsertText

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## InsertText(string)

Returns a command that allows inserting the given `text` content in a WYSIWYG editor or a text field.

```
public static EditorCommand InsertText(string text)
```

## Parameters

`text` string

the text to insert.

## Returns

EditorCommand

The EditorCommand instance.

## Exceptions

ArgumentException

Thrown when the `text` is null, empty or contain only white space.

# Method Italic

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Italic()

Returns a command that allows making the selected text italic in a WYSIWYG editor.

```
public static EditorCommand Italic()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MovePageDown

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MovePageDown()

Returns a command that allows moving the caret position at one page down in a text area or a WYSIWYG editor.

```
public static EditorCommand MovePageDown()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MovePageUp

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MovePageUp()

Returns a command that allows moving the caret position at one page up in a text area or a WYSIWYG editor.

```
public static EditorCommand MovePageUp()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MoveToBeginningOfLine

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MoveToBeginningOfLine()

Returns a command that allows moving a word to the beginning of line in a text area or a WYSIWYG editor.

```
public static EditorCommand MoveToBeginningOfLine()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MoveToBeginningOfLineAndModify Selection

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MoveToBeginningOfLineAndModifySelection()

Returns a command that allows moving a word to the beginning of line and modifying selection in a text area or a WYSIWYG editor.

```
public static EditorCommand MoveToBeginningOfLineAndModifySelection()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MoveToEndOfLine

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MoveToEndOfLine()

Returns a command that allows moving a word to the end of line in a text area or a WYSIWYG editor.

```
public static EditorCommand MoveToEndOfLine()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MoveToEndOfLineAndModifySelection

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MoveToEndOfLineAndModifySelection()

Returns a command that allows moving a word to the end of line and modifying selection in a text area or a WYSIWYG editor.

```
public static EditorCommand MoveToEndOfLineAndModifySelection()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MoveWordLeft

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MoveWordLeft()

Returns a command that allows moving a word left in a text area or a WYSIWYG editor.

```
public static EditorCommand MoveWordLeft()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MoveWordLeftAndModifySelection

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MoveWordLeftAndModifySelection()

Returns a command that allows moving a word left and modifying selection in a text area or a WYSIWYG editor.

```
public static EditorCommand MoveWordLeftAndModifySelection()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MoveWordRight

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MoveWordRight()

Returns a command that allows moving a word right in a text area or a WYSIWYG editor.

```
public static EditorCommand MoveWordRight()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method MoveWordRightAndModifySelection

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## MoveWordRightAndModifySelection()

Returns a command that allows moving a word right and modifying selection in a text area or a WYSIWYG editor.

```
public static EditorCommand MoveWordRightAndModifySelection()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method Paste

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Paste()

Returns a command that allows pasting content of the clipboard in a text field, text area or a WYSIWYG editor.

```
public static EditorCommand Paste()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method Redo

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Redo()

Returns a command that allows reversing the last Undo() action.

```
public static EditorCommand Redo()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ScrollLineDown

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ScrollLineDown()

Returns a command that allows scrolling one line down.

```
public static EditorCommand ScrollLineDown()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ScrollLineUp

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ScrollLineUp()

Returns a command that allows scrolling one line up.

```
public static EditorCommand ScrollLineUp()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ScrollPageBackward

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ScrollPageBackward()

Returns a command that allows scrolling a page backward.

```
public static EditorCommand ScrollPageBackward()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ScrollPageForward

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ScrollPageForward()

Returns a command that allows scrolling a page forward.

```
public static EditorCommand ScrollPageForward()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ScrollToBeginningOfDocument

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ScrollToBeginningOfDocument()

Returns a command that allows scrolling content to the beginning of the document in the frame.

```
public static EditorCommand ScrollToBeginningOfDocument()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ScrollToEndOfDocument

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ScrollToEndOfDocument()

Returns a command that allows scrolling content to the end of the document in the frame.

```
public static EditorCommand ScrollToEndOfDocument()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method SelectAll

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## SelectAll()

Returns a command that allows selecting all text in the frame or the currently focused text field, text area or a WYSIWYG editor.

```
public static EditorCommand SelectAll()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method SelectLine

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## SelectLine()

Returns a command that allows selecting a line at the caret position in a text area or a WYSIWYG editor.

```
public static EditorCommand SelectLine()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method SelectParagraph

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## SelectParagraph()

Returns a command that allows selecting the whole paragraph at the caret position in a text area or a WYSIWYG editor.

```
public static EditorCommand SelectParagraph()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method SelectSentence

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## SelectSentence()

Returns a command that allows selecting the whole sentence at the caret position in a text area or a WYSIWYG editor.

```
public static EditorCommand SelectSentence()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method SelectWord

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## SelectWord()

Returns a command that allows selecting the word at the caret position in a text field, text area or a WYSIWYG editor.

```
public static EditorCommand SelectWord()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ToggleBold

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ToggleBold()

Returns a command that allows toggling bold style for the selected text in a WYSIWYG editor.

```
public static EditorCommand ToggleBold()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ToggleItalic

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ToggleItalic()

Returns a command that allows toggling italic style for the selected text in a WYSIWYG editor.

```
public static EditorCommand ToggleItalic()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method ToggleUnderline

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ToggleUnderline()

Returns a command that allows toggling underline style for the selected text in a WYSIWYG editor.

```
public static EditorCommand ToggleUnderline()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method Underline

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Underline()

Returns a command that allows underlying the selected text in a WYSIWYG editor.

```
public static EditorCommand Underline()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method Undo

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Undo()

Returns a command that allows reversing the last action.

```
public static EditorCommand Undo()
```

## Returns

EditorCommand

The EditorCommand instance.

# Method Unselect

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Unselect()

Returns a command that allows clearing the current selection in a frame.

```
public static EditorCommand Unselect()
```

## Returns

EditorCommand

The EditorCommand instance.

# Interface IFrame

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

Represents a frame in the browser. Each web page loaded in the browser has a main(top-level) frame. The frame itself may have child frames. When a web page is unloaded, its frame and all child frames are closed automatically.

```
public interface IFrame : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

**Extension Methods**

JsonExtensions.ParseJsonString(IFrame, string) , JsonExtensions.ParseJsonString<T>(IFrame, string)

## Properties

Browser

The IBrowser instance associated with this frame.

Children

The collection of the child frames.

Document

The IDocument that can be used to work with DOM of the frame. Can be null if the frame doesn't have a document.

Html

The HTML content of the frame. Can be an empty string if the frame doesn't have a content or its content is empty.

IsMain

Indicates whether the frame is a main (top-level) frame in the browser.

LocalStorage

The `localStorage` instance of this frame.

Name

The name of the frame. Can be an empty string if the frame doesn't have a name.

Parent

The parent frame.

SelectedHtml

The HTML representation of the selected content in the frame. Can be an empty string if there's no selection.

SelectedText

The text representation of the selected content in the frame. Can be an empty string if there's no selection.

SessionStorage

The `sessionStorage` instance of this frame.

Text

Gets the content of the frame and its sub-frames as plain text or an empty string if the frame does not have content or its content is empty.

## Methods

CreateJsArrayBuffer(byte[])

Creates a new IJsArrayBuffer and copies the contents of the byte array to it.

CreateJsArray<T>(IEnumerable<T>)

Creates a new IJsArray and copies the contents of the IEnumerable<T> to it.

CreateJsMap<TKey, TValue>(IDictionary<TKey, TValue>)

Creates a new IJsMap and copies the contents of the IDictionary<TKey, TValue> to it.

CreateJsMap<TKey, TValue>(IReadOnlyDictionary<TKey, TValue>)

Creates a new IJsMap and copies the contents of the IReadOnlyDictionary<TKey, TValue> to it.

CreateJsSet<T>(IEnumerable<T>)

Creates a new IJsSet and copies the contents of the IEnumerable<T> to it.

Execute(EditorCommand)

Executes the given `command` in the frame. Before executing the command, it's recommended to check whether it can be executed or not using the IsCommandEnabled(EditorCommand) method.

ExecuteJavaScript(string, bool)

Asynchronously executes JavaScript code using the current context.

ExecuteJavaScript<T>(string, bool)

Asynchronously executes JavaScript code using the current context and converts the execution result to the specified .NET type.

Inspect(Point)

Inspects the given `location` in the frame and returns the result of inspection.

Inspect(int, int)

Inspects the given location in the frame and returns the result of inspection.

IsCommandEnabled(EditorCommand)

Indicates if the command with the given `command` can be executed in the frame. Some commands can be executed only under certain conditions. For example, the InsertText(string) command can be executed only if there's a focused text field in the frame.

LoadUrl(string)

Navigates the frame to a resource identified by the given `url`.

LoadUrl(string, TimeSpan)

Navigates the frame to a resource identified by the given `url`.

Print()

Requests printing of the currently loaded web page in this frame.

ViewSource()

Opens a popup with the frame's source like the browser does when showing page source.

# Property Browser

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Browser

The IBrowser instance associated with this frame.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

# Property Children

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Children

The collection of the child frames.

```
IEnumerable<IFrame> Children { get; }
```

## Property Value

IEnumerable<IFrame>

# Property Document

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Document

The IDocument that can be used to work with DOM of the frame. Can be null if the frame doesn't have a document.

```
IDocument Document { get; }
```

## Property Value

IDocument

# Property Html

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Html

The HTML content of the frame. Can be an empty string if the frame doesn't have a content or its content is empty.

```
string Html { get; }
```

## Property Value

string

# Property IsMain

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## IsMain

Indicates whether the frame is a main (top-level) frame in the browser.

```
bool IsMain { get; }
```

## Property Value

bool

# Property LocalStorage

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## LocalStorage

The `localStorage` instance of this frame.

```
IWebStorage LocalStorage { get; }
```

## Property Value

IWebStorage

# Property Name

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Name

The name of the frame. Can be an empty string if the frame doesn't have a name.

```
string Name { get; }
```

## Property Value

string

# Property Parent

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Parent

The parent frame.

```
IFrame Parent { get; }
```

## Property Value

IFrame

# Property SelectedHtml

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## SelectedHtml

The HTML representation of the selected content in the frame. Can be an empty string if there's no selection.

```
string SelectedHtml { get; }
```

## Property Value

string

# Property SelectedText

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## SelectedText

The text representation of the selected content in the frame. Can be an empty string if there's no selection.

```
string SelectedText { get; }
```

## Property Value

string

# Property SessionStorage

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## SessionStorage

The `sessionStorage` instance of this frame.

```
IWebStorage SessionStorage { get; }
```

## Property Value

IWebStorage

# Property Text

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Text

Gets the content of the frame and its sub-frames as plain text or an empty string if the frame does not have content or its content is empty.

```
string Text { get; }
```

## Property Value

string

The plain text content of the frame.

## Remarks

Limitations:

If the frame has sub-frames from another domain, their text will not be included into the return value.

No specification is given for how text from one frame will be delimited from the next.

The return value may contain text that is not visible to the user, whether due to scrolling, `display:none`, unexpanded divs, etc.

The ordering of the text does not reflect the ordering of the text on the page as seen by the user. Each element's text may appear in a totally different position in the result from its position on the page.

Please note that this functionality is resource-intensive, consuming significant memory and CPU during a text capture.

# Method CreateJsArray

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## CreateJsArray<T>(IEnumerable<T>)

Creates a new IJsArray and copies the contents of the IEnumerable<T> to it.

```
IJsArray CreateJsArray<T>(IEnumerable<T> enumerable)
```

## Parameters

`enumerable` IEnumerable<T>

The enumerable to copy values from.

## Returns

IJsArray

An IJsArray that contains values from the IEnumerable<T>.

## Type Parameters

`T`

## Remarks

The values are converted to their JavaScript representations during copying.

## Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

# Method CreateJsArrayBuffer

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## CreateJsArrayBuffer(byte[])

Creates a new IJsArrayBuffer and copies the contents of the byte array to it.

```
IJsArrayBuffer CreateJsArrayBuffer(byte[] data)
```

## Parameters

`data` byte[]

The byte array to copy data from.

## Returns

IJsArrayBuffer

An IJsArrayBuffer that contains values from the byte array.

## Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

# Method CreateJsMap

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## CreateJsMap<TKey, TValue>(IDictionary<TKey, TValue>)

Creates a new IJsMap and copies the contents of the IDictionary<TKey, TValue> to it.

```
IJsMap CreateJsMap<TKey, TValue>(IDictionary<TKey, TValue> dictionary)
```

### Parameters

`dictionary` IDictionary<TKey, TValue>

The dictionary to copy elements from.

### Returns

IJsMap

An IJsMap that contains values from the provided dictionary.

### Type Parameters

`TKey`

`TValue`

### Remarks

The values are converted to their JavaScript representations during copying.

### Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

## CreateJsMap<TKey, TValue>(IReadOnlyDictionary<TKey, TValue>)

Creates a new IJsMap and copies the contents of the IReadOnlyDictionary<TKey, TValue> to it.

```
IJsMap CreateJsMap<TKey, TValue>(IReadOnlyDictionary<TKey, TValue> dictionary)
```

### Parameters

`dictionary` IReadOnlyDictionary<TKey, TValue>

The dictionary to copy elements from.

### Returns

IJsMap

An IJsMap that contains values from the provided dictionary.

### Type Parameters

`TKey`

`TValue`

### Remarks

The values are converted to their JavaScript representations during copying.

### Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

# Method CreateJsSet

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## CreateJsSet<T>(IEnumerable<T>)

Creates a new IJsSet and copies the contents of the IEnumerable<T> to it.

```
IJsSet CreateJsSet<T>(IEnumerable<T> collection)
```

## Parameters

`collection` IEnumerable<T>

The collection to copy elements from.

## Returns

IJsSet

An IJsSet that contains values from the provided collection.

## Type Parameters

`T`

## Remarks

The values are converted to their JavaScript representations during copying.

## Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

# Method Execute

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Execute(EditorCommand)

Executes the given `command` in the frame. Before executing the command, it's recommended to check whether it can be executed or not using the IsCommandEnabled(EditorCommand) method.

```
bool Execute(EditorCommand command)
```

### Parameters

`command` EditorCommand

The command to execute. Cannot be null.

### Returns

bool

true if the command has been executed successfully.

### Exceptions

ArgumentNullException

The `command` is null.

ObjectDisposedException

The IFrame object has already been disposed.

# Method ExecuteJavaScript

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ExecuteJavaScript<T>(string, bool)

Asynchronously executes JavaScript code using the current context and converts the execution result to the specified .NET type.

```
Task<T> ExecuteJavaScript<T>(string javaScript, bool userGesture = false)
```

### Parameters

`javaScript` string

The JavaScript code.

`userGesture` bool

Indicates whether JavaScript is initiated by a user gesture.

### Returns

Task<T>

The task that can be used to wait for completion and obtain the result of the JavaScript execution.

### Type Parameters

`T`

.NET type of the result

### Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

ArgumentException

The `javaScript` is null, empty, or contains only blank characters.

## ExecuteJavaScript(string, bool)

Asynchronously executes JavaScript code using the current context.

```
Task<object> ExecuteJavaScript(string javaScript, bool userGesture = false)
```

### Parameters

`javaScript` string

The JavaScript code to execute.

`userGesture` bool

Indicates whether JavaScript is initiated by a user gesture.

### Returns

Task<object>

The task that can be used to wait for completion and obtain the result of the JavaScript execution.

### Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

ArgumentException

The `javaScript` is null, empty, or contains only blank characters.

# Method Inspect

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Inspect(Point)

Inspects the given `location` in the frame and returns the result of inspection.

```
PointInspection Inspect(Point location)
```

### Parameters

`location` Point

A point in the frame's view. Cannot be null.

### Returns

PointInspection

The result of inspection that contains the details about the DOM node at the given location.

### Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

ArgumentException

The `location` is null.

## Inspect(int, int)

Inspects the given location in the frame and returns the result of inspection.

```
PointInspection Inspect(int x, int y)
```

## Parameters

x int

a horizontal coordinate on the web page

y int

a vertical coordinate on the web page

## Returns

PointInspection

The result of inspection that contains the details about the DOM node at the given location.

## Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

# Method IsCommandEnabled

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## IsCommandEnabled(EditorCommand)

Indicates if the command with the given `command` can be executed in the frame. Some commands can be executed only under certain conditions. For example, the InsertText(string) command can be executed only if there's a focused text field in the frame.

```
bool IsCommandEnabled(EditorCommand command)
```

### Parameters

`command` EditorCommand

The command to check.

### Returns

bool

`true` if the command is enabled and can be executed.

### Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

ArgumentException

The `command` is null.

# Method LoadUrl

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## LoadUrl(string)

Navigates the frame to a resource identified by the given `url`.

```
Task<NavigationResult> LoadUrl(string url)
```

### Parameters

`url` string

The URL of the resource to load. Cannot be null, empty or contain only white space.

### Returns

Task<NavigationResult>

A task that represents the asynchronous loading operation. The NavigationResult indicates if the loading operation has been completed, stopped or failed. The task throws a TimeoutException if the loading operation hasn't been completed within a default timeout (100 seconds).

### Exceptions

ArgumentException

thrown when `url` is null, empty or contain only white space.

ObjectDisposedException

The IFrame object has already been disposed.

## LoadUrl(string, TimeSpan)

Navigates the frame to a resource identified by the given `url`.

```
Task<NavigationResult> LoadUrl(string url, TimeSpan timeout)
```

## Parameters

`url` string

The URL of the resource to load. Cannot be null, empty or contain only white space.

`timeout` TimeSpan

The timeout.

## Returns

Task<NavigationResult>

A task that represents the asynchronous loading operation. The NavigationResult indicates if the loading operation has been completed, stopped or failed. The task throws a TimeoutException if the loading operation hasn't been completed within the specified `timeout`.

## Exceptions

ArgumentException

thrown when `url` is null, empty or contain only white space.

ObjectDisposedException

The IFrame object has already been disposed.

# Method Print

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Print()

Requests printing of the currently loaded web page in this frame.

```
void Print()
```

## Remarks

The printing should be allowed in the RequestPrintHandler, in other case, it will be canceled.

## Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

InvalidOperationException

The print operation failed. This can happen when the frame itself does not exist anymore.

# Method ViewSource

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## ViewSource()

Opens a popup with the frame's source like the browser does when showing page source.

```
void ViewSource()
```

## Exceptions

ObjectDisposedException

The IFrame object has already been disposed.

# Interface IWebStorage

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

An HTML WebStorage.

Provides access to the session storage or local storage for a particular document on the loaded web page. Allows you to add, modify, or delete the stored items.

```
public interface IWebStorage : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Count
Gets the number of the key/value pairs in the storage.

this[string]
Gets or sets the value associated with the given `key`. The setter adds the item with the specified `key` and `value` to the storage, or updates it if the item with the given `key` already exists.

Keys
Gets a list of the web storage keys.

## Methods

Clear()
Removes all the items from the storage.

Contains(string)
Checks if the specified key is present in the storage.

Remove(string)

Removes the item with the specified `key` from the storage. Does nothing if there is no item with the specified `key` in the storage.

# Property Count

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Count

Gets the number of the key/value pairs in the storage.

```
int Count { get; }
```

## Property Value

int

The number of the key/value pairs in the storage.

## Exceptions

ObjectDisposedException

The IWebStorage object has already been disposed.

WebStorageException

The IWebStorage operation has failed.

# Property Keys

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Keys

Gets a list of the web storage keys.

```
IReadOnlyList<string> Keys { get; }
```

## Property Value

IReadOnlyList<string>

A readonly list of the web storage keys.

## Exceptions

ObjectDisposedException

The IWebStorage object has already been disposed.

# Property this

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## this[string]

Gets or sets the value associated with the given `key`. The setter adds the item with the specified `key` and `value` to the storage, or updates it if the item with the given `key` already exists.

```
string this[string key] { get; set; }
```

## Parameters

`key` string

The key name of the item to retrieve the value of. Can be empty or blank.

## Property Value

string

The value associated with the given `key`

## Exceptions

ObjectDisposedException

The IWebStorage object has already been disposed.

WebStorageException

The IWebStorage operation has failed.

# Method Clear

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Clear()

Removes all the items from the storage.

```
void Clear()
```

## Exceptions

ObjectDisposedException

The IWebStorage object has already been disposed.

WebStorageException

The IWebStorage operation has failed.

# Method Contains

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Contains(string)

Checks if the specified key is present in the storage.

```
bool Contains(string key)
```

## Parameters

`key` string

the key name to check. Can be empty or blank.

## Returns

bool

`true` if the storage contains an item with the specified `key`, otherwise `false`.

## Exceptions

ObjectDisposedException

The IWebStorage object has already been disposed.

WebStorageException

The IWebStorage operation has failed.

# Method Remove

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## Remove(string)

Removes the item with the specified `key` from the storage. Does nothing if there is no item with the specified `key` in the storage.

```
void Remove(string key)
```

## Parameters

`key` string

the key name of the item to remove. Can be empty or blank.

## Exceptions

ObjectDisposedException

The IWebStorage object has already been disposed.

WebStorageException

The IWebStorage operation has failed.

# Class WebStorageException

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

Thrown by IWebStorage properties and methods to indicate that the requested operation failed. This is a superclass of the exceptions that can be thrown during working with IWebStorage.

```
public class WebStorageException : Exception, ISerializable
```

**Inheritance**

object ← Exception ← WebStorageException

**Implements**

ISerializable

**Derived**

WebStorageOverflowException, WebStorageSecurityException

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

WebStorageException(string)

# Constructor WebStorageException

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## WebStorageException(string)

```
public WebStorageException(string message)
```

## Parameters

`message` string

# Class WebStorageOverflowException

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

Thrown by IWebStorage indexer if the new value size exceeds the available space in the storage.

```
public class WebStorageOverflowException : WebStorageException, ISerializable
```

**Inheritance**

object ← Exception ← WebStorageException ← WebStorageOverflowException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

WebStorageOverflowException(string)

# Constructor WebStorageOverflowException

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## WebStorageOverflowException(string)

```
public WebStorageOverflowException(string message)
```

## Parameters

`message` string

# Class WebStorageSecurityException

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

Thrown by IWebStorage methods if the access to the web storage is forbidden for the current document.

```
public class WebStorageSecurityException : WebStorageException, ISerializable
```

**Inheritance**

object ← Exception ← WebStorageException ← WebStorageSecurityException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

WebStorageSecurityException(string)

# Constructor WebStorageSecurityException

Namespace: DotNetBrowser.Frames

Assembly: DotNetBrowser.dll

## WebStorageSecurityException(string)

```
public WebStorageSecurityException(string message)
```

## Parameters

`message` string

# Namespace DotNetBrowser.Frames.Handlers

## Classes

ConvertJsNameParameters

The parameters of the ConvertJsNameHandler.

ConvertJsNameResponse

The response to the ConvertJsNameHandler.

# Class ConvertJsNameParameters

Namespace: DotNetBrowser.Frames.Handlers

Assembly: DotNetBrowser.dll

The parameters of the ConvertJsNameHandler.

```
public sealed class ConvertJsNameParameters : FrameParameters
```

**Inheritance**

object ← BrowserParameters ← FrameParameters ← ConvertJsNameParameters

**Inherited Members**

FrameParameters.Frame , BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

MemberInfo

Gets the MemberInfo of the .NET object property, field or method.

# Property MemberInfo

Namespace: DotNetBrowser.Frames.Handlers

Assembly: DotNetBrowser.dll

## MemberInfo

Gets the MemberInfo of the .NET object property, field or method.

```
public MemberInfo MemberInfo { get; }
```

## Property Value

MemberInfo

the MemberInfo of the .NET object property, field, or method.

# Class ConvertJsNameResponse

Namespace: DotNetBrowser.Frames.Handlers

Assembly: DotNetBrowser.dll

The response to the ConvertJsNameHandler.

```
public sealed class ConvertJsNameResponse
```

**Inheritance**

object ← ConvertJsNameResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CamelCase

Creates a ConvertJsNameResponse that converts the name to CamelCase (the first letter of the name is lower case).

NoConversion

Creates a ConvertJsNameResponse that use the member name as is.

PascalCase

Creates a ConvertJsNameResponse that converts the name to PascalCase (the first letter of the name is upper case).

## Methods

ConvertTo(string)

Creates a ConvertJsNameResponse that sets the custom name for binding or executing JavaScript.

# Property CamelCase

Namespace: DotNetBrowser.Frames.Handlers

Assembly: DotNetBrowser.dll

## CamelCase

Creates a ConvertJsNameResponse that converts the name to CamelCase (the first letter of the name is lower case).

```
public static ConvertJsNameResponse CamelCase { get; }
```

## Property Value

ConvertJsNameResponse

The ConvertJsNameResponse instance that can be used as a return value in ConvertJsNameHandler implementation.

# Property NoConversion

Namespace: DotNetBrowser.Frames.Handlers

Assembly: DotNetBrowser.dll

## NoConversion

Creates a ConvertJsNameResponse that use the member name as is.

```
public static ConvertJsNameResponse NoConversion { get; }
```

## Property Value

ConvertJsNameResponse

The ConvertJsNameResponse instance that can be used as a return value in ConvertJsNameHandler implementation.

# Property PascalCase

Namespace: DotNetBrowser.Frames.Handlers

Assembly: DotNetBrowser.dll

## PascalCase

Creates a ConvertJsNameResponse that converts the name to PascalCase (the first letter of the name is upper case).

```
public static ConvertJsNameResponse PascalCase { get; }
```

## Property Value

ConvertJsNameResponse

The ConvertJsNameResponse instance that can be used as a return value in ConvertJsNameHandler implementation.

# Method ConvertTo

Namespace: DotNetBrowser.Frames.Handlers

Assembly: DotNetBrowser.dll

## ConvertTo(string)

Creates a ConvertJsNameResponse that sets the custom name for binding or executing JavaScript.

```
public static ConvertJsNameResponse ConvertTo(string customName)
```

## Parameters

`customName` string

The custom name of the JavaScript property, field or method.

## Returns

ConvertJsNameResponse

The ConvertJsNameResponse instance that can be used as a return value in ConvertJsNameHandler implementation.

## Exceptions

ArgumentException

The `customName` is null, empty, or contains only blank characters.

# Namespace DotNetBrowser.Geometry

## Classes

Ellipse

Represents a three numbers that are used to define ellipse size and orientation in the two-dimensional space.

Point

Represents a pair of numbers that in general are used to define coordinates in the two-dimensional space.

Rectangle

Represents a rectangle described by the location and dimensions.

Size

Represents a pair of numbers that in general are used to define dimensions in the two-dimensional space.

# Class Ellipse

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

Represents a three numbers that are used to define ellipse size and orientation in the two-dimensional space.

```
public sealed class Ellipse
```

**Inheritance**

object ← Ellipse

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

Ellipse(float, float, float)

Initializes a new instance of the Ellipse class with the specified `radiusX` and `radiusY` radii.

## Properties

RadiusX

Gets radius X axis length.

RadiusY

Gets radius Y axis length.

RotationAngle

Gets ellipse orientation angle.

## Methods

Equals(object)

GetHashCode()

# Constructor Ellipse

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Ellipse(float, float, float)

Initializes a new instance of the Ellipse class with the specified `radiusX` and `radiusY` radii.

```
public Ellipse(float radiusX, float radiusY, float rotationAngle = 0)
```

## Parameters

`radiusX` float

Radius X axis size.

`radiusY` float

Radius X axis size.

`rotationAngle` float

Ellipse orientation angle.

# Property RadiusX

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## RadiusX

Gets radius X axis length.

```
public float RadiusX { get; }
```

## Property Value

float

# Property RadiusY

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## RadiusY

Gets radius Y axis length.

```
public float RadiusY { get; }
```

## Property Value

float

# Property RotationAngle

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## RotationAngle

Gets ellipse orientation angle.

```
public float RotationAngle { get; }
```

## Property Value

float

# Method Equals

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class Point

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

Represents a pair of numbers that in general are used to define coordinates in the two-dimensional space.

```
public sealed class Point
```

**Inheritance**

object ← Point

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

## Constructors

Point(int, int)

Initializes a new instance of the Point class with the specified `x` and `y` coordinates.

## Fields

Empty

An empty `Point` that has x and y values set to zero.

## Properties

X

Gets the horizontal coordinate.

Y

Gets the vertical coordinate.

## Methods

Equals(object)

GetHashCode()

ToString()

# Constructor Point

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Point(int, int)

Initializes a new instance of the Point class with the specified `x` and `y` coordinates.

```
public Point(int x, int y)
```

## Parameters

`x` int

The horizontal coordinate.

`y` int

The vertical coordinate.

# Field Empty

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

An empty `Point` that has x and y values set to zero.

```
public static readonly Point Empty
```

## Returns

Point

An empty Point that has x and y values set to zero.

# Property X

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## X

Gets the horizontal coordinate.

```
public int X { get; }
```

### Property Value

int

The horizontal coordinate.

# Property Y

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Y

Gets the vertical coordinate.

```
public int Y { get; }
```

## Property Value

int

The vertical coordinate.

# Method Equals

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Method ToString

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class Rectangle

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

Represents a rectangle described by the location and dimensions.

```
public sealed class Rectangle
```

**Inheritance**

object ← Rectangle

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

## Constructors

Rectangle(Point, Size)

Initializes a new instance of the Rectangle class with the specified `origin` and `size`.

Rectangle(int, int, uint, uint)

Initializes a new instance of the Rectangle class with the specified parameters.

## Properties

IsEmpty

Indicates whether all numeric properties of this Rectangle have values of zero.

Origin

Gets the top-left corner coordinate of the rectangle.

Size

Gets the rectangle dimensions.

## Methods

Center()
Calculates the center of this rectangle.

Contains(Point)
Checks whether the rectangle contains the specified point.

Contains(Rectangle)
Checks whether the rectangle contains the specified rectangle.

Equals(object)

GetHashCode()

ToString()

# Constructor Rectangle

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Rectangle(Point, Size)

Initializes a new instance of the Rectangle class with the specified `origin` and `size`.

```
public Rectangle(Point origin, Size size)
```

### Parameters

`origin` Point

The top-left corner coordinate of the rectangle.

`size` Size

The rectangle dimensions.

## Rectangle(int, int, uint, uint)

Initializes a new instance of the Rectangle class with the specified parameters.

```
public Rectangle(int x, int y, uint width, uint height)
```

### Parameters

`x` int

The top-left corner X coordinate of the rectangle.

`y` int

The top-left corner Y coordinate of the rectangle.

`width` uint

The rectangle width.

`height` uint

The rectangle height.

# Property IsEmpty

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## IsEmpty

Indicates whether all numeric properties of this Rectangle have values of zero.

```
public bool IsEmpty { get; }
```

## Property Value

bool

`true` if all numeric properties of this Rectangle have values of zero.

# Property Origin

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Origin

Gets the top-left corner coordinate of the rectangle.

```
public Point Origin { get; }
```

## Property Value

Point

The top-left corner coordinate of the rectangle.

# Property Size

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Size

Gets the rectangle dimensions.

```
public Size Size { get; }
```

## Property Value

Size

The rectangle dimensions.

# Method Center

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Center()

Calculates the center of this rectangle.

```
public Point Center()
```

## Returns

Point

The point that represents the center of this rectangle.

# Method Contains

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Contains(Point)

Checks whether the rectangle contains the specified point.

```
public bool Contains(Point point)
```

### Parameters

`point` Point

the Point instance to check.

### Returns

bool

true if the rectangle contains this point; false otherwise.

## Contains(Rectangle)

Checks whether the rectangle contains the specified rectangle.

```
public bool Contains(Rectangle other)
```

### Parameters

`other` Rectangle

the Rectangle instance to check.

## Returns

bool

true if the rectangle contains the other rectangle; false otherwise.

# Method Equals

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Method ToString

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class Size

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

Represents a pair of numbers that in general are used to define dimensions in the two-dimensional space.

```
public class Size
```

**Inheritance**

object ← Size

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

## Constructors

Size(uint, uint)

Initializes a new instance of the Size class with the specified `width` and `height`.

## Fields

Empty

An empty `Size` that has x and y values set to zero.

## Properties

Height

Gets the vertical dimension.

IsEmpty

Indicates whether width and height of this Size have values of zero.

Width

Gets the horizontal dimension.

## Methods

Equals(Size)

Determines whether the specified `Size` object is equal to the current object.

Equals(object)

GetHashCode()

ToString()

# Constructor Size

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Size(uint, uint)

Initializes a new instance of the Size class with the specified `width` and `height`.

```
public Size(uint width, uint height)
```

## Parameters

`width` uint

the horizontal dimension. Cannot be negative.

`height` uint

the vertical dimension. Cannot be negative.

# Field Empty

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

An empty `Size` that has x and y values set to zero.

```
public static readonly Size Empty
```

## Returns

Size

An empty Size that has x and y values set to zero.

# Property Height

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Height

Gets the vertical dimension.

```
public uint Height { get; }
```

## Property Value

uint

The vertical dimension.

# Property IsEmpty

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## IsEmpty

Indicates whether width and height of this Size have values of zero.

```
public bool IsEmpty { get; }
```

## Property Value

bool

`true` if width and height of this Size have values of zero.

# Property Width

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Width

Gets the horizontal dimension.

```
public uint Width { get; }
```

## Property Value

uint

The horizontal dimension.

# Method Equals

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

## Equals(Size)

Determines whether the specified `Size` object is equal to the current object.

```
protected bool Equals(Size other)
```

### Parameters

`other` Size

The `Size` object to compare with the current object.

### Returns

bool

true if the specified object is equal to the current object; otherwise, false.

# Method GetHashCode

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Method ToString

Namespace: DotNetBrowser.Geometry

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Namespace DotNetBrowser.Handlers

## Classes

AsyncHandler<TParameters, TResponse>
: The default implementation of the IHandler<TParameters, TResponse> interface.

Handler<TParameters>
: The default implementation of the IHandler<TParameters> interface.

Handler<TParameters, TResponse>
: The default implementation of the IHandler<TParameters, TResponse> interface.

## Interfaces

IHandler<TParameters>
: The common interface for all handlers that must be executed synchronously, but do not have to return a value.

IHandler<TParameters, TResponse>
: The common interface for all handlers.

# Class AsyncHandler<TParameters, TResponse>

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

The default implementation of the IHandler<TParameters, TResponse> interface.

```
public sealed class AsyncHandler<TParameters, TResponse> : IHandler<TParameters, TResponse>
```

## Type Parameters

`TParameters`

The handler parameters type.

`TResponse`

The result type.

**Inheritance**

object ← AsyncHandler<TParameters, TResponse>

**Implements**

IHandler<TParameters, TResponse>

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

AsyncHandler(Func<TParameters, Task<TResponse>>)

Initializes an instance of the AsyncHandler<TParameters, TResponse> class.

## Methods

Handle(TParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

# Constructor AsyncHandler

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## AsyncHandler(Func<TParameters, Task<TResponse>>)

Initializes an instance of the AsyncHandler<TParameters, TResponse> class.

```
public AsyncHandler(Func<TParameters, Task<TResponse>> func)
```

## Parameters

`func` Func<TParameters, Task<TResponse>>

The function that must be executed when the Handle(TParameters) method is called.

## Exceptions

ArgumentException

The `func` is null.

# Method Handle

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## Handle(TParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

```
public TResponse Handle(TParameters parameters)
```

## Parameters

`parameters` TParameters

The handler parameters.

## Returns

TResponse

An object that represents the response that should be determined based on the provided parameters.

# Class Handler<TParameters>

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

The default implementation of the IHandler<TParameters> interface.

```
public sealed class Handler<TParameters> : IHandler<TParameters>
```

## Type Parameters

`TParameters`

The handler parameters type.

**Inheritance**

object ← Handler<TParameters>

**Implements**

IHandler<TParameters>

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

Handler(Action<TParameters>)

Initializes an instance of the Handler<TParameters> class.

## Methods

Handle(TParameters)

This method is called when the Chromium callback should be handled synchronously. In a number of cases, the Chromium engine will be blocked until this method returns.

## Operators

implicit operator Handler<TParameters>(Action<TParameters>)

The cast operator that can be used to cast Action<T> to Handler<TParameters>

# Constructor Handler

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## Handler(Action<TParameters>)

Initializes an instance of the Handler<TParameters> class.

```
public Handler(Action<TParameters> func)
```

## Parameters

`func` Action<TParameters>

The function that must be executed when the Handle(TParameters) method is called.

## Exceptions

ArgumentException

The `func` is null.

# Method Handle

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## Handle(TParameters)

This method is called when the Chromium callback should be handled synchronously. In a number of cases, the Chromium engine will be blocked until this method returns.

```
public void Handle(TParameters parameters)
```

## Parameters

`parameters` TParameters

The handler parameters.

# Operator implicit operator

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## implicit operator Handler<TParameters>(Action<TParameters>)

The cast operator that can be used to cast Action<T> to Handler<TParameters>

```
public static implicit operator Handler<TParameters>(Action<TParameters> value)
```

### Parameters

`value` Action<TParameters>

The Action<T> to cast from.

### Returns

Handler<TParameters>

The corresponding Handler<TParameters> instance.

### Exceptions

ArgumentException

The `value` is null.

# Class Handler<TParameters, TResponse>

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

The default implementation of the IHandler<TParameters, TResponse> interface.

```
public sealed class Handler<TParameters, TResponse> : IHandler<TParameters, TResponse>
```

## Type Parameters

`TParameters`

The handler parameters type.

`TResponse`

The response type.

**Inheritance**

object ← Handler<TParameters, TResponse>

**Implements**

IHandler<TParameters, TResponse>

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

Handler(Func<TParameters, TResponse>)

Initializes an instance of the Handler<TParameters, TResponse> class.

## Methods

Handle(TParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

## Operators

implicit operator Handler<TParameters, TResponse>(Func<TParameters, TResponse>)

The cast operator that can be used to cast Func<T, TResult> to Handler<TParameters, TResponse>

# Constructor Handler

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## Handler(Func<TParameters, TResponse>)

Initializes an instance of the Handler<TParameters, TResponse> class.

```
public Handler(Func<TParameters, TResponse> func)
```

## Parameters

`func` Func<TParameters, TResponse>

The function that must be executed when the Handle(TParameters) method is called.

## Exceptions

ArgumentException

The `func` is null.

# Method Handle

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## Handle(TParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

```
public TResponse Handle(TParameters parameters)
```

## Parameters

`parameters` TParameters

The handler parameters.

## Returns

TResponse

An object that represents the response that should be determined based on the provided parameters.

# Operator implicit operator

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## implicit operator Handler<TParameters, TResponse>(Func<TParameters, TResponse>)

The cast operator that can be used to cast Func<T, TResult> to Handler<TParameters, TResponse>

```
public static implicit operator Handler<TParameters, TResponse>(Func<TParameters,
TResponse> func)
```

### Parameters

`func` Func<TParameters, TResponse>

The Func<T, TResult> to cast from.

### Returns

Handler<TParameters, TResponse>

The corresponding Handler<TParameters, TResponse> instance.

### Exceptions

ArgumentException

The `func` is null.

# Interface IHandler<TParameters>

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

The common interface for all handlers that must be executed synchronously, but do not have to return a value.

```
public interface IHandler<in TParameters>
```

## Type Parameters

`TParameters`

The handler parameters type.

## Methods

Handle(TParameters)

This method is called when the Chromium callback should be handled synchronously. In a number of cases, the Chromium engine will be blocked until this method returns.

# Method Handle

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## Handle(TParameters)

This method is called when the Chromium callback should be handled synchronously. In a number of cases, the Chromium engine will be blocked until this method returns.

```
void Handle(TParameters parameters)
```

## Parameters

`parameters` TParameters

The handler parameters.

# Interface IHandler<TParameters, TResponse>

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

The common interface for all handlers.

```
public interface IHandler<in TParameters, out TResponse>
```

## Type Parameters

`TParameters`

The handler parameters type.

`TResponse`

The result type.

## Methods

Handle(TParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

# Method Handle

Namespace: DotNetBrowser.Handlers

Assembly: DotNetBrowser.dll

## Handle(TParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

```
TResponse Handle(TParameters parameters)
```

## Parameters

`parameters` TParameters

The handler parameters.

## Returns

TResponse

An object that represents the response that should be determined based on the provided parameters.

# Namespace DotNetBrowser.Input

## Interfaces

IInputEvent<TIInputEventArgs, TInputEventArgs>
: An input event. It is possible to raise this input event and dispatch it to the currently loaded web page.

IInterceptableEvent<TIInputEventArgs>
: An interceptable input event. It is possible to intercept and suppress this input event before it is actually processed by the browser.

IInterceptableInputEvent<TIInputEventArgs, TInputEventArgs>
: An interceptable and raisable input event. It is possible to intercept and suppress this input event before it is actually processed by the browser or to raise it and dispatch to the currently loaded web page.

## Enums

InputEventResponse
: The response to the input event handlers.

# Interface IInputEvent<TIInputEventArgs, TInputEventArgs>

Namespace: DotNetBrowser.Input

Assembly: DotNetBrowser.dll

An input event. It is possible to raise this input event and dispatch it to the currently loaded web page.

```
public interface IInputEvent<TIInputEventArgs, in TInputEventArgs> where TInputEventArgs
: TIInputEventArgs
```

## Type Parameters

`TIInputEventArgs`

The input event arguments interface.

`TInputEventArgs`

The input event arguments.

## Methods

Raise(TInputEventArgs)

Raises and dispatches the event to the currently loaded web page. It does nothing if the browser hasn't loaded any web page.

# Method Raise

Namespace: DotNetBrowser.Input

Assembly: DotNetBrowser.dll

## Raise(TInputEventArgs)

Raises and dispatches the event to the currently loaded web page. It does nothing if the browser hasn't loaded any web page.

```
void Raise(TInputEventArgs args)
```

### Parameters

`args` TInputEventArgs

The event arguments.

### Exceptions

ObjectDisposedException

The corresponding browser object has already been disposed.

# Interface IInterceptableEvent<TIInputEventArgs>

Namespace: DotNetBrowser.Input

Assembly: DotNetBrowser.dll

An interceptable input event. It is possible to intercept and suppress this input event before it is actually processed by the browser.

```
public interface IInterceptableEvent<TIInputEventArgs>
```

## Type Parameters

`TIInputEventArgs`

The input event arguments interface.

## Properties

Handler

Gets or sets the event handler that can be used to intercept and/or suppress this input event.

# Property Handler

Namespace: DotNetBrowser.Input

Assembly: DotNetBrowser.dll

## Handler

Gets or sets the event handler that can be used to intercept and/or suppress this input event.

```
IHandler<TIInputEventArgs, InputEventResponse> Handler { get; set; }
```

## Property Value

IHandler<TIInputEventArgs, InputEventResponse>

The handler that will be used to intercept and/or suppress this input event.

# Interface IInterceptableInputEvent<TIInputEventArgs, TInputEventArgs>

Namespace: DotNetBrowser.Input

Assembly: DotNetBrowser.dll

An interceptable and raisable input event. It is possible to intercept and suppress this input event before it is actually processed by the browser or to raise it and dispatch to the currently loaded web page.

```
public interface IInterceptableInputEvent<TIInputEventArgs, in TInputEventArgs> :
IInterceptableEvent<TIInputEventArgs>, IInputEvent<TIInputEventArgs, TInputEventArgs> where
TInputEventArgs : TIInputEventArgs
```

## Type Parameters

`TIInputEventArgs`

The input event arguments interface.

`TInputEventArgs`

The input event arguments.

**Inherited Members**

IInterceptableEvent<TIInputEventArgs>.Handler , IInputEvent<TIInputEventArgs, TInputEventArgs>.Raise(TInputEventArgs)

# Enum InputEventResponse

Namespace: DotNetBrowser.Input

Assembly: DotNetBrowser.dll

The response to the input event handlers.

```
public enum InputEventResponse
```

## Fields

`Proceed = 1`

Allow the browser to process this event.

`Suppress = 0`

Suppress this event.

# Namespace DotNetBrowser.Input.DragAndDrop

## Interfaces

IDragAndDrop

The drag and drop functionality of the browser.

# Interface IDragAndDrop

Namespace: DotNetBrowser.Input.DragAndDrop

Assembly: DotNetBrowser.dll

The drag and drop functionality of the browser.

```
public interface IDragAndDrop : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

DropHandler

Gets or sets a handler that is used when the input system reports an underlying drop event with the browser as the drag target.

Enabled

Enables or disables the drag and drop functionality in the browser.

EnterDragHandler

Gets or sets a handler that is used when the input system reports an underlying drag enter event with the browser as the drag target.

# Property DropHandler

Namespace: DotNetBrowser.Input.DragAndDrop

Assembly: DotNetBrowser.dll

## DropHandler

Gets or sets a handler that is used when the input system reports an underlying drop event with the browser as the drag target.

```
IHandler<DropParameters> DropHandler { get; set; }
```

## Property Value

IHandler<DropParameters>

## Remarks

**Important:** the engine will be blocked until the `Handle()` method returns. It is not allowed to invoke any engine methods in the scope of this handler implementation.

The parameters are valid only in the scope of the `Handle()` method and become invalid when this method returns.

## Exceptions

ObjectDisposedException

The IDragAndDrop has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Enabled

Namespace: DotNetBrowser.Input.DragAndDrop

Assembly: DotNetBrowser.dll

## Enabled

Enables or disables the drag and drop functionality in the browser.

```
bool Enabled { get; set; }
```

## Property Value

bool

`true` if DragAndDrop is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IDragAndDrop has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property EnterDragHandler

Namespace: DotNetBrowser.Input.DragAndDrop

Assembly: DotNetBrowser.dll

## EnterDragHandler

Gets or sets a handler that is used when the input system reports an underlying drag enter event with the browser as the drag target.

```
IHandler<EnterDragParameters> EnterDragHandler { get; set; }
```

## Property Value

IHandler<EnterDragParameters>

## Remarks

**Important:** the engine will be blocked until the `Handle()` method returns. It is not allowed to invoke any engine methods in the scope of this handler implementation.

The parameters are valid only in the scope of the `Handle()` method and become invalid when this method returns.

## Exceptions

ObjectDisposedException

The IDragAndDrop has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Namespace DotNetBrowser.Input.DragAndDrop.Handlers

## Classes

DragAndDropParameters

The base class for all IDragAndDrop handlers parameters.

DragEvent

Represents drag and drop event and provides access to the event data.

DropParameters

The parameters of the DropHandler.

EnterDragParameters

The parameters of the EnterDragHandler.

# Class DragAndDropParameters

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

The base class for all IDragAndDrop handlers parameters.

```
public class DragAndDropParameters
```

**Inheritance**

object ← DragAndDropParameters

**Derived**

DropParameters, EnterDragParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

DragAndDrop

Gets the IDragAndDrop instance the handler is associated with.

Event

Gets the drag and drop event.

# Property DragAndDrop

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

## DragAndDrop

Gets the IDragAndDrop instance the handler is associated with.

```
public IDragAndDrop DragAndDrop { get; }
```

## Property Value

IDragAndDrop

The IDragAndDrop instance the handler is associated with.

# Property Event

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

## Event

Gets the drag and drop event.

```
public DragEvent Event { get; }
```

## Property Value

DragEvent

The drag and drop event.

# Class DragEvent

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

Represents drag and drop event and provides access to the event data.

```
public sealed class DragEvent
```

**Inheritance**

object ← DragEvent

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

DataObject

Gets the Windows-specific representation of the drop data.

DropData

Gets the general representation of the drop data.

Location

Gets the location of the mouse cursor in the browser's coordinate system at the time the event occurred.

ScreenLocation

Gets the location of the mouse cursor in the screen's coordinate system at the time the event occurred.

# Property DataObject

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

## DataObject

Gets the Windows-specific representation of the drop data.

```
public IDataObject DataObject { get; }
```

## Property Value

IDataObject

The IDataObject instance or null if the Windows-specific representation is not available.

## Remarks

This functionality is available for .NET Framework on Windows only when the "--enable-com-in-drag-drop" Chromium switch is specified.

# Property DropData

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

## DropData

Gets the general representation of the drop data.

```
public IDropData DropData { get; }
```

## Property Value

IDropData

The drop data.

# Property Location

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

## Location

Gets the location of the mouse cursor in the browser's coordinate system at the time the event occurred.

```
public Point Location { get; }
```

## Property Value

Point

The location of the mouse cursor.

# Property ScreenLocation

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

## ScreenLocation

Gets the location of the mouse cursor in the screen's coordinate system at the time the event occurred.

```
public Point ScreenLocation { get; }
```

## Property Value

Point

The location of the mouse cursor.

# Class DropParameters

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

The parameters of the DropHandler.

```
public class DropParameters : DragAndDropParameters
```

### Inheritance

object ← DragAndDropParameters ← DropParameters

### Inherited Members

DragAndDropParameters.DragAndDrop , DragAndDropParameters.Event , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class EnterDragParameters

Namespace: DotNetBrowser.Input.DragAndDrop.Handlers

Assembly: DotNetBrowser.dll

The parameters of the EnterDragHandler.

```
public sealed class EnterDragParameters : DragAndDropParameters
```

### Inheritance

object ← DragAndDropParameters ← EnterDragParameters

### Inherited Members

DragAndDropParameters.DragAndDrop , DragAndDropParameters.Event , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Namespace DotNetBrowser.Input.Keyboard

## Interfaces

IKeyboard

A service that can be used for keyboard input simulation and handling.

# Interface IKeyboard

Namespace: DotNetBrowser.Input.Keyboard

Assembly: DotNetBrowser.dll

A service that can be used for keyboard input simulation and handling.

```
public interface IKeyboard : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

KeyPressed

Occurs when a key is pressed for the browser. This corresponds to `KeyDown` event of WPF and WinForms controls.

KeyReleased

Occurs when a key is released for the browser. This corresponds to `KeyUp` event of WPF and WinForms controls.

KeyTyped

Occurs when a control code, character, space, enter or backspace key is pressed for the browser. This corresponds to `KeyPress` event of WinForms controls.

# Property KeyPressed

Namespace: DotNetBrowser.Input.Keyboard

Assembly: DotNetBrowser.dll

## KeyPressed

Occurs when a key is pressed for the browser. This corresponds to `KeyDown` event of WPF and WinForms controls.

```
IInterceptableInputEvent<IKeyPressedEventArgs, KeyPressedEventArgs> KeyPressed { get; }
```

## Property Value

IInterceptableInputEvent<IKeyPressedEventArgs, KeyPressedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property KeyReleased

Namespace: DotNetBrowser.Input.Keyboard

Assembly: DotNetBrowser.dll

## KeyReleased

Occurs when a key is released for the browser. This corresponds to `KeyUp` event of WPF and WinForms controls.

```
IInterceptableInputEvent<IKeyReleasedEventArgs, KeyReleasedEventArgs> KeyReleased { get; }
```

## Property Value

IInterceptableInputEvent<IKeyReleasedEventArgs, KeyReleasedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property KeyTyped

Namespace: DotNetBrowser.Input.Keyboard

Assembly: DotNetBrowser.dll

## KeyTyped

Occurs when a control code, character, space, enter or backspace key is pressed for the browser. This corresponds to `KeyPress` event of WinForms controls.

```
IInterceptableInputEvent<IKeyTypedEventArgs, KeyTypedEventArgs> KeyTyped { get; }
```

## Property Value

IInterceptableInputEvent<IKeyTypedEventArgs, KeyTypedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Namespace DotNetBrowser.Input.Keyboard.Events

## Classes

KeyCharEventArgs
The event arguments for key events which provide a character.

KeyEventArgs

KeyModifiers

KeyPressedEventArgs
The event arguments for the KeyPressed event.

KeyReleasedEventArgs
The event arguments for the KeyReleased event.

KeyTypedEventArgs
The event arguments for the KeyTyped event.

## Interfaces

IKeyCharEventArgs
The event arguments for key events which provide a character.

IKeyEventArgs
The base interface of all event arguments for the key events.

IKeyModifiers
The keyboard modifiers applied.

IKeyPressedEventArgs
The event arguments for the KeyPressed event.

IKeyReleasedEventArgs
The event arguments for the KeyReleased event.

IKeyTypedEventArgs
The event arguments for the KeyTyped event.

## Enums

KeyCode

The virtual key codes.

# Interface IKeyCharEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The event arguments for key events which provide a character.

```
public interface IKeyCharEventArgs : IKeyEventArgs
```

**Inherited Members**

IKeyEventArgs.Modifiers , IKeyEventArgs.VirtualKey

## Properties

KeyChar

Gets the character associated with this event.

# Property KeyChar

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## KeyChar

Gets the character associated with this event.

```
string KeyChar { get; }
```

## Property Value

string

The character associated with this event.

# Interface IKeyEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The base interface of all event arguments for the key events.

```
public interface IKeyEventArgs
```

## Properties

Modifiers

Gets the key modifiers.

VirtualKey

Gets the virtual key code.

# Property Modifiers

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## Modifiers

Gets the key modifiers.

```
IKeyModifiers Modifiers { get; }
```

## Property Value

IKeyModifiers

The key modifiers.

# Property VirtualKey

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## VirtualKey

Gets the virtual key code.

```
KeyCode VirtualKey { get; }
```

## Property Value

KeyCode

The virtual key code.

# Interface IKeyModifiers

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The keyboard modifiers applied.

```
public interface IKeyModifiers
```

## Properties

AltDown
: Indicates whether the Alt key is pressed.

AltGraphDown
: Indicates whether the Alt Graph/Right Alt key is pressed.

ControlDown
: Indicates whether the Ctrl key is pressed.

MetaDown
: Indicates whether the Meta key is pressed.

ShiftDown
: Indicates whether the Shift key is pressed.

# Property AltDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## AltDown

Indicates whether the Alt key is pressed.

```
bool AltDown { get; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Property AltGraphDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## AltGraphDown

Indicates whether the Alt Graph/Right Alt key is pressed.

```
bool AltGraphDown { get; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Property ControlDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## ControlDown

Indicates whether the Ctrl key is pressed.

```
bool ControlDown { get; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Property MetaDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## MetaDown

Indicates whether the Meta key is pressed.

```
bool MetaDown { get; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Property ShiftDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## ShiftDown

Indicates whether the Shift key is pressed.

```
bool ShiftDown { get; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Interface IKeyPressedEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The event arguments for the KeyPressed event.

```
public interface IKeyPressedEventArgs : IKeyCharEventArgs, IKeyEventArgs
```

**Inherited Members**

IKeyCharEventArgs.KeyChar , IKeyEventArgs.Modifiers , IKeyEventArgs.VirtualKey

# Interface IKeyReleasedEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The event arguments for the KeyReleased event.

```
public interface IKeyReleasedEventArgs : IKeyEventArgs
```

**Inherited Members**

IKeyEventArgs.Modifiers , IKeyEventArgs.VirtualKey

# Interface IKeyTypedEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The event arguments for the KeyTyped event.

```
public interface IKeyTypedEventArgs : IKeyCharEventArgs, IKeyEventArgs
```

**Inherited Members**

IKeyCharEventArgs.KeyChar , IKeyEventArgs.Modifiers , IKeyEventArgs.VirtualKey

# Class KeyCharEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The event arguments for key events which provide a character.

```
public class KeyCharEventArgs : KeyEventArgs, IKeyCharEventArgs, IKeyEventArgs
```

**Inheritance**

object ← KeyEventArgs ← KeyCharEventArgs

**Implements**

IKeyCharEventArgs, IKeyEventArgs

**Derived**

KeyPressedEventArgs, KeyTypedEventArgs

**Inherited Members**

KeyEventArgs.Modifiers , KeyEventArgs.VirtualKey , KeyEventArgs.Equals(KeyEventArgs) , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

KeyChar

Gets or sets the character associated with this event.

## Methods

Equals(KeyCharEventArgs)

Determines whether the specified object is equal to the current object.

Equals(object)

GetHashCode()

# Property KeyChar

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## KeyChar

Gets or sets the character associated with this event.

```
public string KeyChar { get; set; }
```

## Property Value

string

The character associated with this event.

# Method Equals

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

## Equals(KeyCharEventArgs)

Determines whether the specified object is equal to the current object.

```
protected bool Equals(KeyCharEventArgs other)
```

### Parameters

`other` KeyCharEventArgs

The object to compare with the current object.

### Returns

bool

true if the specified object is equal to the current object; otherwise, false.

# Method GetHashCode

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Enum KeyCode

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The virtual key codes.

```
public enum KeyCode
```

## Fields

`Accept = 30`

IME accept.

`Add = 107`

The add key.

`Apps = 93`

Applications key (Natural keyboard).

`Attn = 246`

The Attn key.

`Back = 8`

The BACKSPACE key (VK_BACK).

`BrowserBack = 166`

The Browser Back key.

`BrowserFavorites = 171`

The Browser Favorites key.

`BrowserForward = 167`

The Browser Forward key.

`BrowserHome = 172`

The Browser Start and Home key.

`BrowserRefresh = 168`

The Browser Refresh key.

`BrowserSearch = 170`

The Browser Search key.

`BrowserStop = 169`

The Browser Stop key.

`Cancel = 3`

Control-break processing (VK_CANCEL).

`Capital = 20`

The CAPS LOCK key (VK_CAPITAL).

`Clear = 12`

The CLEAR key (VK_CLEAR).

`Command = 157`

The Command key.

`Control = 17`

The CTRL key (VK_CONTROL).

`Convert = 28`

IME convert.

`Crsel = 247`

The CrSel key.

`DbeAlphanumeric = 240`

The DBE_ALPHANUMERIC key.

`Decimal = 110`

The decimal key.

`Delete = 46`

The DEL key (VK_DELETE).

`Divide = 111`

The divide key.

`Down = 40`

The DOWN ARROW key.

`End = 35`

The END key.

`Ereof = 249`

The Erase EOF key.

`Escape = 27`

ESC key.

`Execute = 43`

The EXECUTE key

`Exsel = 248`

The ExSel key.

`F1 = 112`

The 'F1' key.

`F10 = 121`

The 'F10' key.

`F11 = 122`

The 'F11' key.

`F12 = 123`

The 'F1' key.

`F13 = 124`

The 'F1' key.

`F14 = 125`

The 'F14' key.

`F15 = 126`

The 'F15' key.

`F16 = 127`

The 'F16' key.

`F17 = 128`

The 'F17' key.

`F18 = 129`

The 'F18' key.

`F19 = 130`

The 'F19' key.

`F2 = 113`

The 'F2' key.

`F20 = 131`

The 'F20' key.

`F21 = 132`

The 'F21' key.

`F22 = 133`

The 'F22' key.

`F23 = 134`

The 'F23' key.

`F24 = 135`

The 'F24' key.

`F3 = 114`

The 'F3' key.

`F4 = 115`

The 'F4' key.

`F5 = 116`

The 'F5' key.

`F6 = 117`

The 'F6' key.

`F7 = 118`

The 'F7' key.

`F8 = 119`

The 'F8' key.

`F9 = 120`

The 'F9' key.

`Final = 24`

IME final mode.

`Hanguel = 21`

IME Hanguel mode (maintained for compatibility; use Hangul).

`Hangul = 21`

IME Hangul mode.

`Hanja = 25`

IME Hanja mode.

`Help = 47`

The HELP key (VK_HELP).

`Home = 36`

The HOME key.

`Insert = 45`

The INS key (VK_INSERT).

`Junja = 23`

IME Junja mode.

`Kana = 21`

IME Kana mode (VK_KANA).

`Kanji = 25`

IME Kanji mode.

`LButton = 1`

The left mouse button (VK_LBUTTON).

`LCmd = 91`

The Command key.

`LControl = 162`

The Left CONTROL key.

`LMenu = 164`

The Left MENU key.

`LShift = 160`

The Left SHIFT key.

`LWin = 91`

Left Windows key (Natural keyboard).

`LaunchApp1 = 182`

The Start Application 1 key.

`LaunchApp2 = 183`

The Start Application 2 key.

`LaunchMail = 180`

The Start Mail key.

`LaunchMediaSelect = 181`

The Select Media key.

`Left = 37`

The LEFT ARROW key.

`MButton = 4`

The middle mouse button (three-button mouse)

`MediaNextTrack = 176`

The Next Track key.

`MediaPlayPause = 179`

The Play/Pause Media key.

`MediaPrevTrack = 177`

The Previous Track key.

`MediaStop = 178`

The Stop Media key.

`Menu = 18`

The ALT key (VK_MENU).

`ModeChange = 31`

IME mode change request.

`Multiply = 106`

The multiply key.

`Next = 34`

The PAGE DOWN key.

`NoName = 252`

Reserved.

`NonConvert = 29`

IME nonconvert.

`Numlock = 144`

The NUM LOCK key.

`Numpad0 = 96`

The numeric keypad '0' key.

`Numpad1 = 97`

The numeric keypad '1' key.

`Numpad2 = 98`

The numeric keypad '2' key.

`Numpad3 = 99`

The numeric keypad '3' key.

`Numpad4 = 100`

The numeric keypad '4' key.

`Numpad5 = 101`

The numeric keypad '5' key.

`Numpad6 = 102`

The numeric keypad '6' key.

`Numpad7 = 103`

The numeric keypad '7' key.

`Numpad8 = 104`

The numeric keypad '8' key.

`Numpad9 = 105`

The numeric keypad '9' key.

`Oem1 = 186`

Used for miscellaneous characters; it can vary by keyboard. For the US standard keyboard, the ';:' key.

`Oem102 = 226`

Either the angle bracket key or the backslash key on the RT 102-key keyboard.

`Oem2 = 191`

Used for miscellaneous characters; it can vary by keyboard. For the US standard keyboard, the '[{' key.

`Oem3 = 192`

Used for miscellaneous characters; it can vary by keyboard. For the US standard keyboard, the '[{' key.

`Oem4 = 219`

Used for miscellaneous characters; it can vary by keyboard. For the US standard keyboard, the '[{' key.

`Oem5 = 220`

Used for miscellaneous characters; it can vary by keyboard. For the US standard keyboard, the '|' key.

`Oem6 = 221`

Used for miscellaneous characters; it can vary by keyboard. For the US standard keyboard, the ']}' key.

`Oem7 = 222`

Used for miscellaneous characters; it can vary by keyboard. For the US standard keyboard, the 'single-quote/double-quote' key.

`Oem8 = 223`

Used for miscellaneous characters; it can vary by keyboard.

`OemClear = 254`

The Clear key.

`OemComma = 188`

For any country/region, the ',' key.

`OemMinus = 189`

For any country/region, the '-' key.

`OemPeriod = 190`

For any country/region, the '.' key.

`OemPlus = 187`

For any country/region, the '+' key.

`Pa1 = 253`

The PA1 key.

`Packet = 231`

Used to pass Unicode characters as if they were keystrokes. The VK_PACKET key is the low word of a 32-bit Virtual Key value used for non-keyboard input methods.

`Pause = 19`

The PAUSE key (VK_PAUSE).

`Play = 250`

The Play key.

`Print = 42`

The PRINT key.

`Prior = 33`

The PAGE UP key.

`Processkey = 229`

The IME PROCESS key.

`RButton = 2`

The right mouse button (VK_RBUTTON).

`RCmd = 93`

The Right Command key.

`RControl = 163`

The Right CONTROL key.

`RMenu = 165`

The Right MENU key.

`RShift = 161`

The Right SHIFT key.

`RWin = 92`

Right Windows key (Natural keyboard).

`Return = 13`

The ENTER key (VK_RETURN).

`Right = 39`

The RIGHT ARROW key.

`Scroll = 145`

The SCROLL LOCK key.

`Select = 41`

The SELECT key.

`Separator = 108`

The separator key.

`Shift = 16`

The SHIFT key (VK_SHIFT).

`Sleep = 95`

Computer Sleep key.

`Snapshot = 44`

The PRINT SCREEN / SysRq key (VK_SNAPSHOT).

`Space = 32`

The spacebar (VK_SPACE).

`Subtract = 109`

The subtract key.

`Tab = 9`

The TAB key (VK_TAB).

`Unknown = 0`

Reserved value.

`Up = 38`

The UP ARROW key.

`Vk0 = 48`

The '0' key.

`Vk1 = 49`

The '1' key.

`Vk2 = 50`

The '2' key.

`Vk3 = 51`

The '3' key.

`Vk4 = 52`

The '4' key.

`Vk5 = 53`

The '5' key.

`Vk6 = 54`

The '6' key.

`Vk7 = 55`

The '7' key.

`Vk8 = 56`

The '8' key.

`Vk9 = 57`

The '9' key.

`VkA = 65`

The 'A' key.

`VkB = 66`

The 'B' key.

`VkC = 67`

The 'C' key.

`VkD = 68`

The 'D' key.

`VkE = 69`

The 'E' key.

`VkF = 70`

The 'F' key.

`VkG = 71`

The 'G' key.

`VkH = 72`

The 'H' key.

`VkI = 73`

The 'I' key.

`VkJ = 74`

The 'J' key.

`VkK = 75`

The 'K' key.

`VkL = 76`

The 'L' key.

`VkM = 77`

The 'M' key.

`VkN = 78`

The 'N' key.

`VkO = 79`

The 'O' key.

`VkP = 80`

The 'P' key.

`VkQ = 81`

The 'Q' key.

`VkR = 82`

The 'R' key.

`VkS = 83`

The 'S' key.

`VkT = 84`

The 'T' key.

`VkU = 85`

The 'U' key.

`VkV = 86`

The 'V' key.

`VkW = 87`

The 'W' key.

`VkX = 88`

The 'X' key.

`VkY = 89`

The 'Y' key.

`VkZ = 90`

The 'Z' key.

`VolumeDown = 174`

The Volume Down key.

`VolumeMute = 173`

The Volume Mute key.

`VolumeUp = 175`

The Volume Up key.

`XButton1 = 5`

The X1 mouse button (VK_XBUTTON1).

`XButton2 = 6`

The X2 mouse button(VK_XBUTTON2).

`Zoom = 251`

The Zoom key.

## See Also

https://docs.microsoft.com/en-us/windows/win32/inputdev/virtual-key-codes

# Class KeyEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

```
public class KeyEventArgs : IKeyEventArgs
```

**Inheritance**

object ← KeyEventArgs

**Implements**

IKeyEventArgs

**Derived**

KeyCharEventArgs, KeyReleasedEventArgs

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Modifiers

Gets or sets the key modifiers.

VirtualKey

Gets or sets the virtual key code.

## Methods

Equals(KeyEventArgs)

Determines whether the specified object is equal to the current object.

Equals(object)

GetHashCode()

# Property Modifiers

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## Modifiers

Gets or sets the key modifiers.

```
public IKeyModifiers Modifiers { get; set; }
```

## Property Value

IKeyModifiers

The key modifiers.

# Property VirtualKey

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## VirtualKey

Gets or sets the virtual key code.

```
public KeyCode VirtualKey { get; set; }
```

## Property Value

KeyCode

The virtual key code.

# Method Equals

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

## Equals(KeyEventArgs)

Determines whether the specified object is equal to the current object.

```
protected bool Equals(KeyEventArgs other)
```

### Parameters

`other` KeyEventArgs

The object to compare with the current object.

### Returns

bool

true if the specified object is equal to the current object; otherwise, false.

# Method GetHashCode

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class KeyModifiers

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

```
public sealed class KeyModifiers : IKeyModifiers
```

**Inheritance**

object ← KeyModifiers

**Implements**

IKeyModifiers

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AltDown

Indicates whether the Alt key is pressed.

AltGraphDown

Indicates whether the Alt Graph/Right Alt key is pressed.

ControlDown

Indicates whether the Ctrl key is pressed.

MetaDown

Indicates whether the Meta key is pressed.

ShiftDown

Indicates whether the Shift key is pressed.

## Methods

Equals(object)

GetHashCode()

# Property AltDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## AltDown

Indicates whether the Alt key is pressed.

```
public bool AltDown { get; set; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Property AltGraphDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## AltGraphDown

Indicates whether the Alt Graph/Right Alt key is pressed.

```
public bool AltGraphDown { get; set; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Property ControlDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## ControlDown

Indicates whether the Ctrl key is pressed.

```
public bool ControlDown { get; set; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Property MetaDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## MetaDown

Indicates whether the Meta key is pressed.

```
public bool MetaDown { get; set; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Property ShiftDown

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## ShiftDown

Indicates whether the Shift key is pressed.

```
public bool ShiftDown { get; set; }
```

## Property Value

bool

`true` if the modifier key is pressed, `false` otherwise.

# Method Equals

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class KeyPressedEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The event arguments for the KeyPressed event.

```
public sealed class KeyPressedEventArgs : KeyCharEventArgs, IKeyPressedEventArgs,
IKeyCharEventArgs, IKeyEventArgs
```

### Inheritance

object ← KeyEventArgs ← KeyCharEventArgs ← KeyPressedEventArgs

### Implements

IKeyPressedEventArgs, IKeyCharEventArgs, IKeyEventArgs

### Inherited Members

KeyCharEventArgs.KeyChar , KeyCharEventArgs.Equals(object) , KeyCharEventArgs.GetHashCode() , KeyEventArgs.Modifiers , KeyEventArgs.VirtualKey , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class KeyReleasedEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The event arguments for the KeyReleased event.

```
public sealed class KeyReleasedEventArgs : KeyEventArgs, IKeyReleasedEventArgs,
IKeyEventArgs
```

### Inheritance

object ← KeyEventArgs ← KeyReleasedEventArgs

### Implements

IKeyReleasedEventArgs, IKeyEventArgs

### Inherited Members

KeyEventArgs.Modifiers , KeyEventArgs.VirtualKey , KeyEventArgs.Equals(object) , KeyEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class KeyTypedEventArgs

Namespace: DotNetBrowser.Input.Keyboard.Events

Assembly: DotNetBrowser.dll

The event arguments for the KeyTyped event.

```
public sealed class KeyTypedEventArgs : KeyCharEventArgs, IKeyTypedEventArgs,
IKeyCharEventArgs, IKeyEventArgs
```

### Inheritance

object ← KeyEventArgs ← KeyCharEventArgs ← KeyTypedEventArgs

### Implements

IKeyTypedEventArgs, IKeyCharEventArgs, IKeyEventArgs

### Inherited Members

KeyCharEventArgs.KeyChar , KeyCharEventArgs.Equals(object) , KeyCharEventArgs.GetHashCode() , KeyEventArgs.Modifiers , KeyEventArgs.VirtualKey , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Namespace DotNetBrowser.Input.Mouse

## Classes

MouseExtensions

The mouse input simulation service extension methods.

## Interfaces

IMouse

A service that can be used for mouse input simulation and handling.

# Interface IMouse

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.dll

A service that can be used for mouse input simulation and handling.

```
public interface IMouse : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

**Extension Methods**
MouseExtensions.SimulateClick(IMouse, MouseButton, IElement) ,
MouseExtensions.SimulateClick(IMouse, MouseButton, Point)

## Properties

Dragged
Gets the mouse dragged event for the browser.

Entered
Occurs when the mouse has entered the web page.

Exited
Occurs when the mouse has exited the web page.

Moved
Occurs when the mouse has moved over the web page.

Pressed
Occurs when the mouse button has been pressed on the web page.

Released
Occurs when the mouse button has been released on the web page.

WheelMoved
Occurs when the mouse wheel has scrolled on the web page.

# Property Dragged

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.dll

## Dragged

Gets the mouse dragged event for the browser.

```
IInputEvent<IMouseDraggedEventArgs, MouseDraggedEventArgs> Dragged { get; }
```

## Property Value

IInputEvent<IMouseDraggedEventArgs, MouseDraggedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property Entered

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.dll

## Entered

Occurs when the mouse has entered the web page.

```
IInterceptableInputEvent<IMouseEnteredEventArgs, MouseEnteredEventArgs> Entered { get; }
```

## Property Value

IInterceptableInputEvent<IMouseEnteredEventArgs, MouseEnteredEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property Exited

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.dll

## Exited

Occurs when the mouse has exited the web page.

```
IInterceptableInputEvent<IMouseExitedEventArgs, MouseExitedEventArgs> Exited { get; }
```

## Property Value

IInterceptableInputEvent<IMouseExitedEventArgs, MouseExitedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property Moved

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.dll

## Moved

Occurs when the mouse has moved over the web page.

```
IInterceptableInputEvent<IMouseMovedEventArgs, MouseMovedEventArgs> Moved { get; }
```

## Property Value

IInterceptableInputEvent<IMouseMovedEventArgs, MouseMovedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property Pressed

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.dll

## Pressed

Occurs when the mouse button has been pressed on the web page.

```
IInterceptableInputEvent<IMousePressedEventArgs, MousePressedEventArgs> Pressed { get; }
```

## Property Value

IInterceptableInputEvent<IMousePressedEventArgs, MousePressedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property Released

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.dll

## Released

Occurs when the mouse button has been released on the web page.

```
IInterceptableInputEvent<IMouseReleasedEventArgs, MouseReleasedEventArgs> Released { get; }
```

## Property Value

IInterceptableInputEvent<IMouseReleasedEventArgs, MouseReleasedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property WheelMoved

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.dll

## WheelMoved

Occurs when the mouse wheel has scrolled on the web page.

```
IInterceptableInputEvent<IMouseWheelMovedEventArgs, MouseWheelMovedEventArgs> WheelMoved {
get; }
```

## Property Value

IInterceptableInputEvent<IMouseWheelMovedEventArgs, MouseWheelMovedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Class MouseExtensions

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.Core.dll

The mouse input simulation service extension methods.

```
public static class MouseExtensions
```

**Inheritance**

object ← MouseExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SimulateClick(IMouse, MouseButton, IElement)

Simulate mouse button click at the specified element.

SimulateClick(IMouse, MouseButton, Point)

Simulate mouse button click at the specified location.

# Method SimulateClick

Namespace: DotNetBrowser.Input.Mouse

Assembly: DotNetBrowser.Core.dll

## SimulateClick(IMouse, MouseButton, Point)

Simulate mouse button click at the specified location.

```
public static void SimulateClick(this IMouse mouse, MouseButton mouseButton,
Point location)
```

### Parameters

`mouse` IMouse

The mouse input simulation service.

`mouseButton` MouseButton

The mouse button that should be used for clicking.

`location` Point

The mouse pointer location relative to the top left corner of the browser.

### Remarks

The mouse click is simulated by raising Moved, Pressed, and Released events in a sequence.

## SimulateClick(IMouse, MouseButton, IElement)

Simulate mouse button click at the specified element.

```
public static void SimulateClick(this IMouse mouse, MouseButton mouseButton,
IElement element)
```

## Parameters

`mouse` IMouse

The mouse input simulation service.

`mouseButton` MouseButton

The mouse button that should be used for clicking.

`element` IElement

The element to click at.

## Remarks

The mouse click is simulated by raising Moved, Pressed, and Released events in a sequence.

The point to click at is determined as the center of the bounding client rectangle of the specified element.

# Namespace DotNetBrowser.Input.Mouse.Events

## Classes

MouseButtonEventArgs
: Event arguments for the IMouse events associated with a mouse button.

MouseDraggedEventArgs
: The event arguments for the Dragged event.

MouseEnteredEventArgs
: The event arguments for the Entered event.

MouseEventArgs

MouseExitedEventArgs
: The event arguments for the Exited event.

MouseModifiers

MouseMovedEventArgs
: The event arguments for the Moved event.

MousePressedEventArgs
: The event arguments for the Pressed event.

MouseReleasedEventArgs
: The event arguments for the Released event.

MouseWheelMovedEventArgs
: The event arguments for the WheelMoved event.

## Interfaces

IMouseButtonEventArgs
: Event arguments for the IMouse events associated with a mouse button.

IMouseDraggedEventArgs
: The event arguments for the Dragged event.

IMouseEnteredEventArgs
: The event arguments for the Entered event.

IMouseEventArgs

The base interface for all IMouse event arguments.

IMouseExitedEventArgs
The event arguments for the Exited event.

IMouseModifiers
The mouse modifiers indicating which mouse buttons are currently pressed.

IMouseMovedEventArgs
The event arguments for the Moved event.

IMousePressedEventArgs
The event arguments for the Pressed event.

IMouseReleasedEventArgs
The event arguments for the Released event.

IMouseWheelMovedEventArgs
The event arguments for the WheelMoved event.

## Enums

MouseButton
The mouse buttons.

MouseScrollType
Scroll type.

# Interface IMouseButtonEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

Event arguments for the IMouse events associated with a mouse button.

```
public interface IMouseButtonEventArgs : IMouseEventArgs
```

**Inherited Members**

IMouseEventArgs.Location , IMouseEventArgs.LocationOnScreen

## Properties

Button

Gets the mouse button this event is related to.

ClickCount

Gets the count of consecutive clicks that happened in a short amount of time.

KeyModifiers

Gets the keyboard modifiers.

MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property Button

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Button

Gets the mouse button this event is related to.

```
MouseButton Button { get; }
```

## Property Value

MouseButton

The mouse button this event is related to.

# Property ClickCount

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## ClickCount

Gets the count of consecutive clicks that happened in a short amount of time.

```
uint ClickCount { get; }
```

## Property Value

uint

The count of consecutive clicks that happened in a short amount of time.

# Property KeyModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## KeyModifiers

Gets the keyboard modifiers.

```
IKeyModifiers KeyModifiers { get; }
```

## Property Value

IKeyModifiers

The keyboard modifiers applied.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

```
IMouseModifiers MouseModifiers { get; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Interface IMouseDraggedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Dragged event.

```
public interface IMouseDraggedEventArgs : IMouseEventArgs
```

**Inherited Members**

IMouseEventArgs.Location , IMouseEventArgs.LocationOnScreen

## Properties

Button

Gets the button that is pressed during the move.

Modifiers

Gets the keyboard modifiers.

MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property Button

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Button

Gets the button that is pressed during the move.

```
MouseButton Button { get; }
```

## Property Value

MouseButton

The button that is pressed during the move.

# Property Modifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Modifiers

Gets the keyboard modifiers.

```
IKeyModifiers Modifiers { get; }
```

## Property Value

IKeyModifiers

The keyboard modifiers applied.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

```
IMouseModifiers MouseModifiers { get; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Interface IMouseEnteredEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Entered event.

```
public interface IMouseEnteredEventArgs : IMouseEventArgs
```

**Inherited Members**

IMouseEventArgs.Location , IMouseEventArgs.LocationOnScreen

## Properties

MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

```
IMouseModifiers MouseModifiers { get; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Interface IMouseEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The base interface for all IMouse event arguments.

```
public interface IMouseEventArgs
```

## Properties

Location

Gets the mouse position relative to the bounds of the browser instance.

LocationOnScreen

Gets the mouse position relative to the bounds of the screen.

# Property Location

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Location

Gets the mouse position relative to the bounds of the browser instance.

```
Point Location { get; }
```

## Property Value

Point

The mouse position relative to the bounds of the browser instance.

# Property LocationOnScreen

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## LocationOnScreen

Gets the mouse position relative to the bounds of the screen.

```
Point LocationOnScreen { get; }
```

## Property Value

Point

The mouse position relative to the bounds of the screen.

# Interface IMouseExitedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Exited event.

```
public interface IMouseExitedEventArgs : IMouseEventArgs
```

**Inherited Members**

IMouseEventArgs.Location , IMouseEventArgs.LocationOnScreen

## Properties

MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

```
IMouseModifiers MouseModifiers { get; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Interface IMouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The mouse modifiers indicating which mouse buttons are currently pressed.

```
public interface IMouseModifiers
```

## Properties

BackButtonDown
: Indicates whether the first extended ("Back") mouse button is pressed.

ForwardButtonDown
: Indicates whether the second extended ("Forward") mouse button is pressed.

MiddleButtonDown
: Indicates whether the middle mouse button is pressed.

PrimaryButtonDown
: Indicates whether the primary mouse button is pressed.

SecondaryButtonDown
: Indicates whether the secondary mouse button is pressed.

# Property BackButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## BackButtonDown

Indicates whether the first extended ("Back") mouse button is pressed.

```
bool BackButtonDown { get; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Property ForwardButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## ForwardButtonDown

Indicates whether the second extended ("Forward") mouse button is pressed.

```
bool ForwardButtonDown { get; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Property MiddleButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MiddleButtonDown

Indicates whether the middle mouse button is pressed.

```
bool MiddleButtonDown { get; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Property PrimaryButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## PrimaryButtonDown

Indicates whether the primary mouse button is pressed.

```
bool PrimaryButtonDown { get; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Property SecondaryButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## SecondaryButtonDown

Indicates whether the secondary mouse button is pressed.

```
bool SecondaryButtonDown { get; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Interface IMouseMovedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Moved event.

```
public interface IMouseMovedEventArgs : IMouseEventArgs
```

**Inherited Members**

IMouseEventArgs.Location , IMouseEventArgs.LocationOnScreen

## Properties

MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets the mouse modifiers indicating which mouse buttons are currently pressed.

```
IMouseModifiers MouseModifiers { get; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Interface IMousePressedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Pressed event.

```
public interface IMousePressedEventArgs : IMouseButtonEventArgs, IMouseEventArgs
```

**Inherited Members**

IMouseButtonEventArgs.Button , IMouseButtonEventArgs.ClickCount , IMouseButtonEventArgs.KeyModifiers , IMouseButtonEventArgs.MouseModifiers , IMouseEventArgs.Location , IMouseEventArgs.LocationOnScreen

# Interface IMouseReleasedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Released event.

```
public interface IMouseReleasedEventArgs : IMouseButtonEventArgs, IMouseEventArgs
```

**Inherited Members**

IMouseButtonEventArgs.Button , IMouseButtonEventArgs.ClickCount , IMouseButtonEventArgs.KeyModifiers , IMouseButtonEventArgs.MouseModifiers , IMouseEventArgs.Location , IMouseEventArgs.LocationOnScreen

# Interface IMouseWheelMovedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the WheelMoved event.

```
public interface IMouseWheelMovedEventArgs : IMouseEventArgs
```

**Inherited Members**

IMouseEventArgs.Location , IMouseEventArgs.LocationOnScreen

## Properties

DeltaX

The amount of units to scroll horizontally.

DeltaY

The amount of units to scroll vertically.

Modifiers

The keyboard modifiers applied.

ScrollType

The scroll type of the event.

# Property DeltaX

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## DeltaX

The amount of units to scroll horizontally.

```
float DeltaX { get; set; }
```

## Property Value

float

# Property DeltaY

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## DeltaY

The amount of units to scroll vertically.

```
float DeltaY { get; set; }
```

## Property Value

float

# Property Modifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Modifiers

The keyboard modifiers applied.

```
IKeyModifiers Modifiers { get; }
```

## Property Value

IKeyModifiers

# Property ScrollType

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## ScrollType

The scroll type of the event.

```
MouseScrollType ScrollType { get; }
```

## Property Value

MouseScrollType

# Enum MouseButton

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The mouse buttons.

```
public enum MouseButton
```

## Fields

`Back = 5`

Back (XButton1) button.

`Forward = 6`

Forward (XButton2) button.

`Left = 2`

Left (primary) button.

`Middle = 3`

Middle button.

`None = 1`

No button.

`Right = 4`

Right (secondary) button.

# Class MouseButtonEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

Event arguments for the IMouse events associated with a mouse button.

```
public class MouseButtonEventArgs : MouseEventArgs, IMouseButtonEventArgs, IMouseEventArgs
```

**Inheritance**

object ← MouseEventArgs ← MouseButtonEventArgs

**Implements**

IMouseButtonEventArgs, IMouseEventArgs

**Derived**

MousePressedEventArgs, MouseReleasedEventArgs

**Inherited Members**

MouseEventArgs.Location , MouseEventArgs.LocationOnScreen , MouseEventArgs.Equals(object) , MouseEventArgs.GetHashCode() , MouseEventArgs.Equals(MouseEventArgs) , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Button

Gets or sets the mouse button this event is related to.

ClickCount

Gets or sets the count of consecutive clicks that happened in a short amount of time.

KeyModifiers

Gets or sets the keyboard modifiers.

MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property Button

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Button

Gets or sets the mouse button this event is related to.

```
public MouseButton Button { get; set; }
```

## Property Value

MouseButton

The mouse button this event is related to.

# Property ClickCount

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## ClickCount

Gets or sets the count of consecutive clicks that happened in a short amount of time.

```
public uint ClickCount { get; set; }
```

## Property Value

uint

The count of consecutive clicks that happened in a short amount of time.

# Property KeyModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## KeyModifiers

Gets or sets the keyboard modifiers.

```
public IKeyModifiers KeyModifiers { get; set; }
```

## Property Value

IKeyModifiers

The keyboard modifiers applied.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

```
public IMouseModifiers MouseModifiers { get; set; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Class MouseDraggedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Dragged event.

```
public sealed class MouseDraggedEventArgs : MouseEventArgs, IMouseDraggedEventArgs,
IMouseEventArgs
```

**Inheritance**

object ← MouseEventArgs ← MouseDraggedEventArgs

**Implements**

IMouseDraggedEventArgs, IMouseEventArgs

**Inherited Members**

MouseEventArgs.Location , MouseEventArgs.LocationOnScreen , MouseEventArgs.Equals(object) , MouseEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Button

Gets or sets the button that is pressed during the move.

Modifiers

Gets or sets the keyboard modifiers.

MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property Button

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Button

Gets or sets the button that is pressed during the move.

```
public MouseButton Button { get; set; }
```

## Property Value

MouseButton

The button that is pressed during the move.

# Property Modifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Modifiers

Gets or sets the keyboard modifiers.

```
public IKeyModifiers Modifiers { get; set; }
```

## Property Value

IKeyModifiers

The keyboard modifiers applied.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

```
public IMouseModifiers MouseModifiers { get; set; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Class MouseEnteredEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Entered event.

```
public sealed class MouseEnteredEventArgs : MouseEventArgs, IMouseEnteredEventArgs,
IMouseEventArgs
```

**Inheritance**

object ← MouseEventArgs ← MouseEnteredEventArgs

**Implements**

IMouseEnteredEventArgs, IMouseEventArgs

**Inherited Members**

MouseEventArgs.Location , MouseEventArgs.LocationOnScreen , MouseEventArgs.Equals(object) , MouseEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

```
public IMouseModifiers MouseModifiers { get; set; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Class MouseEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

```
public abstract class MouseEventArgs : IMouseEventArgs
```

**Inheritance**

object ← MouseEventArgs

**Implements**

IMouseEventArgs

**Derived**

MouseButtonEventArgs, MouseDraggedEventArgs, MouseEnteredEventArgs, MouseExitedEventArgs, MouseMovedEventArgs, MouseWheelMovedEventArgs

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Location

Gets or sets the mouse position relative to the bounds of the browser instance.

LocationOnScreen

Gets or sets the mouse position relative to the bounds of the screen.

## Methods

Equals(MouseEventArgs)

Determines whether the specified object is equal to the current object.

Equals(object)

GetHashCode()

# Property Location

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Location

Gets or sets the mouse position relative to the bounds of the browser instance.

```
public Point Location { get; set; }
```

## Property Value

Point

The mouse position relative to the bounds of the browser instance.

# Property LocationOnScreen

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## LocationOnScreen

Gets or sets the mouse position relative to the bounds of the screen.

```
public Point LocationOnScreen { get; set; }
```

## Property Value

Point

The mouse position relative to the bounds of the screen.

# Method Equals

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

## Equals(MouseEventArgs)

Determines whether the specified object is equal to the current object.

```
protected bool Equals(MouseEventArgs other)
```

### Parameters

`other` MouseEventArgs

The object to compare with the current object.

### Returns

bool

true if the specified object is equal to the current object; otherwise, false.

# Method GetHashCode

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class MouseExitedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Exited event.

```
public sealed class MouseExitedEventArgs : MouseEventArgs, IMouseExitedEventArgs,
IMouseEventArgs
```

**Inheritance**

object ← MouseEventArgs ← MouseExitedEventArgs

**Implements**

IMouseExitedEventArgs, IMouseEventArgs

**Inherited Members**

MouseEventArgs.Location , MouseEventArgs.LocationOnScreen , MouseEventArgs.Equals(object) , MouseEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

```
public IMouseModifiers MouseModifiers { get; set; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Class MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

```
public sealed class MouseModifiers : IMouseModifiers
```

**Inheritance**

object ← MouseModifiers

**Implements**

IMouseModifiers

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

BackButtonDown

Indicates whether the first extended ("Back") mouse button is pressed.

ForwardButtonDown

Indicates whether the second extended ("Forward") mouse button is pressed.

MiddleButtonDown

Indicates whether the middle mouse button is pressed.

PrimaryButtonDown

Indicates whether the primary mouse button is pressed.

SecondaryButtonDown

Indicates whether the secondary mouse button is pressed.

## Methods

Equals(object)

GetHashCode()

# Property BackButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## BackButtonDown

Indicates whether the first extended ("Back") mouse button is pressed.

```
public bool BackButtonDown { get; set; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Property ForwardButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## ForwardButtonDown

Indicates whether the second extended ("Forward") mouse button is pressed.

```
public bool ForwardButtonDown { get; set; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Property MiddleButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MiddleButtonDown

Indicates whether the middle mouse button is pressed.

```
public bool MiddleButtonDown { get; set; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Property PrimaryButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## PrimaryButtonDown

Indicates whether the primary mouse button is pressed.

```
public bool PrimaryButtonDown { get; set; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Property SecondaryButtonDown

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## SecondaryButtonDown

Indicates whether the secondary mouse button is pressed.

```
public bool SecondaryButtonDown { get; set; }
```

## Property Value

bool

`true` if the mouse button is pressed, `false` otherwise.

# Method Equals

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class MouseMovedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Moved event.

```
public sealed class MouseMovedEventArgs : MouseEventArgs, IMouseMovedEventArgs,
IMouseEventArgs
```

**Inheritance**

object ← MouseEventArgs ← MouseMovedEventArgs

**Implements**

IMouseMovedEventArgs, IMouseEventArgs

**Inherited Members**

MouseEventArgs.Location , MouseEventArgs.LocationOnScreen , MouseEventArgs.Equals(object) , MouseEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

# Property MouseModifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## MouseModifiers

Gets or sets the mouse modifiers indicating which mouse buttons are currently pressed.

```
public IMouseModifiers MouseModifiers { get; set; }
```

## Property Value

IMouseModifiers

The mouse modifiers applied.

# Class MousePressedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Pressed event.

```
public sealed class MousePressedEventArgs : MouseButtonEventArgs, IMousePressedEventArgs,
IMouseButtonEventArgs, IMouseEventArgs
```

### Inheritance

object ← MouseEventArgs ← MouseButtonEventArgs ← MousePressedEventArgs

### Implements

IMousePressedEventArgs, IMouseButtonEventArgs, IMouseEventArgs

### Inherited Members

MouseButtonEventArgs.Button , MouseButtonEventArgs.ClickCount , MouseButtonEventArgs.KeyModifiers , MouseButtonEventArgs.MouseModifiers , MouseEventArgs.Location , MouseEventArgs.LocationOnScreen , MouseEventArgs.Equals(object) , MouseEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class MouseReleasedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the Released event.

```
public sealed class MouseReleasedEventArgs : MouseButtonEventArgs, IMouseReleasedEventArgs,
IMouseButtonEventArgs, IMouseEventArgs
```

### Inheritance

object ← MouseEventArgs ← MouseButtonEventArgs ← MouseReleasedEventArgs

### Implements

IMouseReleasedEventArgs, IMouseButtonEventArgs, IMouseEventArgs

### Inherited Members

MouseButtonEventArgs.Button , MouseButtonEventArgs.ClickCount , MouseButtonEventArgs.KeyModifiers , MouseButtonEventArgs.MouseModifiers , MouseEventArgs.Location , MouseEventArgs.LocationOnScreen , MouseEventArgs.Equals(object) , MouseEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Enum MouseScrollType

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

Scroll type.

```
public enum MouseScrollType
```

## Fields

`BlockScroll = 2`

Defines the scroll by page (block).

`UnitScroll = 1`

Defines the scroll by units.

# Class MouseWheelMovedEventArgs

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

The event arguments for the WheelMoved event.

```
public sealed class MouseWheelMovedEventArgs : MouseEventArgs, IMouseWheelMovedEventArgs,
IMouseEventArgs
```

**Inheritance**

object ← MouseEventArgs ← MouseWheelMovedEventArgs

**Implements**

IMouseWheelMovedEventArgs, IMouseEventArgs

**Inherited Members**

MouseEventArgs.Location , MouseEventArgs.LocationOnScreen , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

DeltaX

The amount of units to scroll horizontally.

DeltaY

The amount of units to scroll vertically.

Modifiers

The keyboard modifiers applied.

ScrollType

The scroll type of the event.

## Methods

Equals(object)

GetHashCode()

# Property DeltaX

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## DeltaX

The amount of units to scroll horizontally.

```
public float DeltaX { get; set; }
```

## Property Value

float

# Property DeltaY

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## DeltaY

The amount of units to scroll vertically.

```
public float DeltaY { get; set; }
```

## Property Value

float

# Property Modifiers

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Modifiers

The keyboard modifiers applied.

```
public IKeyModifiers Modifiers { get; set; }
```

## Property Value

IKeyModifiers

# Property ScrollType

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## ScrollType

The scroll type of the event.

```
public MouseScrollType ScrollType { get; set; }
```

## Property Value

MouseScrollType

# Method Equals

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Input.Mouse.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Namespace DotNetBrowser.Input.Touch

## Interfaces

ITouch

A service that can be used for touch input handling.

# Interface ITouch

Namespace: DotNetBrowser.Input.Touch

Assembly: DotNetBrowser.dll

A service that can be used for touch input handling.

```
public interface ITouch : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Canceled
: Occurs when the touch has been canceled.

Ended
: Occurs when the touch has been ended.

Moved
: Occurs when the touch has moved.

Started
: Occurs when the touch has been started.

# Property Canceled

Namespace: DotNetBrowser.Input.Touch

Assembly: DotNetBrowser.dll

## Canceled

Occurs when the touch has been canceled.

```
IInterceptableInputEvent<ITouchEventArgs, TouchCanceledEventArgs> Canceled { get; }
```

## Property Value

IInterceptableInputEvent<ITouchEventArgs, TouchCanceledEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property Ended

Namespace: DotNetBrowser.Input.Touch

Assembly: DotNetBrowser.dll

## Ended

Occurs when the touch has been ended.

```
IInterceptableInputEvent<ITouchEventArgs, TouchEndedEventArgs> Ended { get; }
```

## Property Value

IInterceptableInputEvent<ITouchEventArgs, TouchEndedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property Moved

Namespace: DotNetBrowser.Input.Touch

Assembly: DotNetBrowser.dll

## Moved

Occurs when the touch has moved.

```
IInterceptableInputEvent<ITouchEventArgs, TouchMovedEventArgs> Moved { get; }
```

## Property Value

IInterceptableInputEvent<ITouchEventArgs, TouchMovedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Property Started

Namespace: DotNetBrowser.Input.Touch

Assembly: DotNetBrowser.dll

## Started

Occurs when the touch has been started.

```
IInterceptableInputEvent<ITouchEventArgs, TouchStartedEventArgs> Started { get; }
```

## Property Value

IInterceptableInputEvent<ITouchEventArgs, TouchStartedEventArgs>

## Exceptions

ObjectDisposedException

The IBrowser object has already been disposed.

# Namespace DotNetBrowser.Input.Touch.Events

## Classes

TouchCanceledEventArgs

TouchEndedEventArgs

TouchEventArgs

TouchMovedEventArgs

TouchPoint

TouchStartedEventArgs

## Interfaces

ITouchEventArgs
: Event arguments for the ITouch events associated with a surface touch.

ITouchPoint
: A single contact point on a touch-sensitive device.

## Enums

TouchState
: The possible states of the touch point.

# Interface ITouchEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

Event arguments for the ITouch events associated with a surface touch.

```
public interface ITouchEventArgs
```

## Properties

KeyModifiers

Gets the keyboard modifiers.

TouchPoints

Gets the list of all detected touch points.

# Property KeyModifiers

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## KeyModifiers

Gets the keyboard modifiers.

```
IKeyModifiers KeyModifiers { get; }
```

## Property Value

IKeyModifiers

The keyboard modifiers applied.

# Property TouchPoints

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchPoints

Gets the list of all detected touch points.

```
IReadOnlyList<ITouchPoint> TouchPoints { get; }
```

## Property Value

IReadOnlyList<ITouchPoint>

The list of all detected touch points.

# Interface ITouchPoint

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

A single contact point on a touch-sensitive device.

```
public interface ITouchPoint
```

## Properties

Force
: Gets the amount of pressure being applied to the surface.

Id
: Gets the touch point unique identifier.

LocationOnScreen
: Gets the touch point location relative to the bounds of the screen.

LocationOnWidget
: Gets the touch point location relative to the bounds of the widget.

TouchEllipse
: Gets the touch ellipse that most closely circumscribes the area of contact with the screen.

TouchState
: Gets the current state of touch.

# Property Force

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## Force

Gets the amount of pressure being applied to the surface.

```
float? Force { get; }
```

## Property Value

float?

A float value between 0.0 (no pressure) and 1.0 (maximum pressure) or null if not supported.

## Remarks

Can be null if force detection is not supported.

# Property Id

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## Id

Gets the touch point unique identifier.

```
int Id { get; }
```

## Property Value

int

The touch point unique identifier.

## Remarks

Every touch has a constant identifier while a finger touches the surface and it stay the same during of its holding or movement around the surface. Removing any count of touches from the surface doesn't change the identifiers of remaining touches.

# Property LocationOnScreen

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## LocationOnScreen

Gets the touch point location relative to the bounds of the screen.

```
Point LocationOnScreen { get; }
```

## Property Value

Point

The touch point location relative to the bounds of the screen.

# Property LocationOnWidget

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## LocationOnWidget

Gets the touch point location relative to the bounds of the widget.

```
Point LocationOnWidget { get; }
```

## Property Value

Point

The touch point location relative to the bounds of the widget.

# Property TouchEllipse

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchEllipse

Gets the touch ellipse that most closely circumscribes the area of contact with the screen.

```
Ellipse TouchEllipse { get; }
```

## Property Value

Ellipse

## Remarks

It consists from the major X axis length, the major Y axis length and the touch ellipse rotation angle. Together, these three values describe an ellipse that approximates the size and shape of the area of contact between the user and the screen. The rotation angle is an angle, in degrees, of the contact area ellipse defined by RadiusX and RadiusY. The value may be between 0 and 90.

# Property TouchState

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchState

Gets the current state of touch.

```
TouchState TouchState { get; }
```

## Property Value

TouchState

The current state of touch.

# Class TouchCanceledEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

```
public sealed class TouchCanceledEventArgs : TouchEventArgs, ITouchEventArgs
```

### Inheritance

object ← TouchEventArgs ← TouchCanceledEventArgs

### Implements

ITouchEventArgs

### Inherited Members

TouchEventArgs.KeyModifiers , TouchEventArgs.TouchPoints , TouchEventArgs.Equals(object) , TouchEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

TouchCanceledEventArgs(params TouchPoint[])

Initializes a new instance of the TouchCanceledEventArgs with the specified touch points.

TouchCanceledEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of the TouchCanceledEventArgs with the specified touch points.

# Constructor TouchCanceledEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchCanceledEventArgs(params TouchPoint[])

Initializes a new instance of the TouchCanceledEventArgs with the specified touch points.

```
public TouchCanceledEventArgs(params TouchPoint[] points)
```

### Parameters

`points` TouchPoint[]

The touch points to add.

### Exceptions

InvalidEnumArgumentException

No touch points with the required state are passed.

ArgumentNullException

The `points` is null.

## TouchCanceledEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of the TouchCanceledEventArgs with the specified touch points.

```
public TouchCanceledEventArgs(IEnumerable<TouchPoint> points)
```

### Parameters

`points` IEnumerable<TouchPoint>

The touch points to add.

## Exceptions

InvalidEnumArgumentException

If no points with the required state are passed.

ArgumentNullException

If `points` is null.

# Class TouchEndedEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

```
public sealed class TouchEndedEventArgs : TouchEventArgs, ITouchEventArgs
```

### Inheritance

object ← TouchEventArgs ← TouchEndedEventArgs

### Implements

ITouchEventArgs

### Inherited Members

TouchEventArgs.KeyModifiers , TouchEventArgs.TouchPoints , TouchEventArgs.Equals(object) , TouchEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

TouchEndedEventArgs(params TouchPoint[])

Initializes a new instance of the TouchEndedEventArgs with the specified touch points.

TouchEndedEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of the TouchEndedEventArgs with the specified touch points.

# Constructor TouchEndedEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchEndedEventArgs(params TouchPoint[])

Initializes a new instance of the TouchEndedEventArgs with the specified touch points.

```
public TouchEndedEventArgs(params TouchPoint[] points)
```

### Parameters

`points` TouchPoint[]

The touch points to add.

### Exceptions

InvalidEnumArgumentException

No touch points with the required state are passed.

ArgumentNullException

The `points` is null.

## TouchEndedEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of the TouchEndedEventArgs with the specified touch points.

```
public TouchEndedEventArgs(IEnumerable<TouchPoint> points)
```

### Parameters

`points` IEnumerable<TouchPoint>

The touch points to add.

## Exceptions

InvalidEnumArgumentException

No touch points with the required state are passed.

ArgumentNullException

The `points` is null.

# Class TouchEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

```
public abstract class TouchEventArgs : ITouchEventArgs
```

**Inheritance**

object ← TouchEventArgs

**Implements**

ITouchEventArgs

**Derived**

TouchCanceledEventArgs, TouchEndedEventArgs, TouchMovedEventArgs, TouchStartedEventArgs

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

TouchEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of TouchEventArgs.

## Properties

KeyModifiers

Gets the keyboard modifiers.

TouchPoints

Gets the list of all detected touch points.

## Methods

Equals(object)

GetHashCode()

# Constructor TouchEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of TouchEventArgs.

```
protected TouchEventArgs(IEnumerable<TouchPoint> touchPoints)
```

## Parameters

`touchPoints` IEnumerable<TouchPoint>

The touch points to add.

## Exceptions

InvalidEnumArgumentException

No touch points with the required state are passed.

# Property KeyModifiers

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## KeyModifiers

Gets the keyboard modifiers.

```
public IKeyModifiers KeyModifiers { get; set; }
```

## Property Value

IKeyModifiers

The keyboard modifiers applied.

# Property TouchPoints

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchPoints

Gets the list of all detected touch points.

```
public IReadOnlyList<ITouchPoint> TouchPoints { get; }
```

## Property Value

IReadOnlyList<ITouchPoint>

The list of all detected touch points.

# Method Equals

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class TouchMovedEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

```
public sealed class TouchMovedEventArgs : TouchEventArgs, ITouchEventArgs
```

**Inheritance**

object ← TouchEventArgs ← TouchMovedEventArgs

**Implements**

ITouchEventArgs

**Inherited Members**

TouchEventArgs.KeyModifiers , TouchEventArgs.TouchPoints , TouchEventArgs.Equals(object) , TouchEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

TouchMovedEventArgs(params TouchPoint[])

Initializes a new instance of the TouchMovedEventArgs with the specified touch points.

TouchMovedEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of the TouchMovedEventArgs with the specified touch points.

# Constructor TouchMovedEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchMovedEventArgs(params TouchPoint[])

Initializes a new instance of the TouchMovedEventArgs with the specified touch points.

```
public TouchMovedEventArgs(params TouchPoint[] points)
```

### Parameters

`points` TouchPoint[]

The touch points to add.

### Exceptions

InvalidEnumArgumentException

No touch points with the required state are passed.

ArgumentNullException

The `points` is null.

## TouchMovedEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of the TouchMovedEventArgs with the specified touch points.

```
public TouchMovedEventArgs(IEnumerable<TouchPoint> points)
```

### Parameters

`points` IEnumerable<TouchPoint>

The touch points to add.

## Exceptions

InvalidEnumArgumentException

No touch points with the required state are passed.

ArgumentNullException

The `points` is null.

# Class TouchPoint

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

```
public sealed class TouchPoint : ITouchPoint
```

**Inheritance**

object ← TouchPoint

**Implements**

ITouchPoint

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

TouchPoint(int, TouchState, Point)

Initializes a new instance of TouchPoint with the specified identifier, state, and location.

## Properties

Force

Gets the amount of pressure being applied to the surface.

Id

Gets the touch point unique identifier.

LocationOnScreen

Gets the touch point location relative to the bounds of the screen.

LocationOnWidget

Gets or sets the position relative to the bounds of the widget.

TouchEllipse

Gets or sets the touch ellipse that most closely circumscribes the area of contact with the screen.

TouchState

Gets the current state of touch.

## Methods

Equals(object)

GetHashCode()

# Constructor TouchPoint

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchPoint(int, TouchState, Point)

Initializes a new instance of TouchPoint with the specified identifier, state, and location.

```
public TouchPoint(int id, TouchState state, Point locationOnWidget)
```

## Parameters

`id` int

The touch point identifier. If this instance denotes a new touch point, its identifier should be a unique integer value.If this instance denotes an already existing point that we want to move, end or cancel, its identifier should be equal to the identifier of that existing touch point.

`state` TouchState

The state of the touch point.

`locationOnWidget` Point

The position relative to the bounds of the widget.

## Exceptions

ArgumentNullException

The `locationOnWidget` is null.

# Property Force

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## Force

Gets the amount of pressure being applied to the surface.

```
public float? Force { get; set; }
```

## Property Value

float?

A float value between 0.0 (no pressure) and 1.0 (maximum pressure) or null if not supported.

## Remarks

Can be null if force detection is not supported.

# Property Id

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## Id

Gets the touch point unique identifier.

```
public int Id { get; }
```

## Property Value

int

The touch point unique identifier.

## Remarks

Every touch has a constant identifier while a finger touches the surface and it stay the same during of its holding or movement around the surface. Removing any count of touches from the surface doesn't change the identifiers of remaining touches.

# Property LocationOnScreen

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## LocationOnScreen

Gets the touch point location relative to the bounds of the screen.

```
public Point LocationOnScreen { get; set; }
```

## Property Value

Point

The touch point location relative to the bounds of the screen.

# Property LocationOnWidget

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## LocationOnWidget

Gets or sets the position relative to the bounds of the widget.

```
public Point LocationOnWidget { get; set; }
```

## Property Value

Point

The position relative to the bounds of the widget.

## Exceptions

ArgumentNullException

The LocationOnWidget can't be null.

# Property TouchEllipse

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchEllipse

Gets or sets the touch ellipse that most closely circumscribes the area of contact with the screen.

```
public Ellipse TouchEllipse { get; set; }
```

## Property Value

Ellipse

## Remarks

The touch ellipse approximates the size and shape of the area of contact between the user and the screen. It consists from the major X axis length, the major Y axis length and the touch ellipse rotation angle. The rotation angle is an angle, in degrees, of the contact area ellipse defined by RadiusX and RadiusY. The value may be between 0 and 90.

## Exceptions

ArgumentNullException

The TouchEllipse can't be null.

ArgumentOutOfRangeException

The angle of touch ellipse out of range 0-90 degrees.

# Property TouchState

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchState

Gets the current state of touch.

```
public TouchState TouchState { get; set; }
```

## Property Value

TouchState

The current state of touch.

# Method Equals

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class TouchStartedEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

```
public sealed class TouchStartedEventArgs : TouchEventArgs, ITouchEventArgs
```

### Inheritance

object ← TouchEventArgs ← TouchStartedEventArgs

### Implements

ITouchEventArgs

### Inherited Members

TouchEventArgs.KeyModifiers , TouchEventArgs.TouchPoints , TouchEventArgs.Equals(object) , TouchEventArgs.GetHashCode() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

TouchStartedEventArgs(params TouchPoint[])

Initializes a new instance of the TouchStartedEventArgs with the specified touch points.

TouchStartedEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of the TouchStartedEventArgs with the specified touch points.

# Constructor TouchStartedEventArgs

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

## TouchStartedEventArgs(params TouchPoint[])

Initializes a new instance of the TouchStartedEventArgs with the specified touch points.

```
public TouchStartedEventArgs(params TouchPoint[] points)
```

### Parameters

`points` TouchPoint[]

The touch points to add.

### Exceptions

InvalidEnumArgumentException

No touch points with the required state are passed.

ArgumentNullException

The `points` is null.

## TouchStartedEventArgs(IEnumerable<TouchPoint>)

Initializes a new instance of the TouchStartedEventArgs with the specified touch points.

```
public TouchStartedEventArgs(IEnumerable<TouchPoint> points)
```

### Parameters

`points` IEnumerable<TouchPoint>

The touch points to add.

## Exceptions

InvalidEnumArgumentException

No touch points with the required state are passed.

ArgumentNullException

The `points` is null.

# Enum TouchState

Namespace: DotNetBrowser.Input.Touch.Events

Assembly: DotNetBrowser.dll

The possible states of the touch point.

```
public enum TouchState
```

## Fields

`Canceled = 6`

The touch has been canceled.

`Ended = 3`

The touch has ended.

`Moved = 4`

The touch point has moved.

`Started = 2`

The touch has started.

`Stationary = 5`

The touch point didn't change its state.

`TouchStateUnspecified = 0`

Reserved value.

`Undefined = 1`

The touch point state is undefined.

# Namespace DotNetBrowser.Js

## Classes

JsException
: Thrown when an exception is raised in JavaScript.

JsonExtensions
: Contains methods for working with JSON strings

## Interfaces

IJsFunction
: A JavaScript function that can be passed between .NET and JavaScript as a method argument or a return value. The function lifetime is bound to the lifetime of the frame this function belongs to. When the owner frame is unloaded, all the JavaScript objects are automatically disposed. An attempt to access a disposed JavaScript object will result in ObjectDisposedException

IJsObject
: Represents a JavaScript object. Provides access to the object's properties and functions. The JavaScript object is alive until its JavaScript execution context exist. Once execution context is disposed, all JavaScript objects available in the scope of this context will be automatically disposed. If you try to access already disposed object, you will get an ObjectDisposedException.

IJsObjectPropertyCollection
: The JavaScript object properties.

IJsPromise
: The JavaScript Promise.

# Interface IJsFunction

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

A JavaScript function that can be passed between .NET and JavaScript as a method argument or a return value. The function lifetime is bound to the lifetime of the frame this function belongs to. When the owner frame is unloaded, all the JavaScript objects are automatically disposed. An attempt to access a disposed JavaScript object will result in ObjectDisposedException

```
public interface IJsFunction : IJsObject, IAutoDisposable
```

**Inherited Members**
IJsObject.Frame , IJsObject.Properties , IJsObject.Invoke<T>(string, params object[]) , IJsObject.Invoke(string, params object[]) , IJsObject.InvokeAsync(string, params object[]) , IJsObject.InvokeAsync<T>(string, params object[]) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

**Extension Methods**
JsonExtensions.ToJsonString(IJsObject)

## Remarks

This API is available since DotNetBrowser 2.1.

## Methods

Invoke(IJsObject, params object[])
Executes the function on the given `jsObject` with the `args`. This method blocks current thread execution until the function finishes its execution. If the function raises an exception, then JsException with an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

InvokeAsync(IJsObject, params object[])
Asynchronously executes the function on the given `jsObject` with the `args` without blocking the current thread.

InvokeAsync<T>(IJsObject, params object[])

Asynchronously executes the function on the given `jsObject` with the `args` without blocking the current thread.

Invoke<T>(IJsObject, params object[])

Executes the function on the given `jsObject` with the `args`. This method blocks current thread execution until the function finishes its execution. If the function raises an exception, then a Js Exception with an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

# Method Invoke

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Invoke<T>(IJsObject, params object[])

Executes the function on the given `jsObject` with the `args`. This method blocks current thread execution until the function finishes its execution. If the function raises an exception, then a JsException with an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

```
T Invoke<T>(IJsObject jsObject, params object[] args)
```

### Parameters

`jsObject` IJsObject

The JavaScript object to invoke this function on. Pass `null` to invoke the function as a global function.

`args` object[]

The list of input arguments. The following types are supported: bool, double, string, IJsObject, or object.

### Returns

T

The result of the JavaScript function execution.

### Type Parameters

`T`

The expected type of the result of the JavaScript function execution

## Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsFunction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## Invoke(IJsObject, params object[])

Executes the function on the given `jsObject` with the `args`. This method blocks current thread execution until the function finishes its execution. If the function raises an exception, then JsException with an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

```
object Invoke(IJsObject jsObject, params object[] args)
```

## Parameters

`jsObject` IJsObject

The JavaScript object to invoke this function on. Pass `null` to invoke the function as a global function.

`args` object[]

The list of input arguments. The following types are supported: bool, double, string, IJsObject, or object.

## Returns

object

The result of the JavaScript function execution.

## Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsFunction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method InvokeAsync

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## InvokeAsync(IJsObject, params object[])

Asynchronously executes the function on the given `jsObject` with the `args` without blocking the current thread.

```
Task<object> InvokeAsync(IJsObject jsObject, params object[] args)
```

### Parameters

`jsObject` IJsObject

The JavaScript object to invoke this function on. Pass `null` to invoke the function as a global function.

`args` object[]

The list of input arguments. The following types are supported: bool, double, string, IJsObject, or object.

### Returns

Task<object>

The task that can be used to wait for completion and obtain the result of the JavaScript function execution.

### Remarks

If the function raises an exception, the task will complete with JsException containing an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

### Exceptions

ObjectDisposedException

The IJsFunction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## InvokeAsync<T>(IJsObject, params object[])

Asynchronously executes the function on the given `jsObject` with the `args` without blocking the current thread.

```
Task<T> InvokeAsync<T>(IJsObject jsObject, params object[] args)
```

### Parameters

`jsObject` IJsObject

The JavaScript object to invoke this function on. Pass `null` to invoke the function as a global function.

`args` object[]

The list of input arguments. The following types are supported: bool, double, string, IJsObject, or object.

### Returns

Task<T>

The task that can be used to wait for completion and obtain the result of the JavaScript function execution.

### Type Parameters

`T`

The expected type of the result of the JavaScript function execution.

## Remarks

If the function raises an exception, the task will complete with JsException containing an error message that describes the reason of the exception. Same error message will be printed in JavaScript Console.

## Exceptions

ObjectDisposedException

The IJsFunction has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Interface IJsObject

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

Represents a JavaScript object. Provides access to the object's properties and functions. The JavaScript object is alive until its JavaScript execution context exist. Once execution context is disposed, all JavaScript objects available in the scope of this context will be automatically disposed. If you try to access already disposed object, you will get an ObjectDisposedException.

```
public interface IJsObject : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

**Extension Methods**

JsonExtensions.ToJsonString(IJsObject)

## Properties

Frame

Gets the IFrame instance this JavaScript object is bound to.

Properties

Gets the object properties as a live collection. Modifying this collection will lead to updating the object properties.

## Methods

Invoke(string, params object[])

Executes the function with the given `methodName` and the `args` in the JavaScript object. This method blocks current thread execution until the function finishes its execution. If the function raises an exception, then JsException with an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

InvokeAsync(string, params object[])

Asynchronously executes the function with the given `methodName` and the `args` in the JavaScript object without blocking the current thread.

InvokeAsync<T>(string, params object[])

Asynchronously executes the function with the given `methodName` and the `args` in the JavaScript object without blocking the current thread.

Invoke<T>(string, params object[])

Executes the function with the given `methodName` and the `args` in the JavaScript object. This method blocks current thread execution until the function finishes its execution. If the function raises an exception, then a JsException with an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

# Property Frame

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Frame

Gets the IFrame instance this JavaScript object is bound to.

```
IFrame Frame { get; }
```

## Property Value

IFrame

The IFrame instance this JavaScript object is bound to.

# Property Properties

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Properties

Gets the object properties as a live collection. Modifying this collection will lead to updating the object properties.

```
IJsObjectPropertyCollection Properties { get; }
```

## Property Value

IJsObjectPropertyCollection

The object properties as a live collection.

# Method Invoke

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Invoke<T>(string, params object[])

Executes the function with the given `methodName` and the `args` in the JavaScript object. This method blocks current thread execution until the function finishes its execution. If the function raises an exception, then a JsException with an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

```
T Invoke<T>(string methodName, params object[] args)
```

### Parameters

`methodName` string

The function name.

`args` object[]

the list of input arguments. The following types are supported: bool, double, string, IJsObject, or object.

### Returns

T

The result of the JavaScript function execution.

### Type Parameters

`T`

The expected type of the result of the JavaScript function execution

## Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## Invoke(string, params object[])

Executes the function with the given `methodName` and the `args` in the JavaScript object. This method blocks current thread execution until the function finishes its execution. If the function raises an exception, then JsException with an error message that describes the reason of the exception will be thrown. Same error message will be printed in JavaScript Console.

```
object Invoke(string methodName, params object[] args)
```

## Parameters

`methodName` string

The function name.

`args` object[]

the list of input arguments. The following types are supported: bool, double, string, IJsObject, or object.

## Returns

object

The result of the JavaScript function execution.

## Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method InvokeAsync

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## InvokeAsync(string, params object[])

Asynchronously executes the function with the given `methodName` and the `args` in the JavaScript object without blocking the current thread.

```
Task<object> InvokeAsync(string methodName, params object[] args)
```

### Parameters

`methodName` string

The function name.

`args` object[]

The list of input arguments. The following types are supported: bool, double, string, IJsObject, or object.

### Returns

Task<object>

The task that can be used to wait for completion and obtain the result of the JavaScript function execution.

### Remarks

If the function raises an exception, the task will complete with JsException containing an error message that describes the reason of the exception. Same error message will be printed in JavaScript Console.

### Exceptions

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## InvokeAsync<T>(string, params object[])

Asynchronously executes the function with the given `methodName` and the `args` in the JavaScript object without blocking the current thread.

```
Task<T> InvokeAsync<T>(string methodName, params object[] args)
```

### Parameters

`methodName` string

The function name.

`args` object[]

The list of input arguments. The following types are supported: bool, double, string, IJsObject, or object.

### Returns

Task<T>

The task that can be used to wait for completion and obtain the result of the JavaScript function execution.

### Type Parameters

`T`

The expected type of the result of the JavaScript function execution.

## Remarks

If the function raises an exception, the task will complete with JsException containing an error message that describes the reason of the exception. Same error message will be printed in JavaScript Console.

## Exceptions

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Interface IJsObjectPropertyCollection

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

The JavaScript object properties.

```
public interface IJsObjectPropertyCollection : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

this[string]
Gets or sets a property with the specified name in the current JavaScript object. The `name` parameter represents JavaScript object's property name.

this[uint]
Gets or sets a property with the specified index in the current JavaScript object. The `index` parameter represents JavaScript object's property index.

Names
Gets a collection containing the names of the properties of this object, including properties from prototype objects.

OwnPropertyNames
Gets the names of the properties of this object, excluding properties from prototype objects.

## Methods

Contains(string)
Checks whether the JavaScript object has a property or function with the specified `name`.

Contains(uint)
Checks whether the JavaScript object has a property with the specified `index`.

Remove(string)

Removes a property with the specified `name` from the JavaScript object. Once you remove the property, it will not be available in the current JavaScript object anymore.

Remove(uint)

Removes a property with the specified `index` in the JavaScript object. Once you remove the property, it will not be available in the current JavaScript object anymore.

# Property Names

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Names

Gets a collection containing the names of the properties of this object, including properties from prototype objects.

```
IEnumerable<string> Names { get; }
```

## Property Value

IEnumerable<string>

The collection containing the names of the properties of this object, including properties from prototype objects.

## Remarks

The returned enumerable will enumerate the names in the same way as they would be enumerated by a for-in statement over this object in JavaScript.

## Exceptions

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property OwnPropertyNames

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## OwnPropertyNames

Gets the names of the properties of this object, excluding properties from prototype objects.

```
IEnumerable<string> OwnPropertyNames { get; }
```

## Property Value

IEnumerable<string>

The collection containing the names of the properties of this object, both enumerable and non-enumerable, excluding properties from prototype objects.

## Exceptions

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property this

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## this[string]

Gets or sets a property with the specified name in the current JavaScript object. The `name` parameter represents JavaScript object's property name.

```
object this[string name] { get; set; }
```

### Parameters

`name` string

The name of the property. Cannot be null or empty.

### Property Value

object

The value associated with the specified name.

### Exceptions

ArgumentException

The `name` is null, empty, or contains only blank characters.

ArgumentException

The `value` is an IJsObject with a different JavaScript context.

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## this[uint]

Gets or sets a property with the specified index in the current JavaScript object. The `index` parameter represents JavaScript object's property index.

```
object this[uint index] { get; set; }
```

### Parameters

`index` uint

The index of the property. Cannot be null or empty.

### Property Value

object

The value associated with the specified name.

### Exceptions

ArgumentException

The `value` is an IJsObject with a different JavaScript context.

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method Contains

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Contains(string)

Checks whether the JavaScript object has a property or function with the specified `name`.

```
bool Contains(string name)
```

### Parameters

`name` string

The name of the property or function.

### Returns

bool

`true` if the object has a property or function with the specified `name`.

### Exceptions

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## Contains(uint)

Checks whether the JavaScript object has a property with the specified `index`.

```
bool Contains(uint index)
```

## Parameters

`index` uint

The index of the property.

## Returns

bool

`true` if the object has a property with the given `index`.

## Exceptions

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method Remove

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Remove(string)

Removes a property with the specified `name` from the JavaScript object. Once you remove the property, it will not be available in the current JavaScript object anymore.

```
bool Remove(string name)
```

### Parameters

`name` string

The name of the property or function.

### Returns

bool

`true` if the property was successfully removed.

### Exceptions

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## Remove(uint)

Removes a property with the specified `index` in the JavaScript object. Once you remove the property, it will not be available in the current JavaScript object anymore.

```
bool Remove(uint index)
```

## Parameters

`index` uint

The index of the property or function.

## Returns

bool

`true` if the property was successfully removed.

## Exceptions

ObjectDisposedException

The IJsObject has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Interface IJsPromise

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

The JavaScript Promise.

```
public interface IJsPromise : IJsObject, IAutoDisposable
```

**Inherited Members**
IJsObject.Frame , IJsObject.Properties , IJsObject.Invoke<T>(string, params object[]) , IJsObject.Invoke(string, params object[]) , IJsObject.InvokeAsync(string, params object[]) , IJsObject.InvokeAsync<T>(string, params object[]) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

**Extension Methods**
JsonExtensions.ToJsonString(IJsObject)

## Remarks

The IJsPromise interface can be used to register .NET delegates as Promise handlers.

## Methods

Catch(Action<object>)
Appends the rejection handler to the promise.

Catch(Func<object, object>)
Appends the rejection handler to the promise.

Finally(Action<object>)
Appends the handler that will be invoked if the promise is settled (fulfilled/rejected).

Finally(Func<object, object>)
Appends the handler that will be invoked if the promise is settled (fulfilled/rejected).

Then(Action<object>, Action<object>)
Appends fulfillment and rejection handlers to the promise.

Then(Func<object, object>, Func<object, object>)

Appends fulfillment and rejection handlers to the promise.

# Method Catch

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Catch(Action<object>)

Appends the rejection handler to the promise.

```
IJsPromise Catch(Action<object> onRejected)
```

## Parameters

`onRejected` Action<object>

The rejection handler.

## Returns

IJsPromise

A new IJsPromise resolving to the return value of the called handler, or to its original settled value if the promise was not handled.

## Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsPromise has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## Catch(Func<object, object>)

Appends the rejection handler to the promise.

```
IJsPromise Catch(Func<object, object> onRejected)
```

### Parameters

`onRejected` Func<object, object>

The rejection handler.

### Returns

IJsPromise

A new IJsPromise resolving to the return value of the called handler, or to its original settled value if the promise was not handled.

### Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsPromise has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method Finally

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Finally(Action<object>)

Appends the handler that will be invoked if the promise is settled (fulfilled/rejected).

```
IJsPromise Finally(Action<object> onFinally)
```

## Parameters

`onFinally` Action<object>

The rejection handler.

## Returns

IJsPromise

A new IJsPromiseresolving to the return value of the called handler, or to its original settled value if the promise was not handled.

## Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsPromise has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## Finally(Func<object, object>)

Appends the handler that will be invoked if the promise is settled (fulfilled/rejected).

```
IJsPromise Finally(Func<object, object> onFinally)
```

### Parameters

`onFinally` Func<object, object>

The rejection handler.

### Returns

IJsPromise

A new IJsPromise resolving to the return value of the called handler, or to its original settled value if the promise was not handled.

### Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsPromise has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Method Then

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## Then(Action<object>, Action<object>)

Appends fulfillment and rejection handlers to the promise.

```
IJsPromise Then(Action<object> onFulfilled, Action<object> onRejected = null)
```

## Parameters

`onFulfilled` Action<object>

The fulfillment handler.

`onRejected` Action<object>

The rejection handler.

## Returns

IJsPromise

A new IJsPromise resolving to the return value of the called handler, or to its original settled value if the promise was not handled.

## Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsPromise has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

## Then(Func<object, object>, Func<object, object>)

Appends fulfillment and rejection handlers to the promise.

```
IJsPromise Then(Func<object, object> onFulfilled, Func<object, object> onRejected = null)
```

### Parameters

`onFulfilled` Func<object, object>

The fulfillment handler.

`onRejected` Func<object, object>

The rejection handler.

### Returns

IJsPromise

A new IJsPromise resolving to the return value of the called handler, or to its original settled value if the promise was not handled.

### Exceptions

JsException

The JavaScript function raised an exception.

ObjectDisposedException

The IJsPromise has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Class JsException

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

Thrown when an exception is raised in JavaScript.

```
public sealed class JsException : Exception, ISerializable
```

### Inheritance

object ← Exception ← JsException

### Implements

ISerializable

### Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

# Class JsonExtensions

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

Contains methods for working with JSON strings

```
public static class JsonExtensions
```

**Inheritance**

object ← JsonExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

ParseJsonString(IFrame, string)

Creates an object that represents the result of parsing the given string in JSON format.

ParseJsonString<T>(IFrame, string)

Creates an object that represents the result of parsing the given string in JSON format.

ToJsonString(IJsObject)

Converts the given JavaScript object into a JSON string.

# Method ParseJsonString

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## ParseJsonString<T>(IFrame, string)

Creates an object that represents the result of parsing the given string in JSON format.

```
public static T ParseJsonString<T>(this IFrame frame, string jsonString)
```

## Parameters

`frame` IFrame

The frame to use for conversion.

`jsonString` string

A string in JSON format.

## Returns

T

The result of parsing. Can be `null` if the result of parsing is `null`. For example, when the given JSON string is "null", then the method returns `null`

## Type Parameters

`T`

The expected result type. The following types are supported: bool, double, string, IJsObject, or object.

## Exceptions

JsException

The given JSON string object cannot be parsed.

## ParseJsonString(IFrame, string)

Creates an object that represents the result of parsing the given string in JSON format.

```
public static object ParseJsonString(this IFrame frame, string jsonString)
```

### Parameters

`frame` IFrame

The frame to use for conversion.

`jsonString` string

A string in JSON format.

### Returns

object

The result of parsing. Can be `null` if the result of parsing is `null`. For example, when the given JSON string is "null", then the method returns `null`

### Exceptions

JsException

The given JSON string object cannot be parsed.

# Method ToJsonString

Namespace: DotNetBrowser.Js

Assembly: DotNetBrowser.dll

## ToJsonString(IJsObject)

Converts the given JavaScript object into a JSON string.

```
public static string ToJsonString(this IJsObject jsObject)
```

### Parameters

`jsObject` IJsObject

The JavaScript object to convert.

### Returns

string

A JSON representation of the given JavaScript object.

### Exceptions

JsException

The given JavaScript object cannot be converted to a JSON string. For example, a circular reference was found in the given JavaScript object.

# Namespace DotNetBrowser.Js.Collections

## Interfaces

IJsArray
: A JavaScript array.

IJsArrayBuffer
: A JavaScript array buffer.

IJsMap
: A JavaScript map.

IJsSet
: A JavaScript set.

# Interface IJsArray

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

A JavaScript array.

```
public interface IJsArray : IJsObject, IAutoDisposable
```

**Inherited Members**
IJsObject.Frame , IJsObject.Properties , IJsObject.Invoke<T>(string, params object[]) , IJsObject.Invoke(string, params object[]) , IJsObject.InvokeAsync(string, params object[]) , IJsObject.InvokeAsync<T>(string, params object[]) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

**Extension Methods**
JsonExtensions.ToJsonString(IJsObject)

## Remarks

An array object can be passed between .NET and JavaScript as a method argument or a return value.The object lifetime is bound to the lifetime of the frame this object belongs to. When the owner frame is unloaded, all the bound JavaScript objects are automatically disposed.

## Properties

Count
: Gets the number of elements contained in the IJsArray.

this[uint]
: Gets or sets the array element at the specified `index`. If there is an existing element at the specified `index`, it will be replaced. If the `index` exceeds the array size, the array will be extended and the elements between the last inserted item and the `index` will be set to `null`.

## Methods

ToReadOnlyList()

Copies the contents of the IJsArray to a new IReadOnlyList<T>.

ToReadOnlyList<T>()

Copies the contents of the IJsArray to a new IReadOnlyList<T>.

# Property Count

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Count

Gets the number of elements contained in the IJsArray.

```
ulong Count { get; }
```

## Property Value

ulong

The number of elements contained in the IJsArray.

## Remarks

The maximum size of an array in JavaScript equals to 2^32-1

## Exceptions

ObjectDisposedException

The IJsArray has already been disposed.

# Property this

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## this[uint]

Gets or sets the array element at the specified `index`. If there is an existing element at the specified `index`, it will be replaced. If the `index` exceeds the array size, the array will be extended and the elements between the last inserted item and the `index` will be set to `null`.

```
object this[uint index] { get; set; }
```

### Parameters

`index` uint

The zero-based index of the element to get or set.

### Property Value

object

The element at the specified index. For the proxy objects, the corresponding injected .NET object is returned.

### Remarks

If you pass a non-primitive .NET object to JavaScript, it will be converted into a "proxy" JavaScript object. Method and property calls to this object will be delegated to the .NET object.

### Exceptions

ObjectDisposedException

The IJsArray has already been disposed.

# Method ToReadOnlyList

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## ToReadOnlyList()

Copies the contents of the IJsArray to a new IReadOnlyList<T>.

```
IReadOnlyList<object> ToReadOnlyList()
```

### Returns

IReadOnlyList<object>

A IReadOnlyList<T> that contains elements from the IJsArray.

### Remarks

Proxy objects are mapped to the corresponding injected .NET objects.

### Exceptions

ObjectDisposedException

The IJsArray has already been disposed.

## ToReadOnlyList<T>()

Copies the contents of the IJsArray to a new IReadOnlyList<T>.

```
IReadOnlyList<T> ToReadOnlyList<T>()
```

## Returns

IReadOnlyList<T>

A IReadOnlyList<T> that contains elements from the IJsArray.

## Type Parameters

T

## Remarks

Proxy objects are mapped to the corresponding injected .NET objects.

## Exceptions

ObjectDisposedException

The IJsArray has already been disposed.

# Interface IJsArrayBuffer

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

A JavaScript array buffer.

```
public interface IJsArrayBuffer
```

## Remarks

An array buffer object can be passed between .NET and JavaScript as a method argument or a return value.The object lifetime is bound to the lifetime of the frame this object belongs to.When the owner frame is unloaded, all the JavaScript objects are automatically disposed.

## Properties

Count

Gets the number of elements contained in the IJsArrayBuffer.

## Methods

ToByteArray()

Copies the contents of the IJsArrayBuffer to a new byte array.

# Property Count

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Count

Gets the number of elements contained in the IJsArrayBuffer.

```
long Count { get; }
```

## Property Value

long

The number of elements contained in the IJsArrayBuffer.

## Exceptions

ObjectDisposedException

The IJsArrayBuffer has already been disposed.

# Method ToByteArray

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## ToByteArray()

Copies the contents of the IJsArrayBuffer to a new byte array.

```
byte[] ToByteArray()
```

## Returns

byte[]

An array containing copy of the data in IJsArrayBuffer.

## Exceptions

ObjectDisposedException

The IJsArrayBuffer has already been disposed.

# Interface IJsMap

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

A JavaScript map.

```
public interface IJsMap : IJsObject, IAutoDisposable
```

**Inherited Members**
IJsObject.Frame , IJsObject.Properties , IJsObject.Invoke<T>(string, params object[]) , IJsObject.Invoke(string, params object[]) , IJsObject.InvokeAsync(string, params object[]) , IJsObject.InvokeAsync<T>(string, params object[]) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

**Extension Methods**
JsonExtensions.ToJsonString(IJsObject)

## Remarks

A map object can be passed between .NET and JavaScript as a method argument or a return value. The object lifetime is bound to the lifetime of the frame this object belongs to.When the owner frame is unloaded, all the bound JavaScript objects are automatically disposed.

## Properties

Count
: Gets the number of key-value mappings in this IJsMap.

this[object]
: Gets or sets the element with the specified `key`. If this map contains a value associated with the `key`, it will be replaced.

## Methods

Clear()

Removes all items from this IJsMap. Does nothing if the map is empty.

ContainsKey(object)

Determines whether the IJsMap contains an element with the specified `key`.

Remove(object)

Removes the element with the specified key from the IJsMap.

ToReadOnlyDictionary()

Copies the contents of the IJsMap to a new IReadOnlyDictionary<TKey, TValue>.

ToReadOnlyDictionary<TKey, TVal>()

Copies the contents of the IJsMap to a new IReadOnlyDictionary<TKey, TValue>.

# Property Count

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Count

Gets the number of key-value mappings in this IJsMap.

```
uint Count { get; }
```

## Property Value

uint

The number of key-value mappings in this IJsMap.

## Exceptions

ObjectDisposedException

The IJsMap has already been disposed.

# Property this

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## this[object]

Gets or sets the element with the specified `key`. If this map contains a value associated with the `key`, it will be replaced.

```
object this[object key] { get; set; }
```

### Parameters

`key` object

The key of the element to get or set.

### Property Value

object

The value with the specified `key` or `null` if there is no such key in this map.

### Exceptions

ObjectDisposedException

The IJsMap has already been disposed.

# Method Clear

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Clear()

Removes all items from this IJsMap. Does nothing if the map is empty.

```
void Clear()
```

## Exceptions

ObjectDisposedException

The IJsMap has already been disposed.

# Method ContainsKey

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## ContainsKey(object)

Determines whether the IJsMap contains an element with the specified `key`.

```
bool ContainsKey(object key)
```

## Parameters

`key` object

The key to locate in the IJsMap.

## Returns

bool

`true` if the IJsMap contains an element with the key; otherwise, `false`.

## Exceptions

ObjectDisposedException

The IJsMap has already been disposed.

# Method Remove

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Remove(object)

Removes the element with the specified key from the IJsMap.

```
bool Remove(object key)
```

## Parameters

`key` object

The key of the element to remove.

## Returns

bool

`true` if the element is successfully removed; otherwise, `false`. This method also returns `false` if key was not found in the IJsMap.

## Exceptions

ObjectDisposedException

The IJsMap has already been disposed.

# Method ToReadOnlyDictionary

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## ToReadOnlyDictionary<TKey, TVal>()

Copies the contents of the IJsMap to a new IReadOnlyDictionary<TKey, TValue>.

```
IReadOnlyDictionary<TKey, TVal> ToReadOnlyDictionary<TKey, TVal>()
```

### Returns

IReadOnlyDictionary<TKey, TVal>

A IReadOnlyDictionary<TKey, TValue> that contains key-value pairs from the IJsMap.

### Type Parameters

`TKey`

`TVal`

### Remarks

Proxy objects are mapped to the corresponding injected .NET objects.

### Exceptions

ObjectDisposedException

The IJsMap has already been disposed.

ArgumentException

The elements in this IJsMap cannot be converted to the specified type.

## ToReadOnlyDictionary()

Copies the contents of the IJsMap to a new IReadOnlyDictionary<TKey, TValue>.

```
IReadOnlyDictionary<object, object> ToReadOnlyDictionary()
```

### Returns

IReadOnlyDictionary<object, object>

A IReadOnlyDictionary<TKey, TValue> that contains key-value pairs from the IJsMap.

### Remarks

Proxy objects are mapped to the corresponding injected .NET objects.

### Exceptions

ObjectDisposedException

The IJsMap has already been disposed.

# Interface IJsSet

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

A JavaScript set.

```
public interface IJsSet : IJsObject, IAutoDisposable
```

**Inherited Members**
IJsObject.Frame , IJsObject.Properties , IJsObject.Invoke<T>(string, params object[]) , IJsObject.Invoke(string, params object[]) , IJsObject.InvokeAsync(string, params object[]) , IJsObject.InvokeAsync<T>(string, params object[]) , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

**Extension Methods**
JsonExtensions.ToJsonString(IJsObject)

## Remarks

A set object can be passed between .NET and JavaScript as a method argument or a return value. The object lifetime is bound to the lifetime of the frame this object belongs to. When the owner frame is unloaded, all the JavaScript objects are automatically disposed.

## Properties

Count
: Gets the number of elements contained in the IJsSet.

## Methods

Add(object)
: Adds an item to the IJsSet.

Clear()
: Removes all items from this IJsSet. Does nothing if the set is empty.

Contains(object)

Determines whether the IJsSet contains a specific value.

Remove(object)

Removes the specified `item` from this set.

ToReadOnlyCollection()

Copies the contents of the IJsSet to a new IReadOnlyCollection<T>.

ToReadOnlyCollection<T>()

Copies the contents of the IJsSet to a new IReadOnlyCollection<T>.

# Property Count

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Count

Gets the number of elements contained in the IJsSet.

```
ulong Count { get; }
```

## Property Value

ulong

The number of elements contained in the IJsSet.

## Exceptions

ObjectDisposedException

The IJsSet has already been disposed.

# Method Add

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Add(object)

Adds an item to the IJsSet.

```
void Add(object item)
```

## Parameters

`item` object

The item to the IJsSet.

## Exceptions

ArgumentException

The `item` is a IJsObject from a different web page or frame.

ObjectDisposedException

The IJsSet has already been disposed.

# Method Clear

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Clear()

Removes all items from this IJsSet. Does nothing if the set is empty.

```
void Clear()
```

## Exceptions

ObjectDisposedException

The IJsSet has already been disposed.

# Method Contains

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Contains(object)

Determines whether the IJsSet contains a specific value.

```
bool Contains(object item)
```

### Parameters

`item` object

The object to locate in the IJsSet.

### Returns

bool

`true` if item is found in the IJsSet; otherwise, `false`.

### Exceptions

ObjectDisposedException

The IJsSet has already been disposed.

# Method Remove

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## Remove(object)

Removes the specified `item` from this set.

```
bool Remove(object item)
```

## Parameters

`item` object

The item to be removed from this set.

## Returns

bool

`true` if item was successfully removed from the IJsSet; otherwise, `false`. This method also returns `false` if item is not found in the IJsSet.

## Exceptions

ObjectDisposedException

The IJsSet has already been disposed.

# Method ToReadOnlyCollection

Namespace: DotNetBrowser.Js.Collections

Assembly: DotNetBrowser.dll

## ToReadOnlyCollection<T>()

Copies the contents of the IJsSet to a new IReadOnlyCollection<T>.

```
IReadOnlyCollection<T> ToReadOnlyCollection<T>()
```

## Returns

IReadOnlyCollection<T>

A IReadOnlyCollection<T> that contains values from the IJsSet.

## Type Parameters

T

## Remarks

Proxy objects are mapped to the corresponding injected .NET objects.

## Exceptions

ObjectDisposedException

The IJsSet has already been disposed.

ArgumentException

The elements in this IJsSet cannot be converted to the specified type.

## ToReadOnlyCollection()

Copies the contents of the IJsSet to a new IReadOnlyCollection<T>.

```
IReadOnlyCollection<object> ToReadOnlyCollection()
```

### Returns

IReadOnlyCollection<object>

A IReadOnlyCollection<T> that contains values from the IJsSet.

### Remarks

Proxy objects are mapped to the corresponding injected .NET objects.

### Exceptions

ObjectDisposedException

The IJsSet has already been disposed.

ArgumentException

The elements in this IJsSet cannot be converted to the specified type.

# Namespace DotNetBrowser.Logging

## Classes

LoggerProvider

Provides access to different DotNetBrowser Loggers which are used to log Browser, IPC and Chromium process messages.

# Class LoggerProvider

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

Provides access to different DotNetBrowser Loggers which are used to log Browser, IPC and Chromium process messages.

```
public class LoggerProvider : IDisposable
```

**Inheritance**

object ← LoggerProvider

**Implements**

IDisposable

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ChromiumLogFile

The absolute or relative path to store Chromium log file.

ConsoleLoggingEnabled

Gets or sets a value indicating whether console logging is enabled.

FileLoggingEnabled

Gets or sets a value indicating whether file logging is enabled.

Instance

The LoggerProvider instance that can be used to configure DotNetBrowser logging.

Level

The minimal logging level of the messages that will appear in the logs. This value will be used to filter out log messages with lower logging level.

OutputFile

Gets or sets output file.

## Methods

Dispose()

Dispose object and managed resources.

Dispose(bool)

Dispose object and managed resources.

~LoggerProvider()

Destructor

# Property ChromiumLogFile

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

## ChromiumLogFile

The absolute or relative path to store Chromium log file.

```
public string ChromiumLogFile { get; set; }
```

## Property Value

string

# Property ConsoleLoggingEnabled

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

## ConsoleLoggingEnabled

Gets or sets a value indicating whether console logging is enabled.

```
public bool ConsoleLoggingEnabled { get; set; }
```

## Property Value

bool

# Property FileLoggingEnabled

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

## FileLoggingEnabled

Gets or sets a value indicating whether file logging is enabled.

```
public bool FileLoggingEnabled { get; set; }
```

## Property Value

bool

# Property Instance

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

## Instance

The LoggerProvider instance that can be used to configure DotNetBrowser logging.

```
public static LoggerProvider Instance { get; }
```

## Property Value

LoggerProvider

# Property Level

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

## Level

The minimal logging level of the messages that will appear in the logs. This value will be used to filter out log messages with lower logging level.

```
public SourceLevels Level { get; set; }
```

## Property Value

SourceLevels

# Property OutputFile

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

## OutputFile

Gets or sets output file.

```
public string OutputFile { get; set; }
```

## Property Value

string

# Method Dispose

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

## Dispose()

Dispose object and managed resources.

```
public void Dispose()
```

## Dispose(bool)

Dispose object and managed resources.

```
protected virtual void Dispose(bool disposing)
```

### Parameters

`disposing` bool

True if managed resources should be disposed

# Method ~LoggerProvider

Namespace: DotNetBrowser.Logging

Assembly: DotNetBrowser.Logging.dll

## ~LoggerProvider()

Destructor

```
protected ~LoggerProvider()
```

# Namespace DotNetBrowser.Media

## Classes

MediaDevice

The details of the media input device (audio/video).

## Interfaces

IAudio

Allows controlling audio on the loaded web page and receive notifications when audio has been started or stopped playing.

IMediaDevices

An engine service that allows accessing all the available media input devices.

## Enums

MediaDeviceType

The media device types.

MediaType

List of the known media types. It's used in ShowContextMenuParameters.

# Interface IAudio

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

Allows controlling audio on the loaded web page and receive notifications when audio has been started or stopped playing.

```
public interface IAudio : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Browser
: Gets the IBrowser instance associated with this object.

IsPlaying
: Indicates if the audio is currently playing on the loaded web page.

Muted
: Mutes or unmutes all audio output for this Browser instance.

## Events

AudioPlaybackStarted
: Occurs when an audio has been started playing on the web page.

AudioPlaybackStopped
: Occurs when an audio has been stopped playing on the web page.

# Property Browser

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with this object.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with this object.

# Property IsPlaying

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## IsPlaying

Indicates if the audio is currently playing on the loaded web page.

```
bool IsPlaying { get; }
```

## Property Value

bool

`true` when audio is currently playing, `false` otherwise.

## Exceptions

ObjectDisposedException

The IAudio has already been disposed.

# Property Muted

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## Muted

Mutes or unmutes all audio output for this Browser instance.

```
bool Muted { get; set; }
```

## Property Value

bool

`true` when audio output must be muted, `false` otherwise.

## Exceptions

ObjectDisposedException

The IAudio has already been disposed.

# Event AudioPlaybackStarted

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

Occurs when an audio has been started playing on the web page.

```
event EventHandler<AudioPlaybackStartedEventArgs> AudioPlaybackStarted
```

## Returns

EventHandler<AudioPlaybackStartedEventArgs>

Occurs when an audio has been started playing on the web page.

# Event AudioPlaybackStopped

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

Occurs when an audio has been stopped playing on the web page.

```
event EventHandler<AudioPlaybackStoppedEventArgs> AudioPlaybackStopped
```

## Returns

EventHandler<AudioPlaybackStoppedEventArgs>

Occurs when an audio has been stopped playing on the web page.

# Interface IMediaDevices

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

An engine service that allows accessing all the available media input devices.

```
public interface IMediaDevices : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

AudioCaptureDevices

Gets a collection of the available audio capture devices.

Engine

Gets the IEngine instance associated with this object.

SelectMediaDeviceHandler

Gets or sets a handler that is used when the web page asks which media input device should be used.

VideoCaptureDevices

Gets a collection of the available video capture devices.

# Property AudioCaptureDevices

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## AudioCaptureDevices

Gets a collection of the available audio capture devices.

```
IEnumerable<MediaDevice> AudioCaptureDevices { get; }
```

## Property Value

IEnumerable<MediaDevice>

The collection of the available audio capture devices.

## Exceptions

ObjectDisposedException

The IMediaDevices has already been disposed.

# Property Engine

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property SelectMediaDeviceHandler

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## SelectMediaDeviceHandler

Gets or sets a handler that is used when the web page asks which media input device should be used.

```
IHandler<SelectMediaDeviceParameters, SelectMediaDeviceResponse> SelectMediaDeviceHandler {
get; set; }
```

## Property Value

IHandler<SelectMediaDeviceParameters, SelectMediaDeviceResponse>

## Remarks

Use the Select(MediaDevice) to select a specific media input device.

If there are no media input devices of the requested type (e.g. there are no video input devices), then the callback **will not be invoked**).

**Important:** the engine will be blocked until you return control from the callback.

If the callback throws an exception or the returned media device is not the one from the Devices, the system default media input device will be used.

## Exceptions

ObjectDisposedException

The IMediaDevices has already been disposed.

# Property VideoCaptureDevices

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## VideoCaptureDevices

Gets a collection of the available video capture devices.

```
IEnumerable<MediaDevice> VideoCaptureDevices { get; }
```

## Property Value

IEnumerable<MediaDevice>

The collection of the available video capture devices.

## Exceptions

ObjectDisposedException

The IMediaDevices has already been disposed.

# Class MediaDevice

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

The details of the media input device (audio/video).

```
public sealed class MediaDevice
```

**Inheritance**

object ← MediaDevice

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Id

Gets the device's unique ID.

Name

Gets the device's "friendly" name.

Type

Gets the device's type.

# Property Id

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## Id

Gets the device's unique ID.

```
public string Id { get; }
```

## Property Value

string

The device's unique ID.

# Property Name

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## Name

Gets the device's "friendly" name.

```
public string Name { get; }
```

## Property Value

string

The device's "friendly" name or an empty string when there's no device name. This device name is not guaranteed to be unique.

# Property Type

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

## Type

Gets the device's type.

```
public MediaDeviceType Type { get; }
```

## Property Value

MediaDeviceType

The device's type.

# Enum MediaDeviceType

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

The media device types.

```
public enum MediaDeviceType
```

## Fields

`AudioDevice = 1`

An audio device.

`Unrecognized = -1`

The media device type is not recognized.

`Unspecified = 0`

The media device type is not specified.

`VideoDevice = 2`

A video device.

# Enum MediaType

Namespace: DotNetBrowser.Media

Assembly: DotNetBrowser.dll

List of the known media types. It's used in ShowContextMenuParameters.

```
public enum MediaType
```

## Fields

`Audio = 4`

An audio node is selected.

`Canvas = 5`

A canvas node is selected.

`File = 6`

A file node is selected.

`Image = 2`

An image node is selected.

`None = 1`

No special node is in context.

`Plugin = 7`

A plugin node is selected.

`Unrecognized = -1`

The media type is not recognized.

`Unspecified = 0`

Unspecified node.

`Video = 3`

A video node is selected.

# Namespace DotNetBrowser.Media.Events

## Classes

AudioEventArgs

The base class for IAudio related events.

AudioPlaybackStartedEventArgs

Event arguments for the AudioPlaybackStarted event.

AudioPlaybackStoppedEventArgs

Event arguments for the AudioPlaybackStopped event.

# Class AudioEventArgs

Namespace: DotNetBrowser.Media.Events

Assembly: DotNetBrowser.dll

The base class for IAudio related events.

```
public class AudioEventArgs : BrowserEventArgs
```

**Inheritance**

object ← EventArgs ← BrowserEventArgs ← AudioEventArgs

**Derived**

AudioPlaybackStartedEventArgs, AudioPlaybackStoppedEventArgs

**Inherited Members**

BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Audio

Gets the IAudio instance that initiated this event.

# Property Audio

Namespace: DotNetBrowser.Media.Events

Assembly: DotNetBrowser.dll

## Audio

Gets the IAudio instance that initiated this event.

```
public IAudio Audio { get; }
```

## Property Value

IAudio

The IAudio instance that initiated this event.

# Class AudioPlaybackStartedEventArgs

Namespace: DotNetBrowser.Media.Events

Assembly: DotNetBrowser.dll

Event arguments for the AudioPlaybackStarted event.

```
public sealed class AudioPlaybackStartedEventArgs : AudioEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← AudioEventArgs ← AudioPlaybackStartedEventArgs

### Inherited Members

AudioEventArgs.Audio , BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class AudioPlaybackStoppedEventArgs

Namespace: DotNetBrowser.Media.Events

Assembly: DotNetBrowser.dll

Event arguments for the AudioPlaybackStopped event.

```
public sealed class AudioPlaybackStoppedEventArgs : AudioEventArgs
```

### Inheritance

object ← EventArgs ← BrowserEventArgs ← AudioEventArgs ← AudioPlaybackStoppedEventArgs

### Inherited Members

AudioEventArgs.Audio , BrowserEventArgs.Browser , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Namespace DotNetBrowser.Media.Handlers

## Classes

SelectMediaDeviceParameters

The parameters of the SelectMediaDeviceHandler.

SelectMediaDeviceResponse

A response to the SelectMediaDeviceHandler.

# Class SelectMediaDeviceParameters

Namespace: DotNetBrowser.Media.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SelectMediaDeviceHandler.

```
public class SelectMediaDeviceParameters
```

**Inheritance**

object ← SelectMediaDeviceParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Devices

Gets the collection of the available media stream devices of the requested Type.

MediaDevices

Gets the IMediaDevices instance associated with the handler.

Type

Get the type of the requested media stream.

# Property Devices

Namespace: DotNetBrowser.Media.Handlers

Assembly: DotNetBrowser.dll

## Devices

Gets the collection of the available media stream devices of the requested Type.

```
public IEnumerable<MediaDevice> Devices { get; }
```

## Property Value

IEnumerable<MediaDevice>

The collection of the available media stream devices of the requested Type. If there are no media stream devices available, this will be an empty collection.

# Property MediaDevices

Namespace: DotNetBrowser.Media.Handlers

Assembly: DotNetBrowser.dll

## MediaDevices

Gets the IMediaDevices instance associated with the handler.

```
public IMediaDevices MediaDevices { get; }
```

## Property Value

IMediaDevices

The IMediaDevices instance associated with the handler.

# Property Type

Namespace: DotNetBrowser.Media.Handlers

Assembly: DotNetBrowser.dll

## Type

Get the type of the requested media stream.

```
public MediaDeviceType Type { get; }
```

## Property Value

MediaDeviceType

The type of the requested media stream.

# Class SelectMediaDeviceResponse

Namespace: DotNetBrowser.Media.Handlers

Assembly: DotNetBrowser.dll

A response to the SelectMediaDeviceHandler.

```
public sealed class SelectMediaDeviceResponse
```

**Inheritance**

object ← SelectMediaDeviceResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Select(MediaDevice)

Creates a SelectMediaDeviceResponse that notifies the engine that the given media input device should be used.

# Method Select

Namespace: DotNetBrowser.Media.Handlers

Assembly: DotNetBrowser.dll

## Select(MediaDevice)

Creates a SelectMediaDeviceResponse that notifies the engine that the given media input device should be used.

```
public static SelectMediaDeviceResponse Select(MediaDevice defaultDevice)
```

## Parameters

`defaultDevice` MediaDevice

The media device to use. Must be the one from the list of the media input devices received from the Devices. If an invalid device is passed through this method, the first device from the list of media devices will be used.

## Returns

SelectMediaDeviceResponse

The SelectMediaDeviceResponse instance that can be used as a return value in SelectMediaDevice Handler.

# Namespace DotNetBrowser.Navigation

## Classes

LoadUrlParameters
: The parameters of the load url request.

NavigationResult
: The result of the navigation occurred.

## Interfaces

INavigation
: Allows loading resources in the browser instance and working with the navigation history.

INavigationEntry
: The navigation history entry.

## Enums

LoadResult
: The loading result.

PageType
: Represents a navigation page type.

# Interface INavigation

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Allows loading resources in the browser instance and working with the navigation history.

```
public interface INavigation : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Browser

Gets the IBrowser instance associated with this object.

CurrentEntry

Gets the current navigation item in the back-forward list.

CurrentIndex

Gets the index of the current navigation item in the back-forward list.

EntryCount

Gets the number of items in the back-forward list.

ShowHttpErrorPageHandler

Gets or sets a handler that is used when the engine is about to display an error web page because the web server sends an empty HTTP response with the HTTP status code that represents an error.

ShowNetErrorPageHandler

Gets or sets a handler that is used when the engine is about to display a network error web page because the required web resource cannot be loaded because of a network error.

StartNavigationHandler

Gets or sets a handler that is used before the browser starts navigation to a resource.

## Methods

CanGoBack()
Checks whether the previous location can be loaded.

CanGoForward()
Checks whether the next location can be loaded.

EntryAt(int)
Returns an INavigationEntry instance for the given `index` in the back-forward list.

GoBack()
Loads the previous location in the back-forward list. It does nothing if there's no previous location in the list.

GoForward()
Loads the next location in the back-forward list. It does nothing if there's no next location in the list.

GoTo(int)
Navigates to a specific location at the given `index` in the back-forward list.

LoadUrl(LoadUrlParameters)
Navigates to a resource identified by the specified `parameters`

LoadUrl(LoadUrlParameters, TimeSpan)
Navigates to a resource identified by the given LoadUrlParameters, with the specified timout.

LoadUrl(string)
Navigates to a resource identified by a URL.

LoadUrl(string, TimeSpan)
Navigates to a resource identified by a URL, with the specified `timeout`.

Reload()
Reloads the currently loaded web page.

ReloadAndCheckForRepost()
Reloads the currently loaded web page. If the current web page has POST data, the user will be asked to confirm that he really wants to reload the page.

ReloadIgnoringCache()
Reloads the currently loaded web page ignoring the cached data.

ReloadIgnoringCacheAndCheckForRepost()

Reloads the currently loaded web page ignoring the cached data. If the current web page has POST data, the user will be asked to confirm that he really wants to reload the page.

RemoveEntryAt(int)

Removes the item at the given `index` from the back-forward list and returns true if it was removed successfully.

Stop()

Cancels any pending navigation or download operation and stops any dynamic page elements, such as background sounds and animations.

## Events

FrameDocumentLoadFinished

Occurs when the document loading in the given IFrame has been finished. At this point, deferred scripts were executed, and content scripts marked "document_end" get injected into the IFrame.

FrameLoadFailed

Occurs when the content load was failed.

FrameLoadFinished

Occurs when the content of the IFrame has been loaded completely.

LoadFinished

Occurs when the content loading has been finished. This event corresponds to the moment when the spinner of the tab stops spinning.

LoadProgressChanged

Occurs when the page has made some progress loading.

LoadStarted

Occurs when the content loading has been started. This event corresponds to the moment when the spinner of the tab starts spinning.

NavigationFinished

Occurs when the navigation has been finished. This happens when a navigation is committed, aborted, or replaced by a new one.

NavigationRedirected

Occurs when the navigation has encountered a server redirect.

NavigationStarted

Occurs when the navigation has been started. This is also fired by same-document navigations, such as fragment navigations or pushState/replaceState, which will not result in a document change. To filter these out, use the IsSameDocument property.

Note that more than one navigation can be ongoing in the same frame at the same time(including the main frame). Also, there is no guarantee that the NavigationFinished will be fired for any particular navigation before this event is called on the next.

NavigationStopped

Occurs when the navigation has been stopped.

# Property Browser

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with this object.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with this object.

# Property CurrentEntry

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## CurrentEntry

Gets the current navigation item in the back-forward list.

```
INavigationEntry CurrentEntry { get; }
```

## Property Value

INavigationEntry

The current navigation item in the back-forward list or null if the list is empty.

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property CurrentIndex

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## CurrentIndex

Gets the index of the current navigation item in the back-forward list.

```
int CurrentIndex { get; }
```

## Property Value

int

The index of the current item in the back-forward list or -1 if the list is empty.

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property EntryCount

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## EntryCount

Gets the number of items in the back-forward list.

```
int EntryCount { get; }
```

## Property Value

int

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property ShowHttpErrorPageHandler

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## ShowHttpErrorPageHandler

Gets or sets a handler that is used when the engine is about to display an error web page because the web server sends an empty HTTP response with the HTTP status code that represents an error.

```
IHandler<ShowHttpErrorPageParameters, ShowHttpErrorPageResponse> ShowHttpErrorPageHandler {
get; set; }
```

## Property Value

IHandler<ShowHttpErrorPageParameters, ShowHttpErrorPageResponse>

## Remarks

Use Show(string) to show the custom error page with the given HTML.

Use ShowDefault() to display the default Chromium error page.

**Important:** the engine will be blocked until the `Handle()` method returns. It is not allowed to invoke any engine methods in the scope of this handler implementation.

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Property ShowNetErrorPageHandler

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## ShowNetErrorPageHandler

Gets or sets a handler that is used when the engine is about to display a network error web page because the required web resource cannot be loaded because of a network error.

```
IHandler<ShowNetErrorPageParameters, ShowNetErrorPageResponse> ShowNetErrorPageHandler {
get; set; }
```

## Property Value

IHandler<ShowNetErrorPageParameters, ShowNetErrorPageResponse>

## Remarks

Use Show(string) to show the custom network error page with the given HTML.

Use ShowDefault() to display the default Chromium network error page.

**Important:** the engine will be blocked until the `Handle()` method returns. It is not allowed to invoke any engine methods in the scope of this handler implementation.

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Property StartNavigationHandler

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## StartNavigationHandler

Gets or sets a handler that is used before the browser starts navigation to a resource.

```
IHandler<StartNavigationParameters, StartNavigationResponse> StartNavigationHandler { get;
set; }
```

## Property Value

IHandler<StartNavigationParameters, StartNavigationResponse>

## Remarks

Use Start() to allow navigation start.

Use Ignore() to ignore navigation request.

**Important:** the browser will be blocked until the `Handle()` method returns. It is not allowed to invoke any browser methods in the scope of this handler implementation.

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method CanGoBack

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## CanGoBack()

Checks whether the previous location can be loaded.

```
bool CanGoBack()
```

## Returns

bool

`true` if the previous location in the back-forward list can be loaded.

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method CanGoForward

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## CanGoForward()

Checks whether the next location can be loaded.

```
bool CanGoForward()
```

## Returns

bool

`true` if the next location in the back-forward list can be loaded.

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method EntryAt

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## EntryAt(int)

Returns an INavigationEntry instance for the given `index` in the back-forward list.

```
INavigationEntry EntryAt(int index)
```

## Parameters

`index` int

The location of the item to return in the back-forward list.

## Returns

INavigationEntry

The navigation entry that corresponds to the index.

## Exceptions

ArgumentOutOfRangeException

The `index` is less than 0 or more than the number of items in the back-forward list.

ObjectDisposedException

The INavigation has already been disposed.

# Method GoBack

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## GoBack()

Loads the previous location in the back-forward list. It does nothing if there's no previous location in the list.

```
Task<NavigationResult> GoBack()
```

### Returns

Task<NavigationResult>

A task that represents the asynchronous navigation operation. The NavigationResult indicates if the navigation operation has been completed, stopped or failed. The task can throw TimeoutException if the navigation operation hasn't been completed within a default timeout (100 seconds).

### Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method GoForward

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## GoForward()

Loads the next location in the back-forward list. It does nothing if there's no next location in the list.

```
Task<NavigationResult> GoForward()
```

## Returns

Task<NavigationResult>

A task that represents the asynchronous navigation operation. The NavigationResult indicates if the navigation operation has been completed, stopped or failed. The task can throw TimeoutException if the navigation operation hasn't been completed within a default timeout (100 seconds).

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method GoTo

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## GoTo(int)

Navigates to a specific location at the given `index` in the back-forward list.

```
Task<NavigationResult> GoTo(int index)
```

## Parameters

`index` int

The location of the item to load in the back-forward list. Cannot be negative or more than the number of items in the back-forward list.

## Returns

Task<NavigationResult>

A task that represents the asynchronous navigation operation. The NavigationResult indicates if the navigation operation has been completed, stopped or failed. The task can throw TimeoutException if the navigation operation hasn't been completed within a default timeout (100 seconds).

## Exceptions

ArgumentOutOfRangeException

The `index` is less than 0 or more than the number of items in the back-forward list.

ObjectDisposedException

The INavigation has already been disposed.

# Method LoadUrl

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## LoadUrl(string)

Navigates to a resource identified by a URL.

```
Task<NavigationResult> LoadUrl(string url)
```

### Parameters

`url` string

The URL of the resource to load. Cannot be null, empty or contain only white space.

### Returns

Task<NavigationResult>

A task that represents the asynchronous loading operation. The NavigationResult indicates if the loading operation has been completed, stopped or failed. The task can throw TimeoutException if the loading operation hasn't been completed within a default timeout (100 seconds).

### Exceptions

ArgumentException

The `url` is null, empty or contain only white space.

ObjectDisposedException

The INavigation has already been disposed.

## LoadUrl(string, TimeSpan)

Navigates to a resource identified by a URL, with the specified `timeout`.

```
Task<NavigationResult> LoadUrl(string url, TimeSpan timeout)
```

## Parameters

`url` string

The URL of the resource to load. Cannot be null, empty or contain only white space.

`timeout` TimeSpan

The timeout for the loading operation.

## Returns

Task<NavigationResult>

A task that represents the asynchronous loading operation. The NavigationResult indicates if the loading operation has been completed, stopped or failed. The task can throw TimeoutException if the loading operation hasn't been completed within a `timeout`.

## Exceptions

ArgumentException

The `url` is null, empty or contain only white space.

ObjectDisposedException

The INavigation has already been disposed.

# LoadUrl(LoadUrlParameters)

Navigates to a resource identified by the specified `parameters`

```
Task<NavigationResult> LoadUrl(LoadUrlParameters parameters)
```

## Parameters

`parameters` LoadUrlParameters

The parameters such as URL, POST data, and HTTP headers.

## Returns

Task<NavigationResult>

A task that represents the asynchronous loading operation. The NavigationResult indicates if the loading operation has been completed, stopped or failed. The task can throw TimeoutException if the loading operation hasn't been completed within a default timeout (100 seconds).

## Exceptions

ArgumentNullException

The `parameters` is null.

ObjectDisposedException

The INavigation has already been disposed.

### LoadUrl(LoadUrlParameters, TimeSpan)

Navigates to a resource identified by the given LoadUrlParameters, with the specified timout.

```
Task<NavigationResult> LoadUrl(LoadUrlParameters parameters, TimeSpan timeout)
```

## Parameters

`parameters` LoadUrlParameters

The parameters such as URL, POST data, and HTTP headers.

`timeout` TimeSpan

The timeout for the loading operation.

## Returns

Task<NavigationResult>

A task that represents the asynchronous loading operation. The NavigationResult indicates if the loading operation has been completed, stopped or failed. The task can throw TimeoutException if the loading operation hasn't been completed within a `timeout`.

## Exceptions

ArgumentNullException

The `parameters` is null.

ObjectDisposedException

The INavigation has already been disposed.

# Method Reload

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## Reload()

Reloads the currently loaded web page.

```
Task<NavigationResult> Reload()
```

### Returns

Task<NavigationResult>

A task that represents the asynchronous reloading operation. The NavigationResult indicates if the reloading operation has been completed, stopped or failed. The task can throw TimeoutException if the reloading operation hasn't been completed within a default timeout (100 seconds).

### Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method ReloadAndCheckForRepost

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## ReloadAndCheckForRepost()

Reloads the currently loaded web page. If the current web page has POST data, the user will be asked to confirm that he really wants to reload the page.

```
Task<NavigationResult> ReloadAndCheckForRepost()
```

### Returns

Task<NavigationResult>

A task that represents the asynchronous reloading operation. The NavigationResult indicates if the reloading operation has been completed, stopped or failed. The task can throw TimeoutException if the reloading operation hasn't been completed within a default timeout (100 seconds).

### Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method ReloadIgnoringCache

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## ReloadIgnoringCache()

Reloads the currently loaded web page ignoring the cached data.

```
Task<NavigationResult> ReloadIgnoringCache()
```

## Returns

Task<NavigationResult>

A task that represents the asynchronous reloading operation. The NavigationResult indicates if the reloading operation has been completed, stopped or failed. The task can throw TimeoutException if the reloading operation hasn't been completed within a default timeout (100 seconds).

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method ReloadIgnoringCacheAndCheckForRepost

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## ReloadIgnoringCacheAndCheckForRepost()

Reloads the currently loaded web page ignoring the cached data. If the current web page has POST data, the user will be asked to confirm that he really wants to reload the page.

```
Task<NavigationResult> ReloadIgnoringCacheAndCheckForRepost()
```

### Returns

Task<NavigationResult>

A task that represents the asynchronous reloading operation. The NavigationResult indicates if the reloading operation has been completed, stopped or failed. The task can throw TimeoutException if the reloading operation hasn't been completed within a default timeout (100 seconds).

### Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Method RemoveEntryAt

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## RemoveEntryAt(int)

Removes the item at the given `index` from the back-forward list and returns true if it was removed successfully.

```
bool RemoveEntryAt(int index)
```

## Parameters

`index` int

The location of the item to remove in the back-forward list. Cannot be negative, point to the current item, or more than the number of items in the back-forward list.

## Returns

bool

returns true if the item was removed successfully.

## Exceptions

ArgumentOutOfRangeException

The `index` is less than 0 or more than the number of items in the back-forward list.

ObjectDisposedException

The INavigation has already been disposed.

# Method Stop

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## Stop()

Cancels any pending navigation or download operation and stops any dynamic page elements, such as background sounds and animations.

```
void Stop()
```

## Exceptions

ObjectDisposedException

The INavigation has already been disposed.

# Event FrameDocumentLoadFinished

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the document loading in the given IFrame has been finished. At this point, deferred scripts were executed, and content scripts marked "document_end" get injected into the IFrame.

```
event EventHandler<FrameDocumentLoadFinishedEventArgs> FrameDocumentLoadFinished
```

## Returns

EventHandler<FrameDocumentLoadFinishedEventArgs>

Occurs when the document loading in the given has been finished. At this point, deferred scripts were executed, and content scripts marked "document_end" get injected into the .

# Event FrameLoadFailed

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the content load was failed.

```
event EventHandler<FrameLoadFailedEventArgs> FrameLoadFailed
```

## Returns

EventHandler<FrameLoadFailedEventArgs>

Occurs when the content load was failed.

# Event FrameLoadFinished

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the content of the IFrame has been loaded completely.

```
event EventHandler<FrameLoadFinishedEventArgs> FrameLoadFinished
```

## Returns

EventHandler<FrameLoadFinishedEventArgs>

Occurs when the content of the has been loaded completely.

# Event LoadFinished

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the content loading has been finished. This event corresponds to the moment when the spinner of the tab stops spinning.

```
event EventHandler<LoadFinishedEventArgs> LoadFinished
```

## Returns

EventHandler<LoadFinishedEventArgs>

Occurs when the content loading has been finished. This event corresponds to the moment when the spinner of the tab stops spinning.

# Event LoadProgressChanged

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the page has made some progress loading.

```
event EventHandler<LoadProgressChangedEventArgs> LoadProgressChanged
```

## Returns

EventHandler<LoadProgressChangedEventArgs>

Occurs when the page has made some progress loading.

# Event LoadStarted

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the content loading has been started. This event corresponds to the moment when the spinner of the tab starts spinning.

```
event EventHandler<LoadStartedEventArgs> LoadStarted
```

## Returns

EventHandler<LoadStartedEventArgs>

Occurs when the content loading has been started. This event corresponds to the moment when the spinner of the tab starts spinning.

# Event NavigationFinished

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the navigation has been finished. This happens when a navigation is committed, aborted, or replaced by a new one.

```
event EventHandler<NavigationFinishedEventArgs> NavigationFinished
```

## Returns

EventHandler<NavigationFinishedEventArgs>

Occurs when the navigation has been finished. This happens when a navigation is committed, aborted, or replaced by a new one.

# Event NavigationRedirected

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the navigation has encountered a server redirect.

```
event EventHandler<NavigationRedirectedEventArgs> NavigationRedirected
```

## Returns

EventHandler<NavigationRedirectedEventArgs>

Occurs when the navigation has encountered a server redirect.

# Event NavigationStarted

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the navigation has been started. This is also fired by same-document navigations, such as fragment navigations or pushState/replaceState, which will not result in a document change. To filter these out, use the IsSameDocument property.

Note that more than one navigation can be ongoing in the same frame at the same time(including the main frame). Also, there is no guarantee that the NavigationFinished will be fired for any particular navigation before this event is called on the next.

```
event EventHandler<NavigationStartedEventArgs> NavigationStarted
```

## Returns

EventHandler<NavigationStartedEventArgs>

Occurs when the navigation has been started. This is also fired by same-document navigations, such as fragment navigations or pushState/replaceState, which will not result in a document change. To filter these out, use the property. Note that more than one navigation can be ongoing in the same frame at the same time(including the main frame). Also, there is no guarantee that the will be fired for any particular navigation before this event is called on the next.

# Event NavigationStopped

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Occurs when the navigation has been stopped.

```
event EventHandler<NavigationStoppedEventArgs> NavigationStopped
```

## Returns

EventHandler<NavigationStoppedEventArgs>

Occurs when the navigation has been stopped.

# Interface INavigationEntry

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

The navigation history entry.

```
public interface INavigationEntry
```

## Properties

HttpStatusCode
: Gets the status code of the last known successful navigation.

OriginalRequestUrl
: Gets the URL that caused this entry to be created.

PageType
: Gets the page type that tells if this entry is for an interstitial or error page.

Timestamp
: Gets the time at which the last known local navigation was completed.

Title
: Gets the title as set by the page. This value is empty if there is no title set.

Url
: Gets the actual URL of the web page.

# Property HttpStatusCode

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## HttpStatusCode

Gets the status code of the last known successful navigation.

```
int HttpStatusCode { get; }
```

## Property Value

int

The status code of the last known successful navigation.

# Property OriginalRequestUrl

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## OriginalRequestUrl

Gets the URL that caused this entry to be created.

```
string OriginalRequestUrl { get; }
```

## Property Value

string

The URL that caused this entry to be created.

# Property PageType

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## PageType

Gets the page type that tells if this entry is for an interstitial or error page.

```
PageType PageType { get; }
```

## Property Value

PageType

The page type that tells if this entry is for an interstitial or error page.

# Property Timestamp

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## Timestamp

Gets the time at which the last known local navigation was completed.

```
DateTime? Timestamp { get; }
```

## Property Value

DateTime?

The time at which the last known local navigation was completed. A navigation can be completed more than once if the page is reloaded.

# Property Title

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## Title

Gets the title as set by the page. This value is empty if there is no title set.

```
string Title { get; }
```

## Property Value

string

The title as set by the page. This value is empty if there is no title set.

# Property Url

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## Url

Gets the actual URL of the web page.

```
string Url { get; }
```

## Property Value

string

The actual URL of the web page.

# Enum LoadResult

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

The loading result.

```
public enum LoadResult
```

## Fields

`Completed = 0`

The page loading has completed.

`Failed = 1`

The page loading has failed.

`Stopped = 2`

The page loading has stopped.

# Class LoadUrlParameters

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

The parameters of the load url request.

```
public sealed class LoadUrlParameters
```

**Inheritance**

object ← LoadUrlParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

LoadUrlParameters(string)

Initializes a new instance of LoadUrlParameters with the specified `url`.

## Properties

HttpHeaders

Gets or sets the HTTP headers that will be sent with the request.

UploadData

Gets or sets the POST upload data that will be sent with the request.

Url

Gets the URL of the resource to load.

# Constructor LoadUrlParameters

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## LoadUrlParameters(string)

Initializes a new instance of LoadUrlParameters with the specified `url`.

```
public LoadUrlParameters(string url)
```

## Parameters

`url` string

The URL of the resource to load.

## Exceptions

ArgumentException

The `url` is null, empty or contain only white space.

# Property HttpHeaders

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## HttpHeaders

Gets or sets the HTTP headers that will be sent with the request.

```
public IEnumerable<HttpHeader> HttpHeaders { get; set; }
```

## Property Value

IEnumerable<HttpHeader>

The HTTP headers that will be sent with the request. Cannot be null or contain null items.

## Exceptions

ArgumentException

The `value` is null or contains null items.

# Property UploadData

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## UploadData

Gets or sets the POST upload data that will be sent with the request.

```
public UploadData UploadData { get; set; }
```

## Property Value

UploadData

The POST upload data that will be sent with the request.

# Property Url

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the resource to load.

```
public string Url { get; }
```

## Property Value

string

The URL of the resource to load.

# Class NavigationResult

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

The result of the navigation occurred.

```
public sealed class NavigationResult
```

**Inheritance**

object ← NavigationResult

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Error

Gets the network error associated with the navigation.

LoadResult

Gets the load result, which can be used to determine whether the navigation has succeeded, failed, or stopped.

# Property Error

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## Error

Gets the network error associated with the navigation.

```
public NetError? Error { get; }
```

## Property Value

NetError?

The network error or `null` if there was no network error during the navigation.

# Property LoadResult

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

## LoadResult

Gets the load result, which can be used to determine whether the navigation has succeeded, failed, or stopped.

```
public LoadResult LoadResult { get; }
```

## Property Value

LoadResult

One of the LoadResult values.

# Enum PageType

Namespace: DotNetBrowser.Navigation

Assembly: DotNetBrowser.dll

Represents a navigation page type.

```
public enum PageType
```

## Fields

`Error = 1`

Error page.

`Interstitial = 2`

Interstitial page.

`Normal = 0`

Normal page.

# Namespace DotNetBrowser.Navigation.Events

## Classes

FrameDocumentLoadFinishedEventArgs
: Event arguments for the FrameDocumentLoadFinished event.

FrameLoadFailedEventArgs
: Event arguments for the FrameLoadFailed event.

FrameLoadFinishedEventArgs
: Event arguments for the FrameLoadFinished event.

FrameNavigationEventArgs
: The base class for all the INavigation event arguments containing IFrame.

LoadFinishedEventArgs
: Event arguments for the LoadFinished event.

LoadProgressChangedEventArgs
: Event arguments for the LoadProgressChanged event.

LoadStartedEventArgs
: Event arguments for the LoadStarted event.

NavigationEventArgs
: The base class for INavigation event arguments.

NavigationFinishedEventArgs
: Event arguments for the NavigationFinished event.

NavigationRedirectedEventArgs
: Event arguments for the NavigationRedirected event.

NavigationStartedEventArgs
: The event arguments for the NavigationStarted event.

NavigationStoppedEventArgs
: Event arguments for the NavigationStopped event.

# Class FrameDocumentLoadFinishedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the FrameDocumentLoadFinished event.

```
public class FrameDocumentLoadFinishedEventArgs : FrameNavigationEventArgs
```

### Inheritance

object ← EventArgs ← NavigationEventArgs ← FrameNavigationEventArgs ← FrameDocumentLoadFinishedEventArgs

### Inherited Members

FrameNavigationEventArgs.Frame , NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class FrameLoadFailedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the FrameLoadFailed event.

```
public class FrameLoadFailedEventArgs : FrameNavigationEventArgs
```

**Inheritance**

object ← EventArgs ← NavigationEventArgs ← FrameNavigationEventArgs ← FrameLoadFailedEventArgs

**Inherited Members**

FrameNavigationEventArgs.Frame , NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ErrorCode

Gets the network error code which is one from the NetError enumeration.

ValidatedUrl

Gets the URL address which is failed to load.

# Property ErrorCode

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## ErrorCode

Gets the network error code which is one from the NetError enumeration.

```
public NetError ErrorCode { get; }
```

## Property Value

NetError

The network error code which is one from the NetError enumeration.

# Property ValidatedUrl

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## ValidatedUrl

Gets the URL address which is failed to load.

```
public string ValidatedUrl { get; }
```

## Property Value

string

The URL address which is failed to load.

# Class FrameLoadFinishedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the FrameLoadFinished event.

```
public class FrameLoadFinishedEventArgs : FrameNavigationEventArgs
```

**Inheritance**

object ← EventArgs ← NavigationEventArgs ← FrameNavigationEventArgs ← FrameLoadFinishedEventArgs

**Inherited Members**

FrameNavigationEventArgs.Frame , NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ValidatedUrl

Gets the URL address which is finished loading.

# Property ValidatedUrl

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## ValidatedUrl

Gets the URL address which is finished loading.

```
public string ValidatedUrl { get; }
```

## Property Value

string

The URL address which is finished loading.

# Class FrameNavigationEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

The base class for all the INavigation event arguments containing IFrame.

```
public class FrameNavigationEventArgs : NavigationEventArgs
```

**Inheritance**

object ← EventArgs ← NavigationEventArgs ← FrameNavigationEventArgs

**Derived**

FrameDocumentLoadFinishedEventArgs, FrameLoadFailedEventArgs, FrameLoadFinishedEventArgs, NavigationFinishedEventArgs

**Inherited Members**

NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Frame

Gets the IFrame instance associated with the event.

# Property Frame

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## Frame

Gets the IFrame instance associated with the event.

```
public IFrame Frame { get; }
```

## Property Value

IFrame

The IFrame instance associated with the event.

# Class LoadFinishedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the LoadFinished event.

```
public class LoadFinishedEventArgs : NavigationEventArgs
```

### Inheritance

object ← EventArgs ← NavigationEventArgs ← LoadFinishedEventArgs

### Inherited Members

NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class LoadProgressChangedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the LoadProgressChanged event.

```
public sealed class LoadProgressChangedEventArgs : NavigationEventArgs
```

**Inheritance**

object ← EventArgs ← NavigationEventArgs ← LoadProgressChangedEventArgs

**Inherited Members**

NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Progress

Gets the value between 0 and 1 indicating the loading progress of the web page.

# Property Progress

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## Progress

Gets the value between 0 and 1 indicating the loading progress of the web page.

```
public double Progress { get; }
```

## Property Value

double

a value between 0 and 1 indicating the loading progress of the web page.

# Class LoadStartedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the LoadStarted event.

```
public class LoadStartedEventArgs : NavigationEventArgs
```

### Inheritance

object ← EventArgs ← NavigationEventArgs ← LoadStartedEventArgs

### Inherited Members

NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class NavigationEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

The base class for INavigation event arguments.

```
public class NavigationEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← NavigationEventArgs

**Derived**

FrameNavigationEventArgs, LoadFinishedEventArgs, LoadProgressChangedEventArgs, LoadStartedEventArgs, NavigationRedirectedEventArgs, NavigationStartedEventArgs, NavigationStoppedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Browser

Gets the IBrowser instance associated with the event.

Navigation

Gets the INavigation instance associated with the event.

# Property Browser

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with the event.

```
public IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with the event.

# Property Navigation

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## Navigation

Gets the INavigation instance associated with the event.

```
public INavigation Navigation { get; }
```

## Property Value

INavigation

The INavigation instance associated with the event.

# Class NavigationFinishedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the NavigationFinished event.

```
public class NavigationFinishedEventArgs : FrameNavigationEventArgs
```

**Inheritance**

object ← EventArgs ← NavigationEventArgs ← FrameNavigationEventArgs ← NavigationFinishedEventArgs

**Inherited Members**

FrameNavigationEventArgs.Frame , NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ErrorCode

Gets the navigation error code.

HasCommitted

Indicates whether the navigation has committed.

IsErrorPage

Indicates whether the navigation resulted in an error page.

IsSameDocument

Indicates whether the navigation has been performed in the scope of the same document.

Url

Gets the navigation URL.

WasServerRedirect

Indicates whether the navigation has encountered a server redirect.

# Property ErrorCode

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## ErrorCode

Gets the navigation error code.

```
public NetError ErrorCode { get; }
```

## Property Value

NetError

The navigation error code. In case the navigation has failed, contains the error code of the network error.

# Property HasCommitted

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## HasCommitted

Indicates whether the navigation has committed.

```
public bool HasCommitted { get; }
```

## Property Value

bool

`true` if the navigation has committed, `false` otherwise.

# Property IsErrorPage

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## IsErrorPage

Indicates whether the navigation resulted in an error page.

```
public bool IsErrorPage { get; }
```

## Property Value

bool

`true` if the navigation resulted in an error page, `false` otherwise.

# Property IsSameDocument

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## IsSameDocument

Indicates whether the navigation has been performed in the scope of the same document.

```
public bool IsSameDocument { get; }
```

## Property Value

bool

`true` if the navigation has been performed in the scope of the same document, `false` otherwise.

# Property Url

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## Url

Gets the navigation URL.

```
public string Url { get; }
```

## Property Value

string

The navigation URL.

# Property WasServerRedirect

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## WasServerRedirect

Indicates whether the navigation has encountered a server redirect.

```
public bool WasServerRedirect { get; }
```

## Property Value

bool

`true` if the navigation has encountered a server redirect, `false` otherwise.

# Class NavigationRedirectedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the NavigationRedirected event.

```
public class NavigationRedirectedEventArgs : NavigationEventArgs
```

**Inheritance**

object ← EventArgs ← NavigationEventArgs ← NavigationRedirectedEventArgs

**Inherited Members**

NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Url

Gets the destination URL for the navigation redirect.

# Property Url

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## Url

Gets the destination URL for the navigation redirect.

```
public string Url { get; }
```

## Property Value

string

The destination URL for the navigation redirect.

# Class NavigationStartedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

The event arguments for the NavigationStarted event.

```
public class NavigationStartedEventArgs : NavigationEventArgs
```

**Inheritance**

object ← EventArgs ← NavigationEventArgs ← NavigationStartedEventArgs

**Inherited Members**

NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

IsMainFrame

Indicates whether the navigation is started in the main frame.

IsSameDocument

Indicates whether the navigation will be performed in the scope of the same document.

Url

Gets the URL of the navigation.

# Property IsMainFrame

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## IsMainFrame

Indicates whether the navigation is started in the main frame.

```
public bool IsMainFrame { get; }
```

## Property Value

bool

`true` if the navigation is started in the main frame, `false` otherwise.

# Property IsSameDocument

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## IsSameDocument

Indicates whether the navigation will be performed in the scope of the same document.

```
public bool IsSameDocument { get; }
```

## Property Value

bool

`true` if the navigation will be performed in the scope of the same document, `false` otherwise.

# Property Url

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the navigation.

```
public string Url { get; }
```

## Property Value

string

The navigation URL.

# Class NavigationStoppedEventArgs

Namespace: DotNetBrowser.Navigation.Events

Assembly: DotNetBrowser.dll

Event arguments for the NavigationStopped event.

```
public class NavigationStoppedEventArgs : NavigationEventArgs
```

### Inheritance

object ← EventArgs ← NavigationEventArgs ← NavigationStoppedEventArgs

### Inherited Members

NavigationEventArgs.Browser , NavigationEventArgs.Navigation , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Namespace DotNetBrowser.Navigation.Handlers

## Classes

NavigationParameters

The base class for all INavigation handlers parameters.

ShowHttpErrorPageParameters

The parameters of the ShowHttpErrorPageHandler.

ShowHttpErrorPageResponse

A response to the ShowHttpErrorPageHandler.

ShowNetErrorPageParameters

The parameters of the ShowNetErrorPageHandler.

ShowNetErrorPageResponse

A response to the ShowNetErrorPageHandler.

StartNavigationParameters

The parameters of the StartNavigationHandler.

StartNavigationResponse

A response to the StartNavigationHandler.

# Class NavigationParameters

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

The base class for all INavigation handlers parameters.

```
public class NavigationParameters
```

**Inheritance**

object ← NavigationParameters

**Derived**

ShowHttpErrorPageParameters, ShowNetErrorPageParameters, StartNavigationParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Navigation

Gets the INavigation instance the handler is associated with.

# Property Navigation

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## Navigation

Gets the INavigation instance the handler is associated with.

```
public INavigation Navigation { get; }
```

## Property Value

INavigation

The INavigation instance the handler is associated with.

# Class ShowHttpErrorPageParameters

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

The parameters of the ShowHttpErrorPageHandler.

```
public sealed class ShowHttpErrorPageParameters : NavigationParameters
```

**Inheritance**

object ← NavigationParameters ← ShowHttpErrorPageParameters

**Inherited Members**

NavigationParameters.Navigation , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

HttpStatus

Gets the HTTP status code of the request.

Url

Gets the URL of the unreachable navigation request.

# Property HttpStatus

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## HttpStatus

Gets the HTTP status code of the request.

```
public HttpStatusCode HttpStatus { get; }
```

## Property Value

HttpStatusCode

The HTTP status code of the request.

# Property Url

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the unreachable navigation request.

```
public string Url { get; }
```

## Property Value

string

The URL of the unreachable navigation request.

# Class ShowHttpErrorPageResponse

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

A response to the ShowHttpErrorPageHandler.

```
public sealed class ShowHttpErrorPageResponse
```

**Inheritance**

object ← ShowHttpErrorPageResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Show(string)

Creates a ShowHttpErrorPageResponse that notifies the browser that it should display an error page with the specified `errorHtml`.

ShowDefault()

Creates a ShowHttpErrorPageResponse that notifies the browser that it should display the default error page.

# Method Show

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## Show(string)

Creates a ShowHttpErrorPageResponse that notifies the browser that it should display an error page with the specified `errorHtml`.

```
public static ShowHttpErrorPageResponse Show(string errorHtml)
```

## Parameters

`errorHtml` string

The HTML code of the error page to display.

## Returns

ShowHttpErrorPageResponse

The ShowHttpErrorPageResponse instance that can be used as a return value in ShowHttpErrorPageHandler implementation.

## Exceptions

ArgumentException

The `errorHtml` is null, empty, or contains only white space.

# Method ShowDefault

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## ShowDefault()

Creates a ShowHttpErrorPageResponse that notifies the browser that it should display the default error page.

```
public static ShowHttpErrorPageResponse ShowDefault()
```

## Returns

ShowHttpErrorPageResponse

The ShowHttpErrorPageResponse instance that can be used as a return value in ShowHttpErrorPageHandler implementation.

# Class ShowNetErrorPageParameters

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

The parameters of the ShowNetErrorPageHandler.

```
public sealed class ShowNetErrorPageParameters : NavigationParameters
```

**Inheritance**

object ← NavigationParameters ← ShowNetErrorPageParameters

**Inherited Members**

NavigationParameters.Navigation , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

NetError

Gets the URL request error code.

Url

Gets the URL of the unreachable navigation request.

# Property NetError

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## NetError

Gets the URL request error code.

```
public NetError NetError { get; }
```

## Property Value

NetError

The URL request error code. In case the request has failed, contains the error code of the network. Otherwise, has the default Undefined value.

# Property Url

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the unreachable navigation request.

```
public string Url { get; }
```

## Property Value

string

The URL of the unreachable navigation request.

# Class ShowNetErrorPageResponse

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

A response to the ShowNetErrorPageHandler.

```
public sealed class ShowNetErrorPageResponse
```

**Inheritance**

object ← ShowNetErrorPageResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Show(string)

Creates a ShowNetErrorPageResponse that notifies the browser that it should display an error page with the specified `errorHtml`.

ShowDefault()

Creates a ShowNetErrorPageResponse that notifies the browser that it should display the default error page.

# Method Show

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## Show(string)

Creates a ShowNetErrorPageResponse that notifies the browser that it should display an error page with the specified `errorHtml`.

```
public static ShowNetErrorPageResponse Show(string errorHtml)
```

## Parameters

`errorHtml` string

The HTML code of the error page to display.

## Returns

ShowNetErrorPageResponse

The ShowNetErrorPageResponse instance that can be used as a return value in ShowNetErrorPageHandler implementation.

## Exceptions

ArgumentException

The `errorHtml` is null, empty, or contains only white space.

# Method ShowDefault

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## ShowDefault()

Creates a ShowNetErrorPageResponse that notifies the browser that it should display the default error page.

```
public static ShowNetErrorPageResponse ShowDefault()
```

## Returns

ShowNetErrorPageResponse

The ShowNetErrorPageResponse instance that can be used as a return value in ShowNetErrorPageHandler implementation.

# Class StartNavigationParameters

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

The parameters of the StartNavigationHandler.

```
public sealed class StartNavigationParameters : NavigationParameters
```

**Inheritance**

object ← NavigationParameters ← StartNavigationParameters

**Inherited Members**

NavigationParameters.Navigation , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

HasUserGesture

Indicates whether the navigation was initiated by a user gesture.

IsExternalProtocol

Indicates whether the target URL cannot be handled by the browser's internal protocol handlers.

IsMainFrame

Indicates whether the navigation request URL belongs to a main frame.

IsPost

Indicates whether the navigation is done using HTTP POST method.

IsRedirect

Indicates whether it's a redirect.

Url

Gets the URL of the resource that will be loaded.

# Property HasUserGesture

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## HasUserGesture

Indicates whether the navigation was initiated by a user gesture.

```
public bool HasUserGesture { get; }
```

## Property Value

bool

`true` if the navigation was initiated by a user gesture, `false` otherwise.

# Property IsExternalProtocol

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## IsExternalProtocol

Indicates whether the target URL cannot be handled by the browser's internal protocol handlers.

```
public bool IsExternalProtocol { get; }
```

## Property Value

bool

`true` if the target URL cannot be handled by the browser's internal protocol handlers, `false` otherwise.

# Property IsMainFrame

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## IsMainFrame

Indicates whether the navigation request URL belongs to a main frame.

```
public bool IsMainFrame { get; }
```

## Property Value

bool

`true` if the navigation request URL belongs to a main frame, `false` otherwise.

# Property IsPost

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## IsPost

Indicates whether the navigation is done using HTTP POST method.

```
public bool IsPost { get; }
```

## Property Value

bool

`true` if the navigation is done using HTTP POST method, `false` otherwise.

# Property IsRedirect

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## IsRedirect

Indicates whether it's a redirect.

```
public bool IsRedirect { get; }
```

## Property Value

bool

`true` if the navigation is a redirect, `false` otherwise.

# Property Url

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the resource that will be loaded.

```
public string Url { get; }
```

## Property Value

string

The URL of the resource that will be loaded.

# Class StartNavigationResponse

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

A response to the StartNavigationHandler.

```
public sealed class StartNavigationResponse
```

**Inheritance**

object ← StartNavigationResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Ignore()

Creates a StartNavigationResponse that notifies the browser that the load request should be ignored.

Start()

Creates a StartNavigationResponse that notifies the browser that it should continue loading the URL.

# Method Ignore

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## Ignore()

Creates a StartNavigationResponse that notifies the browser that the load request should be ignored.

```
public static StartNavigationResponse Ignore()
```

## Returns

StartNavigationResponse

The StartNavigationResponse instance that can be used as a return value in StartNavigationHandler implementation.

# Method Start

Namespace: DotNetBrowser.Navigation.Handlers

Assembly: DotNetBrowser.dll

## Start()

Creates a StartNavigationResponse that notifies the browser that it should continue loading the URL.

```
public static StartNavigationResponse Start()
```

## Returns

StartNavigationResponse

The StartNavigationResponse instance that can be used as a return value in StartNavigationHandler implementation.

# Namespace DotNetBrowser.Net

## Classes

BytesData
: The upload data as bytes. Can be empty if the form doesn't contain any data.

FileValue

FormData
: The list of key-value pairs each representing a segment of a form data. Can be empty if the form doesn't contain any data.

HostPortPair
: A host/port pair of the URI.

HttpHeader

MimeType
: The MIME type.

MultipartFormData
: The list of key-value pairs each representing a segment of a multi-part form data. Can be empty if the form doesn't contain any data.

MultipartFormDataKeyValuePair
: A key-value pair that represents a segment of a multi-part form data. Can contain values corresponding a form field content, an upload file content, etc.

Scheme
: The scheme component of a URL.

TextData
: The upload data of the `text/plain` content type.

UploadData
: The base class for all upload data.

UrlRequest
: Represents the URL request received from the Chromium engine.

UrlRequestJob
: The URL request job for the intercepted URL request, which allows you to provide the response data for this URL request.

## Interfaces

IFileValue
: File data.

IHttpAuthPreferences
: The HTTP authorization preferences.

IHttpHeader
: Represents the single HTTP header with all its values.

INetwork
: Allows access and modifying to the network-level activities.

IUploadData
: The upload data associated with a `UrlRequest`. The upload data can be in the `text/plain`, `application/x-www-form-urlencoded`, or `multipart/form-data` format.The upload data type depends on the content of the `Content-Type` header. If the `Content-Type` header is missing or doesn't include a valid substring indicating the corresponding upload data type, the upload data will be in a binary format.

IUploadData<T>
: The upload data associated with a `UrlRequest`. The upload data can be in the `text/plain`, `application/x-www-form-urlencoded`, or `multipart/form-data` format.The upload data type depends on the content of the `Content-Type` header. If the `Content-Type` header is missing or doesn't include a valid substring indicating the corresponding upload data type, the upload data will be in a binary format.

## Enums

NetError
: The network errors.

RequestStatus
: The status of a URL request.

SslVersion
: The supported SSL connection versions.

# Class BytesData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The upload data as bytes. Can be empty if the form doesn't contain any data.

```
public sealed class BytesData : UploadData, IUploadData<IReadOnlyList<byte>>, IUploadData
```

**Inheritance**

object ← UploadData ← BytesData

**Implements**

IUploadData<IReadOnlyList<byte>>, IUploadData

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

BytesData(byte[])

Creates a new instance containing the specified data.

## Properties

Data

The upload data as a read-only list.

## Operators

implicit operator BytesData(byte[])

Converts the specified data to a new instance of BytesData.

# Constructor BytesData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## BytesData(byte[])

Creates a new instance containing the specified data.

```
public BytesData(byte[] data)
```

## Parameters

`data` byte[]

The upload data as a byte array.

# Property Data

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Data

The upload data as a read-only list.

```
public IReadOnlyList<byte> Data { get; }
```

## Property Value

IReadOnlyList<byte>

# Operator implicit operator

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## implicit operator BytesData(byte[])

Converts the specified data to a new instance of BytesData.

```
public static implicit operator BytesData(byte[] data)
```

## Parameters

`data` byte[]

The upload data as a byte array.

## Returns

BytesData

The new BytesData instance.

# Class FileValue

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

```
public sealed class FileValue : IFileValue
```

**Inheritance**

object ← FileValue

**Implements**

IFileValue

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

FileValue(string, MimeType, byte[])

Constructor for FileValue.

FileValue(string, MimeType, string)

Constructor for FileValue.

## Properties

Bytes

The file value as an array of bytes.

ContentType

The content type determined by the file extension. Equals `application/octet-stream` when there is no MIME type associated with the file extension.

FileName

The file name.

FilePath

The file value as a file path.

# Constructor FileValue

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## FileValue(string, MimeType, string)

Constructor for FileValue.

```
public FileValue(string fileName, MimeType contentType, string filePath)
```

### Parameters

`fileName` string

The file name. Cannot be empty.

`contentType` MimeType

A string representing the content type determined by the file extension. Equals `application/octet-stream` when there's no MIME type associated with the file extension.

`filePath` string

The file path. Cannot be empty.

### Exceptions

ArgumentException

when file name or path is null or empty.

## FileValue(string, MimeType, byte[])

Constructor for FileValue.

```
public FileValue(string fileName, MimeType contentType, byte[] bytes)
```

## Parameters

`fileName` string

The file name. Cannot be empty.

`contentType` MimeType

A string representing the content type determined by the file extension. Equals `application/octet-stream` when there's no MIME type associated with the file extension.

`bytes` byte[]

The byte sequence representing the file upload data segment.

## Exceptions

ArgumentException

when file name is null or empty. when bytes array is null.

# Property Bytes

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Bytes

The file value as an array of bytes.

```
public byte[] Bytes { get; }
```

## Property Value

byte[]

# Property ContentType

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## ContentType

The content type determined by the file extension. Equals `application/octet-stream` when there is no MIME type associated with the file extension.

```
public MimeType ContentType { get; }
```

## Property Value

MimeType

# Property FileName

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## FileName

The file name.

```
public string FileName { get; }
```

## Property Value

string

# Property FilePath

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## FilePath

The file value as a file path.

```
public string FilePath { get; }
```

## Property Value

string

# Class FormData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The list of key-value pairs each representing a segment of a form data. Can be empty if the form doesn't contain any data.

```
public sealed class FormData : UploadData,
IUploadData<IReadOnlyCollection<KeyValuePair<string, string>>>, IUploadData
```

**Inheritance**

object ← UploadData ← FormData

**Implements**

IUploadData<IReadOnlyCollection<KeyValuePair<string, string>>>, IUploadData

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

FormData(IReadOnlyCollection<KeyValuePair<string, string>>)

Creates a new instance containing the specified key-value pairs.

## Properties

Data

The key-value pairs that represent segments of the form data.

# Constructor FormData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## FormData(IReadOnlyCollection<KeyValuePair<string, string>>)

Creates a new instance containing the specified key-value pairs.

```
public FormData(IReadOnlyCollection<KeyValuePair<string, string>> data)
```

## Parameters

`data` IReadOnlyCollection<KeyValuePair<string, string>>

The key-value pairs that represent segments of the form data.

# Property Data

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Data

The key-value pairs that represent segments of the form data.

```
public IReadOnlyCollection<KeyValuePair<string, string>> Data { get; }
```

## Property Value

IReadOnlyCollection<KeyValuePair<string, string>>

# Class HostPortPair

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

A host/port pair of the URI.

```
public sealed class HostPortPair
```

**Inheritance**

object ← HostPortPair

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object)

## Properties

Host
: Gets the host of the URI.

Port
: Gets the port of the URI.

## Methods

HasPort()
: Checks whether this host/port pair has a port.

Parse(string)
: Converts a string representation of a host/port pair to the HostPortPair instance.

ToString()

# Property Host

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Host

Gets the host of the URI.

```
public string Host { get; }
```

## Property Value

string

The host of the URI.

# Property Port

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Port

Gets the port of the URI.

```
public int Port { get; }
```

## Property Value

int

The port of the URI.

# Method HasPort

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## HasPort()

Checks whether this host/port pair has a port.

```
public bool HasPort()
```

## Returns

bool

true if the port of this host/port pair is greater than null.

# Method Parse

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Parse(string)

Converts a string representation of a host/port pair to the HostPortPair instance.

```
public static HostPortPair Parse(string hostPort)
```

## Parameters

`hostPort` string

The string representation of the host/port pair separated with the colon.

## Returns

HostPortPair

the HostPortPair instance.

# Method ToString

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class HttpHeader

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

```
public sealed class HttpHeader : IHttpHeader
```

**Inheritance**

object ← HttpHeader

**Implements**

IHttpHeader

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

HttpHeader(string, params string[])

Creates the HTTP header instance using its name and values collection.

## Properties

Name

Gets the HTTP header name.

Values

Gets the collection of the HTTP header values.

# Constructor HttpHeader

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## HttpHeader(string, params string[])

Creates the HTTP header instance using its name and values collection.

```
public HttpHeader(string name, params string[] values)
```

## Parameters

`name` string

The HTTP header name

`values` string[]

The collection of the HTTP header values

# Property Name

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Name

Gets the HTTP header name.

```
public string Name { get; }
```

## Property Value

string

The HTTP header name.

# Property Values

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Values

Gets the collection of the HTTP header values.

```
public IEnumerable<string> Values { get; }
```

## Property Value

IEnumerable<string>

The collection of the HTTP header values.

# Interface IFileValue

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

File data.

```
public interface IFileValue
```

## Remarks

Depending on how the request was constructed can contain the path to a file, or the bytes representing a file.

For example, it contains the path when the value is taken from an `<input type="file">` element of the submitted form.

May contain bytes in case the request in the `multipart/form-data` format was constructed manually in a JavaScript code.

## Properties

Bytes
: The file value as an array of bytes.

ContentType
: The content type determined by the file extension. Equals `application/octet-stream` when there is no MIME type associated with the file extension.

FileName
: The file name.

FilePath
: The file value as a file path.

# Property Bytes

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Bytes

The file value as an array of bytes.

```
byte[] Bytes { get; }
```

## Property Value

byte[]

# Property ContentType

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## ContentType

The content type determined by the file extension. Equals `application/octet-stream` when there is no MIME type associated with the file extension.

```
MimeType ContentType { get; }
```

## Property Value

MimeType

# Property FileName

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## FileName

The file name.

```
string FileName { get; }
```

## Property Value

string

# Property FilePath

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## FilePath

The file value as a file path.

```
string FilePath { get; }
```

## Property Value

string

# Interface IHttpAuthPreferences

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The HTTP authorization preferences.

```
public interface IHttpAuthPreferences
```

## Properties

DelegateWhiteList

Gets or sets the HTTP network delegate authorization white list of URLs. By default, network delegate white list is empty.

ServerWhiteList

Gets or sets the server HTTP authorization white list of URLs. By default, server white list is empty.

# Property DelegateWhiteList

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## DelegateWhiteList

Gets or sets the HTTP network delegate authorization white list of URLs. By default, network delegate white list is empty.

```
string DelegateWhiteList { get; set; }
```

## Property Value

string

The comma/semicolon separated HTTP network delegate authorization white list of URLs.

## Exceptions

ObjectDisposedException

The IHttpAuthPreferences object has already been disposed.

# Property ServerWhiteList

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## ServerWhiteList

Gets or sets the server HTTP authorization white list of URLs. By default, server white list is empty.

```
string ServerWhiteList { get; set; }
```

## Property Value

string

The comma/semicolon separated server HTTP authorization white list of URLs.

## Exceptions

ObjectDisposedException

The IHttpAuthPreferences object has already been disposed.

# Interface IHttpHeader

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Represents the single HTTP header with all its values.

```
public interface IHttpHeader
```

## Properties

Name

Gets the HTTP header name.

Values

Gets the collection of the HTTP header values.

# Property Name

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Name

Gets the HTTP header name.

```
string Name { get; }
```

## Property Value

string

The HTTP header name.

# Property Values

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Values

Gets the collection of the HTTP header values.

```
IEnumerable<string> Values { get; }
```

## Property Value

IEnumerable<string>

The collection of the HTTP header values.

# Interface INetwork

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Allows access and modifying to the network-level activities.

```
public interface INetwork : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

AcceptLanguage
Gets or sets the `Accept-Language` request header value.

AuthenticateHandler
Gets or sets a handler that is used when the website requests authentication.

CanAccessFileHandler
Gets or sets a handler that is used when the Chromium engine is about to access the requested file. Can be used to disallow accessing the file.

CanGetCookiesHandler
Gets or sets a handler that is used when Chromium engine decides whether cookies can be sent back to the web server.

CanSetCookieHandler
Gets or sets a handler that is used when Chromium engine decides whether cookie can be saved for the URL or not.

Engine
Gets the IEngine instance associated with this object.

HttpAuthPreferences
Gets the HTTP authorization preferences.

Profile
Gets the IProfile instance associated with this object.

ReceiveHeadersHandler
: Gets or sets a handler that is used each time that an HTTP(S) response header is received. Due to redirects and authentication requests this can happen multiple times per request. This event is intended to allow adding, modifying, and deleting HTTP response headers, such as incoming "Set-Cookie" headers.

SendUploadDataHandler
: Gets or sets a handler that is used when the upload data is about to send to the web server. Can be used to override the upload data before it is sent.

SendUrlRequestHandler
: Gets or sets a handler that is used when an HTTP request is about to occur. It can be used to override the requested URL and redirect the request to another location.

StartTransactionHandler
: Gets or sets a handler that is used when the request is about to start the transaction process. It allows adding, modifying, and deleting HTTP request headers.

UserAgent
: Gets the default user-agent string.

VerifyCertificateHandler
: Gets or sets a handler that is used to verify the SSL certificate provided by the web server.

## Methods

CreateUrlRequestJob(UrlRequest, UrlRequestJobOptions)
: Creates a new UrlRequestJob instance with the given `options`.

  The URL request job is used to provide the response data for the intercepted URL request.

## Events

ConnectionTypeChanged
: Occurs when the network connection type has been changed.

PacScriptErrorOccurred
: Occurs when the error has been occured in the PAC script.

RedirectResponseCodeReceived

Occurs when the redirect has been occured.

RequestCompleted

Occurs when the request has been completed.

RequestDestroyed

Occurs when the request has been destroyed.

ResponseBytesReceived

Occurs when a part of HTTP response body has been received over the network.

ResponseStarted

Occurs when the response has been started.

# Property AcceptLanguage

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## AcceptLanguage

Gets or sets the `Accept-Language` request header value.

```
string AcceptLanguage { get; set; }
```

## Property Value

string

The Accept-Language value.

## Examples

`"fr, en-gb;q=0.8, en;q=0.7"` would mean: "I prefer French, but will accept British English and other types of English." Note, that all languages which are assigned a quality factor greater than 0 are acceptable.

## Remarks

This field restricts the set of natural languages that are preferred as a response to the request. The default `Accept-Language` is `"en-us"`. See https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html W3 Documentation.

## Exceptions

ArgumentException

The value is set to null or an empty string.

ObjectDisposedException

The INetwork object has already been disposed.

# Property AuthenticateHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## AuthenticateHandler

Gets or sets a handler that is used when the website requests authentication.

```
IHandler<AuthenticateParameters, AuthenticateResponse> AuthenticateHandler { get; set; }
```

## Property Value

IHandler<AuthenticateParameters, AuthenticateResponse>

## Remarks

Use the Continue(string, string) method to continue authentication process.

Use the Cancel() method to cancel authentication process.

If exception occurs inside the handler, the default behavior will be applied - the Cancel() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property CanAccessFileHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## CanAccessFileHandler

Gets or sets a handler that is used when the Chromium engine is about to access the requested file. Can be used to disallow accessing the file.

```
IHandler<CanAccessFileParameters, CanAccessFileResponse> CanAccessFileHandler { get; set; }
```

## Property Value

IHandler<CanAccessFileParameters, CanAccessFileResponse>

## Remarks

Use the Can() method to allow access to the file.

Use the Cannot() method to disallow access to the file.

If exception occurs inside the handler, the default behavior will be applied - the Can() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property CanGetCookiesHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## CanGetCookiesHandler

Gets or sets a handler that is used when Chromium engine decides whether cookies can be sent back to the web server.

```
IHandler<CanGetCookiesParameters, CanGetCookiesResponse> CanGetCookiesHandler { get; set; }
```

## Property Value

IHandler<CanGetCookiesParameters, CanGetCookiesResponse>

## Remarks

Use the Allow() method to allow cookies to be sent to the web server.

Use the Deny() method to disallow cookies to be sent to the web server.

If exception occurs inside the handler, then default behavior will be applied - the Allow() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property CanSetCookieHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## CanSetCookieHandler

Gets or sets a handler that is used when Chromium engine decides whether cookie can be saved for the URL or not.

```
IHandler<CanSetCookieParameters, CanSetCookieResponse> CanSetCookieHandler { get; set; }
```

## Property Value

IHandler<CanSetCookieParameters, CanSetCookieResponse>

## Remarks

Use the Allow() method to allow engine to save the cookie.

Use the Deny() method to deny engine to save the cookie. It will not be available via the ICookieStore object.

If exception occurs inside the handler, then default behavior will be applied - the Allow() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property Engine

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property HttpAuthPreferences

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## HttpAuthPreferences

Gets the HTTP authorization preferences.

```
IHttpAuthPreferences HttpAuthPreferences { get; }
```

## Property Value

IHttpAuthPreferences

The IHttpAuthPreferences instance associated with this object.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property Profile

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Property ReceiveHeadersHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## ReceiveHeadersHandler

Gets or sets a handler that is used each time that an HTTP(S) response header is received. Due to redirects and authentication requests this can happen multiple times per request. This event is intended to allow adding, modifying, and deleting HTTP response headers, such as incoming "Set-Cookie" headers.

```
IHandler<ReceiveHeadersParameters, ReceiveHeadersResponse> ReceiveHeadersHandler { get;
set; }
```

## Property Value

IHandler<ReceiveHeadersParameters, ReceiveHeadersResponse>

## Remarks

Use the OverrideHeaders(IEnumerable<HttpHeader>) method to override headers.

Use the Continue() method to continue without modifications.

If exception occurs inside the handler, the default behavior will be applied - the Continue() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property SendUploadDataHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## SendUploadDataHandler

Gets or sets a handler that is used when the upload data is about to send to the web server. Can be used to override the upload data before it is sent.

```
IHandler<SendUploadDataParameters, SendUploadDataResponse> SendUploadDataHandler { get;
set; }
```

## Property Value

IHandler<SendUploadDataParameters, SendUploadDataResponse>

## Remarks

Use one of the following methods to modify the upload data:

- Override(TextData)
- Override(FormData)
- Override(MultipartFormData)
- Override(BytesData)

Use the Continue() method if you do not need to modify the upload data.

If exception occurs inside the handler, the default behavior will be applied - the Continue() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property SendUrlRequestHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## SendUrlRequestHandler

Gets or sets a handler that is used when an HTTP request is about to occur. It can be used to override the requested URL and redirect the request to another location.

```
IHandler<SendUrlRequestParameters, SendUrlRequestResponse> SendUrlRequestHandler { get;
set; }
```

## Property Value

IHandler<SendUrlRequestParameters, SendUrlRequestResponse>

## Remarks

Use the Override(string) method to override the requested URL.

Use the Continue() method to proceed without changes.

If exception occurs inside the handler, the default behavior will be applied - the Continue() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property StartTransactionHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## StartTransactionHandler

Gets or sets a handler that is used when the request is about to start the transaction process. It allows adding, modifying, and deleting HTTP request headers.

```
IHandler<StartTransactionParameters, StartTransactionResponse> StartTransactionHandler {
get; set; }
```

## Property Value

IHandler<StartTransactionParameters, StartTransactionResponse>

## Remarks

Use the OverrideHeaders(IEnumerable<HttpHeader>) method to override headers.

Use the Continue() method to continue with original headers.

If exception occurs inside the handler, the default behavior will be applied - the Continue() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property UserAgent

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## UserAgent

Gets the default user-agent string.

```
string UserAgent { get; }
```

## Property Value

string

The default user-agent string.

## Remarks

The default user-agent string can be configured through the UserAgent property. If the default user-agent string has not been specified, then this method returns a string obtained from the Chromium engine.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Property VerifyCertificateHandler

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## VerifyCertificateHandler

Gets or sets a handler that is used to verify the SSL certificate provided by the web server.

```
IHandler<VerifyCertificateParameters, VerifyCertificateResponse> VerifyCertificateHandler {
get; set; }
```

## Property Value

IHandler<VerifyCertificateParameters, VerifyCertificateResponse>

## Remarks

Use the Valid() method if the SSL certificate should be accepted.

Use the Invalid() method if the SSL certificate must be rejected.

Use the Default() method to let Chromium decide whether SSL certificate should be accepted or rejected.

If this method throws an exception, then default behavior will be applied - the Default() method will be used.

## Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Method CreateUrlRequestJob

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## CreateUrlRequestJob(UrlRequest, UrlRequestJobOptions)

Creates a new UrlRequestJob instance with the given `options`.

The URL request job is used to provide the response data for the intercepted URL request.

```
UrlRequestJob CreateUrlRequestJob(UrlRequest request, UrlRequestJobOptions options = null)
```

### Parameters

`request` UrlRequest

The intercepted UrlRequest instance to bind the job with.

`options` UrlRequestJobOptions

The options which are used to initialize the UrlRequestJob

### Returns

UrlRequestJob

A new UrlRequestJob instance.

### Exceptions

ObjectDisposedException

The INetwork object has already been disposed.

# Event ConnectionTypeChanged

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Occurs when the network connection type has been changed.

```
event EventHandler<ConnectionTypeChangedEventArgs> ConnectionTypeChanged
```

## Returns

EventHandler<ConnectionTypeChangedEventArgs>

Occurs when the network connection type has been changed.

## Remarks

The event is raised whenever a change affects the route network packets take to any network server.

For example, the event is triggered when:

- A network connection becomes available or going away;
- A VPN tunnel is established or taken down;
- An active network connection's IP address changes.

# Event PacScriptErrorOccurred

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Occurs when the error has been occured in the PAC script.

```
event EventHandler<PacScriptErrorEventArgs> PacScriptErrorOccurred
```

## Returns

EventHandler<PacScriptErrorEventArgs>

Occurs when the error has been occured in the PAC script.

# Event RedirectResponseCodeReceived

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Occurs when the redirect has been occured.

```
event EventHandler<RedirectResponseCodeReceivedEventArgs> RedirectResponseCodeReceived
```

## Returns

EventHandler<RedirectResponseCodeReceivedEventArgs>

Occurs when the redirect has been occured.

# Event RequestCompleted

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Occurs when the request has been completed.

```
event EventHandler<RequestCompletedEventArgs> RequestCompleted
```

## Returns

EventHandler<RequestCompletedEventArgs>

Occurs when the request has been completed.

# Event RequestDestroyed

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Occurs when the request has been destroyed.

```
event EventHandler<RequestDestroyedEventArgs> RequestDestroyed
```

## Returns

EventHandler<RequestDestroyedEventArgs>

Occurs when the request has been destroyed.

# Event ResponseBytesReceived

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Occurs when a part of HTTP response body has been received over the network.

```
event EventHandler<ResponseBytesReceivedEventArgs> ResponseBytesReceived
```

## Returns

EventHandler<ResponseBytesReceivedEventArgs>

Occurs when a part of HTTP response body has been received over the network.

# Event ResponseStarted

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Occurs when the response has been started.

```
event EventHandler<ResponseStartedEventArgs> ResponseStarted
```

## Returns

EventHandler<ResponseStartedEventArgs>

Occurs when the response has been started.

# Interface IUploadData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The upload data associated with a `UrlRequest`. The upload data can be in the `text/plain`, `application/x-www-form-urlencoded`, or `multipart/form-data` format.The upload data type depends on the content of the `Content-Type` header. If the `Content-Type` header is missing or doesn't include a valid substring indicating the corresponding upload data type, the upload data will be in a binary format.

```
public interface IUploadData
```

# Interface IUploadData<T>

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The upload data associated with a `UrlRequest`. The upload data can be in the `text/plain`, `application/x-www-form-urlencoded`, or `multipart/form-data` format.The upload data type depends on the content of the `Content-Type` header. If the `Content-Type` header is missing or doesn't include a valid substring indicating the corresponding upload data type, the upload data will be in a binary format.

```
public interface IUploadData<out T> : IUploadData
```

## Type Parameters

`T`

The upload data type.

## Properties

Data

Gets the upload data.

# Property Data

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Data

Gets the upload data.

```
T Data { get; }
```

## Property Value

T

# Class MimeType

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The MIME type.

```
public sealed class MimeType : TypedEnum<string>
```

**Inheritance**

object ← TypedEnum<string> ← MimeType

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

ApplicationFormUrlEncoded

Gets the "application/x-www-form-urlencoded" MIME type.

ApplicationJson

Gets the "application/json" MIME type.

ApplicationOctetStream

Gets the "application/octet-stream" MIME type.

ApplicationPdf

Gets the "application/pdf" MIME type.

ApplicationXml

Gets the "application/xml" MIME type.

ApplicationZip

Gets the "application/zip" MIME type.

ImageGif

Gets the "image/gif" MIME type.

ImageJpeg

Gets the "image/jpeg" MIME type.

ImagePng
Gets the "image/png" MIME type.

MultipartFormData
Gets the "multipart/form-data" MIME type.

TextCss
Gets the "text/css" MIME type.

TextHtml
Gets the "text/html" MIME type.

TextJavascript
Gets the "text/javascript" MIME type.

TextPlain
Gets the "text/plain" MIME type.

## Methods

Create(string)
Creates the MimeType instance from string representation.

## Operators

explicit operator MimeType(string)
Creates the MimeType instance from string representation.

# Field ApplicationFormUrlEncoded

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "application/x-www-form-urlencoded" MIME type.

```
public static readonly MimeType ApplicationFormUrlEncoded
```

## Returns

MimeType

The MimeType instance associated with the "application/x-www-form-urlencoded" MIME type.

# Field ApplicationJson

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "application/json" MIME type.

```
public static readonly MimeType ApplicationJson
```

## Returns

MimeType

The MimeType instance associated with the "application/json" MIME type.

# Field ApplicationOctetStream

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "application/octet-stream" MIME type.

```
public static readonly MimeType ApplicationOctetStream
```

## Returns

MimeType

The MimeType instance associated with the "application/octet-stream" MIME type.

# Field ApplicationPdf

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "application/pdf" MIME type.

```
public static readonly MimeType ApplicationPdf
```

## Returns

MimeType

The MimeType instance associated with the "application/pdf" MIME type.

# Field ApplicationXml

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "application/xml" MIME type.

```
public static readonly MimeType ApplicationXml
```

## Returns

MimeType

The MimeType instance associated with the "application/xml" MIME type.

# Field ApplicationZip

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "application/zip" MIME type.

```
public static readonly MimeType ApplicationZip
```

## Returns

MimeType

The MimeType instance associated with the "application/zip" MIME type.

# Field ImageGif

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "image/gif" MIME type.

```
public static readonly MimeType ImageGif
```

## Returns

MimeType

The MimeType instance associated with the "image/gif" MIME type.

# Field ImageJpeg

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "image/jpeg" MIME type.

```
public static readonly MimeType ImageJpeg
```

## Returns

MimeType

The MimeType instance associated with the "image/jpeg" MIME type.

# Field ImagePng

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "image/png" MIME type.

```
public static readonly MimeType ImagePng
```

## Returns

MimeType

The MimeType instance associated with the "image/png" MIME type.

# Field MultipartFormData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "multipart/form-data" MIME type.

```
public static readonly MimeType MultipartFormData
```

## Returns

MimeType

The MimeType instance associated with the "multipart/form-data" MIME type.

# Field TextCss

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "text/css" MIME type.

```
public static readonly MimeType TextCss
```

## Returns

MimeType

The MimeType instance associated with the "text/css" MIME type.

# Field TextHtml

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "text/html" MIME type.

```
public static readonly MimeType TextHtml
```

## Returns

MimeType

The MimeType instance associated with the "text/html" MIME type.

# Field TextJavascript

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "text/javascript" MIME type.

```
public static readonly MimeType TextJavascript
```

## Returns

MimeType

The MimeType instance associated with the "text/javascript" MIME type.

# Field TextPlain

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Gets the "text/plain" MIME type.

```
public static readonly MimeType TextPlain
```

## Returns

MimeType

The MimeType instance associated with the "text/plain" MIME type.

# Method Create

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Create(string)

Creates the MimeType instance from string representation.

```
public static MimeType Create(string value)
```

## Parameters

`value` string

string representation

## Returns

MimeType

the corresponding MimeType instance.

# Operator explicit operator

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## explicit operator MimeType(string)

Creates the MimeType instance from string representation.

```
public static explicit operator MimeType(string value)
```

### Parameters

`value` string

string representation

### Returns

MimeType

the corresponding MimeType instance.

# Class MultipartFormData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The list of key-value pairs each representing a segment of a multi-part form data. Can be empty if the form doesn't contain any data.

```
public sealed class MultipartFormData : UploadData,
IUploadData<IReadOnlyCollection<MultipartFormDataKeyValuePair>>, IUploadData
```

**Inheritance**

object ← UploadData ← MultipartFormData

**Implements**

IUploadData<IReadOnlyCollection<MultipartFormDataKeyValuePair>>, IUploadData

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

MultipartFormData(IReadOnlyCollection<MultipartFormDataKeyValuePair>)

Creates a new instance containing the specified key-value pairs.

## Properties

Data

The key-value pairs that represent segments of the multi-part form data.

# Constructor MultipartFormData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## MultipartFormData(IReadOnlyCollection<MultipartFormDataKeyValuePair>)

Creates a new instance containing the specified key-value pairs.

```
public MultipartFormData(IReadOnlyCollection<MultipartFormDataKeyValuePair> data)
```

### Parameters

`data` IReadOnlyCollection<MultipartFormDataKeyValuePair>

The key-value pairs that represent segments of the multi-part form data.

# Property Data

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Data

The key-value pairs that represent segments of the multi-part form data.

```
public IReadOnlyCollection<MultipartFormDataKeyValuePair> Data { get; }
```

## Property Value

IReadOnlyCollection<MultipartFormDataKeyValuePair>

# Class MultipartFormDataKeyValuePair

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

A key-value pair that represents a segment of a multi-part form data. Can contain values corresponding a form field content, an upload file content, etc.

```
public sealed class MultipartFormDataKeyValuePair
```

**Inheritance**

object ← MultipartFormDataKeyValuePair

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

MultipartFormDataKeyValuePair(string, FileValue)

Creates an instance from the given key and value.

MultipartFormDataKeyValuePair(string, string)

Creates an instance from the given key and value.

## Properties

FileValue

The file representation of the data.

Key

The form content segment key.

StringValue

The string representation of the data.

# Constructor MultipartFormDataKeyValuePair

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## MultipartFormDataKeyValuePair(string, FileValue)

Creates an instance from the given key and value.

```
public MultipartFormDataKeyValuePair(string key, FileValue value)
```

### Parameters

`key` string

The form content segment key.

`value` FileValue

The segment value representing the file content.

## MultipartFormDataKeyValuePair(string, string)

Creates an instance from the given key and value.

```
public MultipartFormDataKeyValuePair(string key, string value)
```

### Parameters

`key` string

The form content segment key.

`value` string

The segment value representing the form content segment value. Can be empty if the `key` does not correspond to any value. For example, an empty form field

# Property FileValue

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## FileValue

The file representation of the data.

```
public IFileValue FileValue { get; }
```

## Property Value

IFileValue

# Property Key

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Key

The form content segment key.

```
public string Key { get; }
```

## Property Value

string

# Property StringValue

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## StringValue

The string representation of the data.

```
public string StringValue { get; }
```

## Property Value

string

# Enum NetError

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The network errors.

```
public enum NetError
```

## Fields

`Aborted = -3`

An operation was aborted ( due to user action).

`AccessDenied = -10`

Permission to access a resource, other than the network, was denied.

`AddUserCertFailed = -503`

An error adding to the OS certificate database = e.g. OS X Keychain,.

`AddressInUse = -147`

Returned when attempting to bind an address that is already in use.

`AddressInvalid = -108`

The IP address or port number is invalid = e.g., cannot connect to the IP address 0 or the port 0.

`AddressUnreachable = -109`

The IP address is unreachable. This usually means that there is no route to the specified host or network.

`BadSslClientAuthCert = -117`

The SSL handshake failed because of a bad or missing client certificate.

`BlockedByAdministrator = -22`

The request was blocked by the URL blacklist configured by the domain administrator.

`BlockedByClient = -20`

The client chose to block the request.

`CacheChecksumMismatch = -408`

The cache found an entry with an invalid checksum. This can be returned from attempts to read from the cache. It is an internal error, returned by the SimpleCache backend, but not by any URLRequest methods or members.

`CacheChecksumReadFailure = -407`

The cache was unable to read a checksum record on an entry. This can be returned from attempts to read from the cache. It is an internal error, returned by the SimpleCache backend, but not by any URLRequest methods or members.

`CacheCreateFailure = -405`

The disk cache is unable to create this entry.

`CacheMiss = -400`

The cache does not have the requested entry.

`CacheOpenFailure = -404`

The disk cache is unable to open this entry.

`CacheOperationNotSupported = -403`

The operation is not supported for this entry.

`CacheRace = -406`

Multiple transactions are racing to create disk cache entries. This is an internal error returned from the HttpCache to the HttpCacheTransaction that tells the transaction to restart the entry-creation logic because the state of the cache has changed.

`CacheReadFailure = -401`

Unable to read from the disk cache.

`CacheWriteFailure = -402`

Unable to write to the disk cache.

`CertAuthorityInvalid = -202`

The server responded with a certificate that is signed by an authority we don't trust. The could mean:

1. An attacker has substituted the real certificate for a cert that contains his public key and is signed by his cousin.

2. The server operator has a legitimate certificate from a CA we don't know about, but should trust.

3. The server is presenting a self-signed certificate, providing no defense against active attackers = but foiling passive attackers.

`CertCommonNameInvalid = -200`

The server responded with a certificate whose common name did not match the host name. This could mean:

1. An attacker has redirected our traffic to his server and is presenting a certificate for which he knows the private key.

2. The server is misconfigured and responding with the wrong cert.

3. The user is on a wireless network and is being redirected to the network's login page.

4. The OS has used a DNS search suffix and the server doesn't have a certificate for the abbreviated name in the address bar.

`CertContainsErrors = -203`

The server responded with a certificate that contains errors. This error is not recoverable.

MSDN describes this error as follows: "The SSL certificate contains errors." NOTE: It's unclear how this differs from ERR_CERT_INVALID. For consistency, use that code instead of this one from now on.

`CertDateInvalid = -201`

The server responded with a certificate that, by our clock, appears to either not yet be valid or to have expired. This could mean:

1. An attacker is presenting an old certificate for which he has managed to obtain the private key.

2. The server is misconfigured and is not presenting a valid cert.

3. Our clock is wrong.

`CertEnd = -212`

The value immediately past the last certificate error code.

`CertErrorInSslRenegotiation = -116`

During SSL renegotiation = rehandshake,, the server sent a certificate with an error. Note: this error is not in the -2xx range so that it won't be handled as a certificate error.

`CertInvalid = -207`

The server responded with a certificate that is invalid. This error is not recoverable.

MSDN describes this error as follows: "The SSL certificate is invalid."

`CertNoRevocationMechanism = -204`

The certificate has no mechanism for determining if it is revoked. In effect, this certificate cannot be revoked.

`CertNonUniqueName = -210`

The host name specified in the certificate is not unique.

`CertRevoked = -206`

The server responded with a certificate has been revoked. We have the capability to ignore this error, but it is probably not the thing to do.

`CertUnableToCheckRevocation = -205`

Revocation information for the security certificate for this site is not available. This could mean:

1. An attacker has compromised the private key in the certificate and is blocking our attempt to find out that the cert was revoked.

2. The certificate is unrevoked, but the revocation server is busy or unavailable.

`CertWeakKey = -211`

The server responded with a certificate that contains a weak key = e.g. a too-small RSA key.

`CertWeakSignatureAlgorithm = -208`

The server responded with a certificate that is signed using a weak signature algorithm.

`ClientAuthCertTypeUnsupported = -151`

Server request for client certificate did not contain any types we support.

`ConnectionAborted = -103`

A connection timed out as a result of not receiving an ACK for data sent. This can include a FIN packet that did not get ACK'd.

`ConnectionClosed = -100`

A connection was closed = corresponding to a TCP FIN.

`ConnectionFailed = -104`

A connection attempt failed.

`ConnectionRefused = -102`

A connection attempt was refused.

`ConnectionReset = -101`

A connection was reset = corresponding to a TCP RST.

`ConnectionTimedOut = -118`

A connection attempt timed out.

`ContentDecodingFailed = -330`

Content decoding of the response body failed.

`ContentLengthMismatch = -354`

The HTTP response body transferred fewer bytes than were advertised by the Content-Length header when the connection is closed.

`DisallowedUrlScheme = -301`

The scheme of the URL is disallowed.

`DnsCacheMiss = -804`

The entry was not found in cache, for cache-only lookups.

`DnsMalformedResponse = -800`

DNS resolver received a malformed response.

`DnsSearchEmpty = -805`

Suffix search list rules prevent resolution of the given host name.

`DnsServerFailed = -802`

DNS server failed. This error is returned for all of the following error conditions:

1 - Format error - The name server was unable to interpret the query.

2 - Server failure - The name server was unable to process this query due to a problem with the name server.

4 - Not Implemented - The name server does not support the requested kind of query.

5 - Refused - The name server refuses to perform the specified operation for policy reasons.

`DnsServerRequiresTcp = -801`

DNS server requires TCP

`DnsSortError = -806`

Failed to sort addresses according to RFC3484.

`DnsTimedOut = -803`

DNS transaction timed out.

`EmptyResponse = -324`

The server closed the connection without sending any data.

`EncodingConversionFailed = -333`

Converting the response to target encoding failed.

`EncodingDetectionFailed = -340`

Detecting the encoding of the response failed.

`Failed = -2`

A generic failure occurred.

`FileExists = -16`

The file already exists.

`FileNoSpace = -18`

Not enough room left on the disk.

`FileNotFound = -6`

The file or directory cannot be found.

`FilePathTooLong = -17`

The path or file name is too long.

`FileTooBig = -8`

The file is too large.

`FileVirusInfected = -19`

The file has a virus.

`FtpBadCommandSequence = -607`

Server rejected our command because we didn't issue the commands in right order. FTP response code 503.

`FtpCommandNotSupported = -606`

Server does not support the command we issued. FTP response codes 502, 504.

`FtpFailed = -601`

A generic error for failed FTP control connection command. If possible, please use or add a more specific error code.

`FtpFileBusy = -604`

The file is busy, or some other temporary error condition on opening the file. FTP response code 450.

`FtpServiceUnavailable = -602`

The server cannot fulfill the request at this point. This is a temporary error. FTP response code 421.

`FtpSyntaxError = -605`

Server rejected our command because of syntax errors. FTP response codes 500, 501.

`FtpTransferAborted = -603`

The server has aborted the transfer. FTP response code 426.

`HeadersTruncated = -357`

The HTTP headers were truncated by an EOF.

`HostResolverQueueTooLarge = -119`

There are too many pending DNS resolves, so a request in the queue was aborted.

`HttpsProxyTunnelResponse = -140`

A request to create an SSL tunnel connection through the HTTPS proxy received a non-200 = OK, and non-407 = Proxy Auth, response. The response body might include a description of why the request failed.

`ImportCaCertFailed = -705`

CA import failed due to some other error.

`ImportCaCertNotCa = -703`

CA import failed - not a CA cert.

`ImportCertAlreadyExists = -704`

Import failed - certificate already exists in database. Note it's a little weird this is an error but reimporting a PKCS12 is ok = no-op,. That's how Mozilla does it, though.

`ImportServerCertFailed = -706`

Server certificate import failed due to some internal error.

`IncompleteChunkedEncoding = -355`

The HTTP response body is transferred with Chunked-Encoding, but the terminating zero-length chunk was never sent when the connection is closed.

`IncompleteSpdyHeaders = -347`

SPDY Headers have been received, but not all of them - status or version headers are missing, so we're expecting additional frames to complete them.

`InsecureResponse = -501`

The server's response was insecure = e.g. there was a cert error,.

`InsufficientResources = -12`

There were not enough resources to complete the operation.

`InternetDisconnected = -106`

The Internet connection has been lost.

`InvalidArgument = -4`

An argument to the function is incorrect.

`InvalidAuthCredentials = -338`

Credentials could not be established during HTTP Authentication.

`InvalidChunkedEncoding = -321`

Error in chunked transfer encoding.

`InvalidHandle = -5`

The handle or file descriptor is invalid.

`InvalidResponse = -320`

The server's response was invalid.

`InvalidSpdyStream = -335`

Attempted use of an unknown SPDY stream id.

`InvalidUrl = -300`

The URL is invalid.

`IoPending = -1`

An asynchronous IO operation is not yet complete. This usually does not indicate a fatal error. Typically this error will be generated as a notification to wait for some external notification that the IO operation finally completed.

`KeyGenerationFailed = -710`

Key generation failed.

`MalformedIdentity = -329`

The identity used for authentication is invalid.

`MandatoryProxyConfigurationFailed = -131`

A mandatory proxy configuration could not be used. Currently this means that a mandatory PAC script could not be fetched, parsed or executed.

`MethodNotSupported = -322`

The server did not support the request method.

`MisconfiguredAuthEnvironment = -343`

The environment was not set up correctly for authentication = for example, no KDC could be found or the principal is unknown.

`MissingAuthCredentials = -341`

= GSSAPI, No Kerberos credentials were available during HTTP Authentication.

`MsgTooBig = -142`

The message was too large for the transport. = for example a UDP message which exceeds size threshold,.

`NameNotResolved = -105`

The host name could not be resolved.

`NameResolutionFailed = -137`

An error occurred when trying to do a name resolution = DNS,.

`NetworkAccessDenied = -138`

Permission to access the network was denied. This is used to distinguish errors that were most likely caused by a firewall from other access denied errors. See also ERR_ACCESS_DENIED.

`NetworkChanged = -21`

The network changed.

`NetworkIoSuspended = -331`

An operation could not be completed because all network IO is suspended.

`NoPrivateKeyForCert = -502`

The server responded to a <keygen> with a generated client cert that we don't have the matching private key for.

`NoSslVersionsEnabled = -112`

No SSL protocol versions are enabled.

`NoSupportedProxies = -336`

There are no supported proxies in the provided list.

`NotImplemented = -11`

The operation failed because of unimplemented functionality.

`NpnNegotiationFailed = -122`

The request to negotiate an alternate protocol failed.

`Ok = 1`

No error.

`OriginBoundCertGenerationFailed = -711`

Server-bound certificate generation failed.

`OriginBoundCertGenerationTypeMismatch = -152`

Server requested one type of cert, then requested a different type while the first was still being generated.

`OutOfMemory = -13`

Memory allocation failed.

`PacNotInDhcp = -348`

No PAC URL configuration could be retrieved from DHCP. This can indicate either a failure to retrieve the DHCP configuration, or that there was no PAC URL configured in DHCP.

`PacScriptFailed = -327`

The evaluation of the PAC script failed.

`PacStatusNotOk = -326`

The PAC requested by HTTP did not have a valid status code = non-200.

`PipelineEviction = -353`

The request couldn't be completed on an HTTP pipeline. Client should retry.

`Pkcs12ImportBadPassword = -701`

PKCS #12 import failed due to incorrect password.

`Pkcs12ImportFailed = -702`

PKCS #12 import failed due to other error.

`Pkcs12ImportInvalidFile = -708`

PKCS #12 import failed due to invalid/corrupt file.

`Pkcs12ImportInvalidMac = -707`

PKCS #12 import failed due to invalid MAC.

`Pkcs12ImportUnsupported = -709`

PKCS #12 import failed due to unsupported features.

`PreconnectMaxSocketLimit = -133`

We've hit the max socket limit for the socket pool while preconnecting. We don't bother trying to preconnect more sockets.

`PrivateKeyExportFailed = -712`

Failure to export private key.

`ProtocolSwitched = -146`

Connection was aborted for switching to another protocol. WebSocket abort SocketStream connection when alternate protocol is found.

`ProxyAuthRequested = -127`

The proxy requested authentication = for tunnel establishment.

`ProxyAuthUnsupported = -115`

The proxy requested authentication = for tunnel establishment, with an unsupported method.

`ProxyCertificateInvalid = -136`

The certificate presented by the HTTPS Proxy was invalid.

`ProxyConnectionFailed = -130`

Could not create a connection to the proxy server. An error occurred either in resolving its name, or in connecting a socket to it. Note that this does NOT include failures during the actual "CONNECT" method of an HTTP proxy.

`QuicProtocolError = -356`

There is a QUIC protocol error.

`RequestRangeNotSatisfiable = -328`

The response was 416 = Requested range not satisfiable, and the server cannot satisfy the range requested.

`ResponseBodyTooBigToDrain = -345`

The HTTP response was too big to drain.

`ResponseHeadersMultipleContentDisposition = -349`

The HTTP response contained multiple Content-Disposition headers.

`ResponseHeadersMultipleContentLength = -346`

The HTTP response contained multiple distinct Content-Length headers.

`ResponseHeadersMultipleLocation = -350`

The HTTP response contained multiple Location headers.

`ResponseHeadersTooBig = -325`

The headers section of the response is too large.

`SocketIsConnected = -23`

The socket is already connected.

`SocketNotConnected = -15`

The socket is not connected.

`SocksConnectionFailed = -120`

Failed establishing a connection to the SOCKS proxy server for a target host.

`SocksConnectionHostUnreachable = -121`

The SOCKS proxy server failed establishing connection to the target host because that host is unreachable.

`SpdyPingFailed = -352`

SPDY server didn't respond to the PING message.

`SpdyProtocolError = -337`

There is a SPDY protocol error.

`SpdyServerRefusedStream = -351`

SPDY server refused the stream. Client should retry. This should never be a user-visible error.

`SpdySessionAlreadyExists = -143`

A SPDY session already exists, and should be used instead of this connection.

`SslBadPeerPublicKey = -149`

SSL peer's public key is invalid.

`SslBadRecordMacAlert = -126`

An SSL peer sent us a fatal bad_record_mac alert. This has been observed from servers with buggy DEFLATE support.

`SslClientAuthCertNeeded = -110`

The server requested a client certificate for SSL client authentication.

`SslClientAuthCertNoPrivateKey = -135`

The SSL client certificate has no private key.

`SslClientAuthPrivateKeyAccessDenied = -134`

The permission to use the SSL client certificate's private key was denied.

`SslClientAuthSignatureFailed = -141`

We were unable to sign the CertificateVerify data of an SSL client auth handshake with the client certificate's private key.

Possible causes for this include the user implicitly or explicitly denying access to the private key, the private key may not be valid for signing, the key may be relying on a cached handle which is no longer

valid, or the CSP won't allow arbitrary data to be signed.

`SslDecompressionFailureAlert = -125`

An SSL peer sent us a fatal decompression_failure alert. This typically occurs when a peer selects DEFLATE compression in the mistaken belief that it supports it.

`SslHandshakeNotCompleted = -148`

An operation failed because the SSL handshake has not completed.

`SslNoRenegotiation = -123`

The peer sent an SSL no_renegotiation alert message.

`SslPinnedKeyNotInCertChain = -150`

The certificate didn't match the built-in public key pins for the host name. The pins are set in net/http/transport_security_state.cc and require that one of a set of public keys exist on the path from the leaf to the root.

`SslProtocolError = -107`

An SSL protocol error occurred.

`SslRenegotiationRequested = -114`

The server requested a renegotiation = rehandshake,.

`SslUnsafeNegotiation = -128`

A known TLS strict server didn't offer the renegotiation extension.

`SslVersionOrCipherMismatch = -113`

The client and server don't support a common SSL protocol version or cipher suite.

`SslWeakServerEphemeralDhKey = -129`

The SSL server attempted to use a weak ephemeral Diffie-Hellman key.

`SynReplyNotReceived = -332`

FLIP data received without receiving a SYN_REPLY on the stream.

`TemporarilyThrottled = -139`

The request throttler module cancelled this request to avoid DDOS.

`TimedOut = -7`

An operation timed out.

`TooManyRedirects = -310`

Attempting to load an URL resulted in too many redirects.

`TunnelConnectionFailed = -111`

A tunnel connection through the proxy could not be established.

`Undefined = 0`

Undefined

`UndocumentedSecurityLibraryStatus = -344`

An undocumented SSPI or GSSAPI status code was returned.

`Unexpected = -9`

An unexpected error. This may be caused by a programming mistake or an invalid assumption.

`UnexpectedProxyAuth = -323`

The response was 407 = Proxy Authentication Required,, yet we did not send the request to a proxy.

`UnexpectedSecurityLibraryStatus = -342`

An unexpected, but documented, SSPI or GSSAPI status code was returned.

`UnknownUrlScheme = -302`

The scheme of the URL is unknown.

`UnrecognizedFtpDirectoryListingFormat = -334`

The server sent an FTP directory listing in a format we do not understand.

`UnsafePort = -312`

Attempting to load an URL with an unsafe port number. These are port numbers that correspond to services, which are not robust to spurious input that may be constructed as a result of an allowed web construct = e.g., HTTP looks a lot like SMTP, so form submission to port 25 is denied,.

`UnsafeRedirect = -311`

Attempting to load an URL resulted in an unsafe redirect = e.g., a redirect to file:// is considered unsafe.

`UnsupportedAuthScheme = -339`

An HTTP Authentication scheme was tried which is not supported on this machine.

`UploadFileChanged = -14`

The file upload failed because the file's modification time was different from the expectation.

`WinsockUnexpectedWrittenBytes = -124`

Winsock sometimes reports more data written than passed. This is probably due to a broken LSP.

`WsProtocolError = -145`

Websocket protocol error. Indicates that we are terminating the connection due to a malformed frame or other protocol violation.

# Enum RequestStatus

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The status of a URL request.

```
public enum RequestStatus
```

## Fields

`Canceled = 3`

A request was cancelled programmatically.

`Failed = 4`

A request failed for some reason.

`IoPending = 2`

An IO request is pending. It is expected that the request issuer will be informed when it is completed.

`Success = 1`

A request succeeded.

`Unknown = 0`

The request status is undefined.

`Unregognized = -1`

The request status is not recognized.

# Class Scheme

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The scheme component of a URL.

```
public sealed class Scheme : TypedEnum<string>
```

**Inheritance**

object ← TypedEnum<string> ← Scheme

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Http

The HTTP scheme.

Https

The HTTPS scheme.

## Properties

IsInterceptable

Indicates whether this scheme is interceptable.

## Methods

Create(string)

Creates the Scheme instance from string representation.

## Operators

explicit operator Scheme(string)

Creates the Scheme instance from string representation.

# Field Http

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The HTTP scheme.

```
public static readonly Scheme Http
```

## Returns

Scheme

The HTTP scheme.

# Field Https

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The HTTPS scheme.

```
public static readonly Scheme Https
```

## Returns

Scheme

The HTTPS scheme.

# Property IsInterceptable

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## IsInterceptable

Indicates whether this scheme is interceptable.

```
public bool IsInterceptable { get; }
```

## Property Value

bool

# Method Create

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Create(string)

Creates the Scheme instance from string representation.

```
public static Scheme Create(string value)
```

## Parameters

`value` string

string representation

## Returns

Scheme

the corresponding Scheme instance.

# Operator explicit operator

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## explicit operator Scheme(string)

Creates the Scheme instance from string representation.

```
public static explicit operator Scheme(string value)
```

## Parameters

`value` string

string representation

## Returns

Scheme

the corresponding Scheme instance.

# Enum SslVersion

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The supported SSL connection versions.

```
public enum SslVersion
```

## Fields

`Quic = 8`

QUIC

`Ssl2 = 2`

SSL 2.0

`Ssl3 = 3`

SSL 3.0

`Tls1 = 4`

TLS 1.0

`Tls11 = 5`

TLS 1.1

`Tls12 = 6`

TLS 1.2

`Tls13 = 7`

TLS 1.3

`Unknown = 1`

The SSL version cannot be determined.

`Unspecified = 0`

Reserved value.

# Class TextData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The upload data of the `text/plain` content type.

```
public sealed class TextData : UploadData, IUploadData<string>, IUploadData
```

**Inheritance**

object ← UploadData ← TextData

**Implements**

IUploadData<string>, IUploadData

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

TextData(string)

Creates a new instance containing the specified data.

## Properties

Data

The upload data as a `string`.

## Operators

implicit operator TextData(string)

Converts the specified data to a new instance of BytesData.

# Constructor TextData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## TextData(string)

Creates a new instance containing the specified data.

```
public TextData(string data)
```

## Parameters

`data` string

The upload data as a `string`.

# Property Data

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Data

The upload data as a `string`.

```
public string Data { get; }
```

## Property Value

string

# Operator implicit operator

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## implicit operator TextData(string)

Converts the specified data to a new instance of BytesData.

```
public static implicit operator TextData(string data)
```

## Parameters

`data` string

The upload data as a byte array.

## Returns

TextData

The new TextDatainstance.

# Class UploadData

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The base class for all upload data.

```
public abstract class UploadData : IUploadData
```

### Inheritance

object ← UploadData

### Implements

IUploadData

### Derived

BytesData, FormData, MultipartFormData, TextData

### Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Class UrlRequest

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Represents the URL request received from the Chromium engine.

```
public sealed class UrlRequest
```

**Inheritance**

object ← UrlRequest

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Browser

Gets the IBrowser instance associated with this request.

Method

Gets the HTTP method used to perform current request.

ResourceType

Gets the ResourceType of the resource the request is loading.

SslVersion

Gets the SSL connection version used to make this request if it is available and the current request represents an HTTPS request.

TotalBytesReceived

Gets the total amount of data received from network after SSL decoding and proxy handling.

TotalBytesSent

Gets the total amount of data sent over the network before SSL encoding and proxy handling.

Url

Gets the requested URL address.

## Methods

Equals(object)

GetHashCode()

# Property Browser

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with this request.

```
public IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance or null if the corresponding instance has not been determined or not available anymore.

# Property Method

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Method

Gets the HTTP method used to perform current request.

```
public string Method { get; }
```

## Property Value

string

the HTTP method used to perform current request.

# Property ResourceType

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## ResourceType

Gets the ResourceType of the resource the request is loading.

```
public ResourceType ResourceType { get; }
```

## Property Value

ResourceType

The type of the resource the request is loading.

# Property SslVersion

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## SslVersion

Gets the SSL connection version used to make this request if it is available and the current request represents an HTTPS request.

```
public SslVersion? SslVersion { get; }
```

## Property Value

SslVersion?

The SslVersion or null if it is not available.

# Property TotalBytesReceived

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## TotalBytesReceived

Gets the total amount of data received from network after SSL decoding and proxy handling.

```
public long TotalBytesReceived { get; }
```

## Property Value

long

The total amount of data received over the network.

# Property TotalBytesSent

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## TotalBytesSent

Gets the total amount of data sent over the network before SSL encoding and proxy handling.

```
public long TotalBytesSent { get; }
```

## Property Value

long

The total amount of data sent over the network.

# Property Url

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Url

Gets the requested URL address.

```
public string Url { get; }
```

## Property Value

string

The requested URL address.

# Method Equals

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class UrlRequestJob

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

The URL request job for the intercepted URL request, which allows you to provide the response data for this URL request.

```
public sealed class UrlRequestJob : IAutoDisposable
```

**Inheritance**

object ← UrlRequestJob

**Implements**

IAutoDisposable

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

IsClosed

Indicates if the request is handled and all the data is sent.

IsDisposed

Indicates if the object is already disposed.

UrlRequest

Gets the UrlRequest associated with this job.

## Methods

Complete()

Marks the request as completed. It can be used to indicate that all response data are already sent.

Fail()

Marks the request as failed. It can be used to indicate that an error occurred during writing the response data.

Write(byte[])
: Appends a chunk of the response data. This method may be called multiple times to append several chunks.

## Events

Disposed
: Occurs when the object has been disposed.

# Property IsClosed

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## IsClosed

Indicates if the request is handled and all the data is sent.

```
public bool IsClosed { get; }
```

## Property Value

bool

true if if the request is handled.

## Remarks

The request is marked as handled when either Complete() or Fail() is called.

# Property IsDisposed

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## IsDisposed

Indicates if the object is already disposed.

```
public bool IsDisposed { get; }
```

## Property Value

bool

`true` if the object is already disposed and no longer access to the members allowed. The access attempt may cause ObjectDisposedException or an unexpected behaviour.

# Property UrlRequest

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## UrlRequest

Gets the UrlRequest associated with this job.

```
public UrlRequest UrlRequest { get; }
```

## Property Value

UrlRequest

The UrlRequest associated with this job.

# Method Complete

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Complete()

Marks the request as completed. It can be used to indicate that all response data are already sent.

```
public void Complete()
```

# Method Fail

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Fail()

Marks the request as failed. It can be used to indicate that an error occurred during writing the response data.

```
public void Fail()
```

# Method Write

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

## Write(byte[])

Appends a chunk of the response data. This method may be called multiple times to append several chunks.

```
public void Write(byte[] data)
```

### Parameters

`data` byte[]

The response data bytes.

### Remarks

When all the chunks are written either the Complete() or Fail() method must be called to indicate that the request is completed or failed.

# Event Disposed

Namespace: DotNetBrowser.Net

Assembly: DotNetBrowser.dll

Occurs when the object has been disposed.

```
public event EventHandler Disposed
```

## Returns

EventHandler

Occurs when the object has been disposed.

**Implements**
Disposed

# Namespace DotNetBrowser.Net.Certificates

## Classes

Certificate

Provides information about the digital certificate. This certificate represents a X.509 certificate, which consists of a particular identity or end-entity certificate, such as an server identity or a client public key certificate, and zero or more intermediate certificates.

CertificateVerificationError

Provides information about an error found by Chromium when verifying an SSL certificate.

ExtendedKeyUsage

Defines how the certificate key can be used.

## Enums

CertificateVerificationStatus

The status that indicates result of SSL certificate verification by default Chromium certificate verifier.

# Class Certificate

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Provides information about the digital certificate. This certificate represents a X.509 certificate, which consists of a particular identity or end-entity certificate, such as an server identity or a client public key certificate, and zero or more intermediate certificates.

```
public sealed class Certificate
```

**Inheritance**

object ← Certificate

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object)

## Constructors

Certificate(X509Certificate)

Constructs a new Certificate instance from an X509Certificate.

Certificate(X509Certificate, X509CertificateCollection)

Constructs a new Certificate instance from an X509Certificate and the list of intermediate X.509 certificates associated with this the certificate that may be needed for chain building.

## Properties

CaFingerPrint

The CA Fingerprint of certificate.

Expired

Indicates whether the certificate has already expired.

ExtendedKeyUsages

A collection of extended key usages. This collection can be empty if the extended key usages info was not extracted from certificate because of corrupt data.

Fingerprint
A certificate fingerprint. Can be an empty string if the fingerprint was not extracted from certificate because of corrupt data.

IntermediateCertificates
Gets the list of intermediate certificates associated with this certificate that may be needed for chain building.

Issuer
The Issuer entity of certificate. Can be null if the issuer was not extracted from certificate because of corrupt certificate data.

IssuerName
The name of the issuer of the certificate.

KeyUsages
The key usages as a combination of flags. Can be null if the key usages info was not extracted from the certificate because of corrupt data.

NotAfter
The DateTime that describes until what time the certificate is valid.

NotBefore
The DateTime starting from the certificate is valid.

SerialNumber
The serial number of certificate. Can be an empty string if the serial number was not extracted from certificate because of corrupt data.

Subject
The Subject entity of certificate. Can be null if the subject was not extracted from certificate data because of corrupt certificate data.

SubjectName
The name of the subject of the certificate. For HTTPS server certificates, this represents the web server. The common name of the subject should match the host name of the web server.

X509Certificate
The X509Certificate that provides access to all certificate information. Can be null if the information was not extracted because of corrupt certificate data.

X509Certificate2

The X509Certificate2 that provides access to all certificate information. Can be null if the information was not extracted because of corrupt certificate data.

## Methods

ToString()

# Constructor Certificate

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## Certificate(X509Certificate)

Constructs a new Certificate instance from an X509Certificate.

```
public Certificate(X509Certificate certificate)
```

### Parameters

`certificate` X509Certificate

the certificate.

### Exceptions

ArgumentException

when value is null.

## Certificate(X509Certificate, X509CertificateCollection)

Constructs a new Certificate instance from an X509Certificate and the list of intermediate X.509 certificates associated with this the certificate that may be needed for chain building.

```
public Certificate(X509Certificate certificate, X509CertificateCollection
intermediateCertificates)
```

### Parameters

`certificate` X509Certificate

The X.509 certificate

`intermediateCertificates` X509CertificateCollection

The intermediate X.509 certificates associated with this the certificate.

## Exceptions

ArgumentException

when certificate or intermediate certificates are null.

# Property CaFingerPrint

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## CaFingerPrint

The CA Fingerprint of certificate.

```
public string CaFingerPrint { get; }
```

## Property Value

string

A string containing the CaFingerPrint.

# Property Expired

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## Expired

Indicates whether the certificate has already expired.

```
public bool Expired { get; }
```

## Property Value

bool

`false` if the certificate is not expired, otherwise - `true`.

# Property ExtendedKeyUsages

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## ExtendedKeyUsages

A collection of extended key usages. This collection can be empty if the extended key usages info was not extracted from certificate because of corrupt data.

```
public IEnumerable<ExtendedKeyUsage> ExtendedKeyUsages { get; }
```

## Property Value

IEnumerable<ExtendedKeyUsage>

The collection of `ExtendedKeyUsage`.

# Property Fingerprint

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## Fingerprint

A certificate fingerprint. Can be an empty string if the fingerprint was not extracted from certificate because of corrupt data.

```
public string Fingerprint { get; }
```

## Property Value

string

A string containing the fingerprint.

# Property IntermediateCertificates

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## IntermediateCertificates

Gets the list of intermediate certificates associated with this certificate that may be needed for chain building.

```
public IReadOnlyList<Certificate> IntermediateCertificates { get; }
```

## Property Value

IReadOnlyList<Certificate>

The list of intermediate certificates.

# Property Issuer

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## Issuer

The Issuer entity of certificate. Can be null if the issuer was not extracted from certificate because of corrupt certificate data.

```
public string Issuer { get; }
```

## Property Value

string

a string containing the Issuer.

# Property IssuerName

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## IssuerName

The name of the issuer of the certificate.

```
public string IssuerName { get; }
```

## Property Value

string

a string containing the IssuerName.

# Property KeyUsages

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## KeyUsages

The key usages as a combination of flags. Can be null if the key usages info was not extracted from the certificate because of corrupt data.

```
public X509KeyUsageFlags? KeyUsages { get; }
```

## Property Value

X509KeyUsageFlags?

returns the key usages as a combination of flags.

# Property NotAfter

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## NotAfter

The DateTime that describes until what time the certificate is valid.

```
public DateTime NotAfter { get; }
```

## Property Value

DateTime

Returns the time until what the certificate is valid as DataTime object

# Property NotBefore

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## NotBefore

The DateTime starting from the certificate is valid.

```
public DateTime NotBefore { get; }
```

## Property Value

DateTime

Returns the time from what the certificate is valid as DataTime object

# Property SerialNumber

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## SerialNumber

The serial number of certificate. Can be an empty string if the serial number was not extracted from certificate because of corrupt data.

```
public string SerialNumber { get; }
```

## Property Value

string

A string containing the SerialNumber.

# Property Subject

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## Subject

The Subject entity of certificate. Can be null if the subject was not extracted from certificate data because of corrupt certificate data.

```
public string Subject { get; }
```

## Property Value

string

A string containing the Subject.

# Property SubjectName

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## SubjectName

The name of the subject of the certificate. For HTTPS server certificates, this represents the web server. The common name of the subject should match the host name of the web server.

```
public string SubjectName { get; }
```

## Property Value

string

A string containing the SubjectName.

# Property X509Certificate

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## X509Certificate

The X509Certificate that provides access to all certificate information. Can be null if the information was not extracted because of corrupt certificate data.

```
public X509Certificate X509Certificate { get; }
```

## Property Value

X509Certificate

The X.509 certificate as an X509Certificate object.

# Property X509Certificate2

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## X509Certificate2

The X509Certificate2 that provides access to all certificate information. Can be null if the information was not extracted because of corrupt certificate data.

```
public X509Certificate2 X509Certificate2 { get; }
```

## Property Value

X509Certificate2

returns a X.509Certificate as an X509Certificate object.

# Method ToString

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class CertificateVerificationError

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Provides information about an error found by Chromium when verifying an SSL certificate.

```
public sealed class CertificateVerificationError
```

**Inheritance**

object ← CertificateVerificationError

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

DetailedDescription

Gets a detailed localized human-friendly description of the verification error.

ShortDescription

Gets a short localized human-friendly description of the verification error.

VerifyStatus

Gets the verification statuses that indicate result of SSL certificate verification by the Chromium engine.

## Methods

Equals(object)

GetHashCode()

# Property DetailedDescription

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## DetailedDescription

Gets a detailed localized human-friendly description of the verification error.

```
public string DetailedDescription { get; }
```

## Property Value

string

A string containing detailed localized human-friendly description of the verification error.

## Remarks

This message is formatted with HTML.

# Property ShortDescription

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## ShortDescription

Gets a short localized human-friendly description of the verification error.

```
public string ShortDescription { get; }
```

## Property Value

string

A string containing short localized human-friendly description of the verification error.

# Property VerifyStatus

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## VerifyStatus

Gets the verification statuses that indicate result of SSL certificate verification by the Chromium engine.

```
public CertificateVerificationStatus VerifyStatus { get; }
```

## Property Value

CertificateVerificationStatus

The verification statuses that indicate result of SSL certificate verification by the Chromium engine.

# Method Equals

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Enum CertificateVerificationStatus

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

The status that indicates result of SSL certificate verification by default Chromium certificate verifier.

```
public enum CertificateVerificationStatus
```

## Fields

`AuthorityInvalid = 3`

`CertificateTransparencyRequired = 18`

`CommonNameInvalid = 1`

`CtComplianceFailed = 17`

`DateInvalid = 2`

`Invalid = 7`

`IsEv = 14`

`NameConstraintViolation = 12`

`NoRevocationMechanism = 4`

`NonUniqueName = 9`

`PinnedKeyMissing = 11`

`RevCheckingEnabled = 15`

`Revoked = 6`

`Sha1SignaturePresent = 16`

`UnableToCheckRevocation = 5`

`ValidityTooLong = 13`

```
WeakKey = 10

WeakSignatureAlgorithm = 8
```

# Class ExtendedKeyUsage

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Defines how the certificate key can be used.

```
public sealed class ExtendedKeyUsage : TypedEnum<string>
```

**Inheritance**

object ← TypedEnum<string> ← ExtendedKeyUsage

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

ClientAuth

Gets the key can be used for TLS WWW client authentication.

CodeSigning

Gets the key can be used for signing of downloadable executable code.

EmailProtection

Gets the key can be used for email protection.

IpsecEndSystem

Gets the key can be used for IP security end system.

IpsecTunnel

Gets the key can be used for IP security tunnel termination.

IpsecUser

Gets the key can be used for IP security user.

OcspSigning

Gets the key can be used for signing OCSP responses.

ServerAuth

Gets the key can be used for TLS WWW server authentication.

TimeStamping

Gets the key can be used for binding the hash of an object to a time.

## Operators

explicit operator ExtendedKeyUsage(string)

Creates the ExtendedKeysUsage instance from string representation

# Field ClientAuth

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for TLS WWW client authentication.

```
public static readonly ExtendedKeyUsage ClientAuth
```

## Returns

ExtendedKeyUsage

The key can be used for TLS WWW client authentication.

# Field CodeSigning

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for signing of downloadable executable code.

```
public static readonly ExtendedKeyUsage CodeSigning
```

## Returns

ExtendedKeyUsage

The key can be used for signing of downloadable executable code.

# Field EmailProtection

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for email protection.

```
public static readonly ExtendedKeyUsage EmailProtection
```

## Returns

ExtendedKeyUsage

The key can be used for email protection.

# Field IpsecEndSystem

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for IP security end system.

```
public static readonly ExtendedKeyUsage IpsecEndSystem
```

## Returns

ExtendedKeyUsage

The key can be used for IP security end system.

# Field IpsecTunnel

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for IP security tunnel termination.

```
public static readonly ExtendedKeyUsage IpsecTunnel
```

## Returns

ExtendedKeyUsage

The key can be used for IP security tunnel termination.

# Field IpsecUser

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for IP security user.

```
public static readonly ExtendedKeyUsage IpsecUser
```

## Returns

ExtendedKeyUsage

The key can be used for IP security user.

# Field OcspSigning

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for signing OCSP responses.

```
public static readonly ExtendedKeyUsage OcspSigning
```

## Returns

ExtendedKeyUsage

The key can be used for signing OCSP responses.

# Field ServerAuth

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for TLS WWW server authentication.

```
public static readonly ExtendedKeyUsage ServerAuth
```

## Returns

ExtendedKeyUsage

The key can be used for TLS WWW server authentication.

# Field TimeStamping

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

Gets the key can be used for binding the hash of an object to a time.

```
public static readonly ExtendedKeyUsage TimeStamping
```

## Returns

ExtendedKeyUsage

The key can be used for binding the hash of an object to a time.

# Operator explicit operator

Namespace: DotNetBrowser.Net.Certificates

Assembly: DotNetBrowser.dll

## explicit operator ExtendedKeyUsage(string)

Creates the ExtendedKeysUsage instance from string representation

```
public static explicit operator ExtendedKeyUsage(string str)
```

## Parameters

`str` string

string representation

## Returns

ExtendedKeyUsage

a corresponding ExtendedKeyUsage instance.

# Namespace DotNetBrowser.Net.Events

## Classes

ConnectionTypeChangedEventArgs
: Event arguments for the ConnectionTypeChanged event.

NetworkEventArgs
: The base class for INetwork event arguments.

PacScriptErrorEventArgs
: Event arguments for the PacScriptErrorOccurred event.

RedirectResponseCodeReceivedEventArgs
: Event arguments for the RedirectResponseCodeReceived event.

RequestCompletedEventArgs
: Event arguments for the RequestCompleted event.

RequestDestroyedEventArgs
: Event arguments for the RequestDestroyed event.

ResponseBytesReceivedEventArgs
: Event arguments for the ResponseBytesReceived event.

ResponseStartedEventArgs
: Event arguments for the ResponseStarted event.

## Enums

ConnectionType
: The network connection type.

# Enum ConnectionType

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

The network connection type.

```
public enum ConnectionType
```

## Fields

`Bluetooth = 5`

A Bluetooth connection.

`Ethernet = 6`

An Ethernet connection.

`None = 7`

No connection.

`Type2G = 1`

A 2G connection.

`Type3G = 2`

A 3G connection.

`Type4G = 3`

A 4G connection.

`Type5G = 4`

A 5G connection.

`Unknown = 8`

A connection exists, but its type is unknown.

`Unspecified = 0`

Reserved value.

`Wifi = 9`

A Wi-Fi connection.

# Class ConnectionTypeChangedEventArgs

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

Event arguments for the ConnectionTypeChanged event.

```
public sealed class ConnectionTypeChangedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← ConnectionTypeChangedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ConnectionType

Gets the connection type of the network at the time of the event.

# Property ConnectionType

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## ConnectionType

Gets the connection type of the network at the time of the event.

```
public ConnectionType ConnectionType { get; }
```

## Property Value

ConnectionType

# Class NetworkEventArgs

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

The base class for INetwork event arguments.

```
public class NetworkEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← NetworkEventArgs

**Derived**

RedirectResponseCodeReceivedEventArgs, RequestCompletedEventArgs, RequestDestroyedEventArgs, ResponseBytesReceivedEventArgs, ResponseStartedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

UrlRequest

The URL request received from the Chromium engine.

# Property UrlRequest

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## UrlRequest

The URL request received from the Chromium engine.

```
public UrlRequest UrlRequest { get; }
```

## Property Value

UrlRequest

# Class PacScriptErrorEventArgs

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

Event arguments for the PacScriptErrorOccurred event.

```
public sealed class PacScriptErrorEventArgs
```

**Inheritance**

object ← PacScriptErrorEventArgs

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ErrorText

Gets the ErrorText.

LineNumber

Gets the LineNumber which contains the error.

# Property ErrorText

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## ErrorText

Gets the ErrorText.

```
public string ErrorText { get; }
```

## Property Value

string

The error text.

# Property LineNumber

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## LineNumber

Gets the LineNumber which contains the error.

```
public uint LineNumber { get; }
```

## Property Value

uint

The line number.

# Class RedirectResponseCodeReceivedEventArgs

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

Event arguments for the RedirectResponseCodeReceived event.

```
public sealed class RedirectResponseCodeReceivedEventArgs : NetworkEventArgs
```

**Inheritance**

object ← EventArgs ← NetworkEventArgs ← RedirectResponseCodeReceivedEventArgs

**Inherited Members**

NetworkEventArgs.UrlRequest , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

NewUrl

Gets the new URL.

ResponseCode

Gets the HTTP response code (e.g., 200, 404, and so on).

# Property NewUrl

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## NewUrl

Gets the new URL.

```
public string NewUrl { get; }
```

## Property Value

string

A string containing the new Url.

# Property ResponseCode

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## ResponseCode

Gets the HTTP response code (e.g., 200, 404, and so on).

```
public int ResponseCode { get; }
```

## Property Value

int

An integer that represents the response code.

# Class RequestCompletedEventArgs

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

Event arguments for the RequestCompleted event.

```
public sealed class RequestCompletedEventArgs : NetworkEventArgs
```

**Inheritance**

object ← EventArgs ← NetworkEventArgs ← RequestCompletedEventArgs

**Inherited Members**

NetworkEventArgs.UrlRequest , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ErrorCode

Gets the URL request error code. In case the request has failed, contains the error code of the network. Otherwise, has the default Undefined value.

IsCached

Indicates if the response has been taken from cache.

ResponseCode

Gets the response code of the current URL request (e.g., 200, 404, and so on).

Status

Gets the status of the current URL request.

# Property ErrorCode

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## ErrorCode

Gets the URL request error code. In case the request has failed, contains the error code of the network. Otherwise, has the default Undefined value.

```
public NetError ErrorCode { get; }
```

## Property Value

NetError

The URL request error code.

# Property IsCached

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## IsCached

Indicates if the response has been taken from cache.

```
public bool IsCached { get; }
```

## Property Value

bool

`true` if the response has been taken from cache, `false` otherwise.

# Property ResponseCode

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## ResponseCode

Gets the response code of the current URL request (e.g., 200, 404, and so on).

```
public int ResponseCode { get; }
```

## Property Value

int

The response code of the current URL request or 0 if the request has been failed or canceled.

# Property Status

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## Status

Gets the status of the current URL request.

```
public RequestStatus Status { get; }
```

## Property Value

RequestStatus

The status of the current URL request.

# Class RequestDestroyedEventArgs

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

Event arguments for the RequestDestroyed event.

```
public sealed class RequestDestroyedEventArgs : NetworkEventArgs
```

### Inheritance

object ← EventArgs ← NetworkEventArgs ← RequestDestroyedEventArgs

### Inherited Members

NetworkEventArgs.UrlRequest , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class ResponseBytesReceivedEventArgs

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

Event arguments for the ResponseBytesReceived event.

```
public sealed class ResponseBytesReceivedEventArgs : NetworkEventArgs
```

**Inheritance**

object ← EventArgs ← NetworkEventArgs ← ResponseBytesReceivedEventArgs

**Inherited Members**

NetworkEventArgs.UrlRequest , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Data

Gets the received part of HTTP response body.

MimeType

Gets the MIME type from the headers.

# Property Data

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## Data

Gets the received part of HTTP response body.

```
public byte[] Data { get; }
```

## Property Value

byte[]

An array of bytes containing the Data.

# Property MimeType

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## MimeType

Gets the MIME type from the headers.

```
public MimeType MimeType { get; }
```

## Property Value

MimeType

The MIME type from the headers.

# Class ResponseStartedEventArgs

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

Event arguments for the ResponseStarted event.

```
public class ResponseStartedEventArgs : NetworkEventArgs
```

**Inheritance**

object ← EventArgs ← NetworkEventArgs ← ResponseStartedEventArgs

**Inherited Members**

NetworkEventArgs.UrlRequest , EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

ResponseCode

Gets the response code of the current URL request (e.g., 200, 404, and so on). Returns 0 if the request has been failed or canceled.

# Property ResponseCode

Namespace: DotNetBrowser.Net.Events

Assembly: DotNetBrowser.dll

## ResponseCode

Gets the response code of the current URL request (e.g., 200, 404, and so on). Returns 0 if the request has been failed or canceled.

```
public int ResponseCode { get; }
```

## Property Value

int

The response code of the current URL request (e.g., 200, 404, and so on). Returns 0 if the request has been failed or canceled.

# Namespace DotNetBrowser.Net.Handlers

## Classes

AuthenticateParameters
The parameters of the AuthenticateHandler.

AuthenticateResponse
A response for the AuthenticateHandler.

CanAccessFileParameters
The parameters of the CanAccessFileHandler.

CanAccessFileResponse
A response for the CanAccessFileHandler.

CanGetCookiesParameters
The parameters of the CanGetCookiesHandler.

CanGetCookiesResponse
A response for the CanGetCookiesHandler.

CanSetCookieParameters
The parameters of the CanSetCookieHandler.

CanSetCookieResponse
A response for the CanSetCookieHandler.

InterceptRequestParameters
The parameters of the scheme handler.

InterceptRequestResponse
A response for the scheme handler.

NetworkParameters
The base class for INetwork-related handlers parameters.

ReceiveHeadersParameters
The parameters of the ReceiveHeadersHandler.

ReceiveHeadersResponse
The response for the ReceiveHeadersHandler.

SendUploadDataParameters

The parameters of the SendUploadDataHandler.

SendUploadDataResponse
The response for the SendUploadDataHandler.

SendUrlRequestParameters
The parameters of the SendUrlRequestHandler.

SendUrlRequestResponse
A response of the SendUrlRequestHandler.

StartTransactionParameters
The parameters of the StartTransactionHandler.

StartTransactionResponse
A response for the StartTransactionHandler.

UrlRequestJobOptions
The options needed to create the UrlRequestJob.

UrlRequestParameters
The base class for all the INetwork-related handlers parameters containing UrlRequest.

VerifyCertificateParameters
The parameters of the VerifyCertificateHandler.

VerifyCertificateResponse
A response for the VerifyCertificateHandler

## Enums

ResourceType
The type of the resource being loaded.

# Class AuthenticateParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the AuthenticateHandler.

```
public sealed class AuthenticateParameters : NetworkParameters
```

**Inheritance**

object ← NetworkParameters ← AuthenticateParameters

**Inherited Members**

NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object)

## Properties

Host

Gets the host of the service issuing the authentication challenge.

IsProxy

Indicates whether this authentication challenge came from a server or a proxy.

Port

Gets the port of the service issuing the authentication challenge.

Realm

Gets the realm of the authentication challenge. This method can return empty string.

Scheme

Gets the authentication scheme used, such as "basic" or "digest". In case of FTP server, this method returns empty string.

Url

Gets the URL of a web page that causes authentication.

# Methods

ToString()

# Property Host

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Host

Gets the host of the service issuing the authentication challenge.

```
public string Host { get; }
```

## Property Value

string

The host of the service issuing the authentication challenge.

# Property IsProxy

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## IsProxy

Indicates whether this authentication challenge came from a server or a proxy.

```
public bool IsProxy { get; }
```

## Property Value

bool

`true` if this authentication challenge came from a server or a proxy, `false` otherwise.

# Property Port

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Port

Gets the port of the service issuing the authentication challenge.

```
public int Port { get; }
```

## Property Value

int

The port of the service issuing the authentication challenge.

# Property Realm

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Realm

Gets the realm of the authentication challenge. This method can return empty string.

```
public string Realm { get; }
```

## Property Value

string

The realm of the authentication challenge. Can be null or empty string.

# Property Scheme

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Scheme

Gets the authentication scheme used, such as "basic" or "digest". In case of FTP server, this method returns empty string.

```
public string Scheme { get; }
```

## Property Value

string

The authentication scheme used, such as "basic" or "digest".

# Property Url

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of a web page that causes authentication.

```
public string Url { get; }
```

## Property Value

string

The URL of a web page that causes authentication.

# Method ToString

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Class AuthenticateResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

A response for the AuthenticateHandler.

```
public sealed class AuthenticateResponse
```

**Inheritance**

object ← AuthenticateResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Password

Gets the password that will be used for authenticating on a server or a proxy.

Username

Gets the username that will be used for authenticating on a server or a proxy.

## Methods

Cancel()

Creates a AuthenticateResponse that cancels authentication process.

Continue(string, string)

Creates a AuthenticateResponse that continues authentication process with the specified credentials.

# Property Password

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Password

Gets the password that will be used for authenticating on a server or a proxy.

```
public string Password { get; }
```

## Property Value

string

The

# Property Username

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Username

Gets the username that will be used for authenticating on a server or a proxy.

```
public string Username { get; }
```

## Property Value

string

The username that will be used for authenticating on a server or a proxy.

# Method Cancel

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a AuthenticateResponse that cancels authentication process.

```
public static AuthenticateResponse Cancel()
```

## Returns

AuthenticateResponse

The AuthenticateResponse instance that can be used as a return value in AuthenticateHandler implementation.

# Method Continue

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Continue(string, string)

Creates a AuthenticateResponse that continues authentication process with the specified credentials.

```
public static AuthenticateResponse Continue(string username, string password)
```

## Parameters

`username` string

Username used for authentication

`password` string

Password used for authentication

## Returns

AuthenticateResponse

The AuthenticateResponse instance that can be used as a return value in AuthenticateHandler implementation.

# Class CanAccessFileParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the CanAccessFileHandler.

```
public sealed class CanAccessFileParameters : NetworkParameters
```

**Inheritance**

object ← NetworkParameters ← CanAccessFileParameters

**Inherited Members**

NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AbsolutePath

Gets the absolute path to the file the request would like to access. On Linux and macOS returns the path after resolving all symbolic links. On Windows, if the requested file is a shortcut, the callback will be invoked twice: once for the shortcut, and once for the destination file path.

# Property AbsolutePath

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## AbsolutePath

Gets the absolute path to the file the request would like to access. On Linux and macOS returns the path after resolving all symbolic links. On Windows, if the requested file is a shortcut, the callback will be invoked twice: once for the shortcut, and once for the destination file path.

```
public string AbsolutePath { get; }
```

## Property Value

string

The absolute path to the file the request would like to access.

# Class CanAccessFileResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

A response for the CanAccessFileHandler.

```
public sealed class CanAccessFileResponse
```

**Inheritance**

object ← CanAccessFileResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Can()

Creates a CanAccessFileResponse that allows access to the file.

Cannot()

Creates a CanAccessFileResponse that denies access to the file.

# Method Can

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Can()

Creates a CanAccessFileResponse that allows access to the file.

```
public static CanAccessFileResponse Can()
```

## Returns

CanAccessFileResponse

The CanAccessFileResponse instance that can be used as a return value in CanAccessFileHandler implementation.

# Method Cannot

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Cannot()

Creates a CanAccessFileResponse that denies access to the file.

```
public static CanAccessFileResponse Cannot()
```

## Returns

CanAccessFileResponse

The CanAccessFileResponse instance that can be used as a return value in CanAccessFileHandler implementation.

# Class CanGetCookiesParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the CanGetCookiesHandler.

```
public sealed class CanGetCookiesParameters : NetworkParameters
```

**Inheritance**

object ← NetworkParameters ← CanGetCookiesParameters

**Inherited Members**

NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Cookies

Gets a collection of cookies that are going to be sent with the URL request.

Url

Gets the URL associated with the cookies.

# Property Cookies

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Cookies

Gets a collection of cookies that are going to be sent with the URL request.

```
public IEnumerable<Cookie> Cookies { get; }
```

## Property Value

IEnumerable<Cookie>

A collection of cookies that are going to be sent with the URL request.

# Property Url

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL associated with the cookies.

```
public string Url { get; }
```

## Property Value

string

The URL associated with the cookies.

# Class CanGetCookiesResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

A response for the CanGetCookiesHandler.

```
public sealed class CanGetCookiesResponse
```

**Inheritance**

object ← CanGetCookiesResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Allow()

Creates a CanGetCookiesResponse instance that allows sending cookies back to web server.

Deny()

Creates a CanGetCookiesResponse instance that denies sending cookies back to web server.

# Method Allow

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Allow()

Creates a CanGetCookiesResponse instance that allows sending cookies back to web server.

```
public static CanGetCookiesResponse Allow()
```

## Returns

CanGetCookiesResponse

The CanGetCookiesResponse instance that can be used as a return value in CanGetCookiesHandler implementation.

# Method Deny

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Deny()

Creates a CanGetCookiesResponse instance that denies sending cookies back to web server.

```
public static CanGetCookiesResponse Deny()
```

## Returns

CanGetCookiesResponse

The CanGetCookiesResponse instance that can be used as a return value in CanGetCookiesHandler implementation.

# Class CanSetCookieParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the CanSetCookieHandler.

```
public sealed class CanSetCookieParameters : NetworkParameters
```

**Inheritance**

object ← NetworkParameters ← CanSetCookieParameters

**Inherited Members**

NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Cookie

Gets the cookie received from the web server.

Url

Gets the URL associated with the cookie.

# Property Cookie

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Cookie

Gets the cookie received from the web server.

```
public Cookie Cookie { get; }
```

## Property Value

Cookie

The cookie received from the web server.

# Property Url

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL associated with the cookie.

```
public string Url { get; }
```

## Property Value

string

The URL associated with the cookie.

# Class CanSetCookieResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

A response for the CanSetCookieHandler.

```
public class CanSetCookieResponse
```

**Inheritance**

object ← CanSetCookieResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Allow()

Creates a CanSetCookieResponse that allows saving the cookie.

Deny()

Creates a CanSetCookieResponse that denies saving the cookie.

# Method Allow

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Allow()

Creates a CanSetCookieResponse that allows saving the cookie.

```
public static CanSetCookieResponse Allow()
```

## Returns

CanSetCookieResponse

The CanSetCookieResponse instance that can be used as a return value in CanSetCookieHandler implementation.

# Method Deny

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Deny()

Creates a CanSetCookieResponse that denies saving the cookie.

```
public static CanSetCookieResponse Deny()
```

## Returns

CanSetCookieResponse

The CanSetCookieResponse instance that can be used as a return value in CanSetCookieHandler implementation.

# Class InterceptRequestParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the scheme handler.

```
public sealed class InterceptRequestParameters : UrlRequestParameters
```

**Inheritance**

object ← NetworkParameters ← UrlRequestParameters ← InterceptRequestParameters

**Inherited Members**

UrlRequestParameters.UrlRequest , NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Headers

Gets the collection of HTTP headers sent in the scope of the request.

UploadData

Gets the data uploaded in the scope of the request.

# Property Headers

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Headers

Gets the collection of HTTP headers sent in the scope of the request.

```
public IEnumerable<IHttpHeader> Headers { get; }
```

## Property Value

IEnumerable<IHttpHeader>

The collection of HTTP headers sent in the scope of the request.

# Property UploadData

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## UploadData

Gets the data uploaded in the scope of the request.

```
public IUploadData UploadData { get; }
```

## Property Value

IUploadData

The data uploaded in the scope of the request.

# Class InterceptRequestResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

A response for the scheme handler.

```
public sealed class InterceptRequestResponse
```

**Inheritance**

object ← InterceptRequestResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Intercept(UrlRequestJob)

Creates a InterceptRequestResponse instance indicating that the request should be intercepted.

Proceed()

Creates a InterceptRequestResponse instance indicating that the request should be handled by the Chromium engine.

# Method Intercept

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Intercept(UrlRequestJob)

Creates a InterceptRequestResponse instance indicating that the request should be intercepted.

```
public static InterceptRequestResponse Intercept(UrlRequestJob job)
```

## Parameters

`job` UrlRequestJob

The UrlRequestJob providing data of the HTTP response that can be created using the CreateUrlRequestJob(UrlRequest, UrlRequestJobOptions) method.

## Returns

InterceptRequestResponse

The InterceptRequestResponse instance that can be used as a return value in handler implementation.

# Method Proceed

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Proceed()

Creates a InterceptRequestResponse instance indicating that the request should be handled by the Chromium engine.

```
public static InterceptRequestResponse Proceed()
```

## Returns

InterceptRequestResponse

The InterceptRequestResponse instance that can be used as a return value in the scheme handler implementation.

# Class NetworkParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The base class for INetwork-related handlers parameters.

```
public class NetworkParameters
```

**Inheritance**

object ← NetworkParameters

**Derived**

AuthenticateParameters, CanAccessFileParameters, CanGetCookiesParameters, CanSetCookieParameters, UrlRequestParameters, VerifyCertificateParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Network

The INetwork instance associated with the handler.

# Property Network

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Network

The INetwork instance associated with the handler.

```
public INetwork Network { get; }
```

## Property Value

INetwork

The INetwork instance associated with the handler.

# Class ReceiveHeadersParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the ReceiveHeadersHandler.

```
public sealed class ReceiveHeadersParameters : UrlRequestParameters
```

**Inheritance**

object ← NetworkParameters ← UrlRequestParameters ← ReceiveHeadersParameters

**Inherited Members**

UrlRequestParameters.UrlRequest , NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Charset

Gets the charset in lower case from the headers. If there's no charset, returns empty string.

ContentLength

Gets the number of the content length.

Headers

Gets the collection of HTTP headers with all its values.

MimeType

Gets the MIME type in lower case from the headers. If there's no MIME type, returns empty string.

StatusLine

Gets the first line of a Response message which is the StatusLine, consisting of the protocol version followed by a numeric status code and its associated textual phrase.

StatusText

Gets the status text.

# Property Charset

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Charset

Gets the charset in lower case from the headers. If there's no charset, returns empty string.

```
public string Charset { get; }
```

## Property Value

string

The charset in lower case from the headers.

# Property ContentLength

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## ContentLength

Gets the number of the content length.

```
public long ContentLength { get; }
```

## Property Value

long

The number of the content length.

# Property Headers

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Headers

Gets the collection of HTTP headers with all its values.

```
public IEnumerable<IHttpHeader> Headers { get; }
```

## Property Value

IEnumerable<IHttpHeader>

The collection of HTTP headers with all its values.

# Property MimeType

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## MimeType

Gets the MIME type in lower case from the headers. If there's no MIME type, returns empty string.

```
public MimeType MimeType { get; }
```

## Property Value

MimeType

The MIME type in lower case from the headers.

# Property StatusLine

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## StatusLine

Gets the first line of a Response message which is the StatusLine, consisting of the protocol version followed by a numeric status code and its associated textual phrase.

```
public string StatusLine { get; }
```

## Property Value

string

The first line of a Response message which is the StatusLine.

# Property StatusText

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## StatusText

Gets the status text.

```
public string StatusText { get; }
```

## Property Value

string

The status text.

# Class ReceiveHeadersResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The response for the ReceiveHeadersHandler.

```
public sealed class ReceiveHeadersResponse
```

**Inheritance**

object ← ReceiveHeadersResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

HttpHeaders

Gets the collection of the of HTTP headers with all its values.

## Methods

Continue()

Creates a ReceiveHeadersResponse that uses headers without modification.

OverrideHeaders(IEnumerable<HttpHeader>)

Creates a ReceiveHeadersResponse that overrides the received headers.

# Property HttpHeaders

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## HttpHeaders

Gets the collection of the of HTTP headers with all its values.

```
public IEnumerable<IHttpHeader> HttpHeaders { get; }
```

## Property Value

IEnumerable<IHttpHeader>

The collection of the of HTTP headers with all its values.

# Method Continue

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Continue()

Creates a ReceiveHeadersResponse that uses headers without modification.

```
public static ReceiveHeadersResponse Continue()
```

## Returns

ReceiveHeadersResponse

The ReceiveHeadersResponse instance that can be used as a return value in ReceiveHeadersHandler implementation.

# Method OverrideHeaders

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## OverrideHeaders(IEnumerable<HttpHeader>)

Creates a ReceiveHeadersResponse that overrides the received headers.

```
public static ReceiveHeadersResponse OverrideHeaders(IEnumerable<HttpHeader> httpHeaders)
```

## Parameters

`httpHeaders` IEnumerable<HttpHeader>

HttpHeaders used for overriding

## Returns

ReceiveHeadersResponse

The ReceiveHeadersResponse instance that can be used as a return value in ReceiveHeadersHandler implementation.

# Enum ResourceType

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The type of the resource being loaded.

```
public enum ResourceType
```

## Fields

`CspReport = 17`

The Content Security Policy report.

`Favicon = 13`

A favicon.

`FontResource = 6`

A font.

`Image = 5`

An image (jpg/gif/png/etc).

`MainFrame = 1`

Top level page.

`Media = 9`

A media resource.

`Object = 8`

An object (or embed) tag for a plugin, or a resource that a plugin requested.

`Ping = 15`

A ping request for <a ping>.

`PluginResource = 18`

The main plugin resource.

`Prefetch = 12`

An explicitly requested prefetch.

`Script = 4`

An external script.

`ServiceWorker = 16`

The main resource of a service worker.

`SharedWorker = 11`

The main resource of a shared worker.

`Stylesheet = 3`

A CSS stylesheet.

`SubFrame = 1`

Frame or IFRAME.

`SubResource = 7`

An "other" sub-resource.

`UnknownResource = 0`

The resource of the unknown type.

`Worker = 10`

The main resource of a dedicated worker.

`Xhr = 14`

A XMLHttpRequest.

# Class SendUploadDataParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SendUploadDataHandler.

```
public sealed class SendUploadDataParameters : UrlRequestParameters
```

**Inheritance**

object ← NetworkParameters ← UrlRequestParameters ← SendUploadDataParameters

**Inherited Members**

UrlRequestParameters.UrlRequest , NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

UploadData

The upload data associated with a `UrlRequest`. The upload data can be in the `text/plain`, `application/x-www-form-urlencoded`, or `multipart/form-data` format.The upload data type depends on the content of the `Content-Type` header.If the `Content-Type` header is missing or doesn't include a valid substring indicating the corresponding upload data type, the upload data will be in a binary format.

# Property UploadData

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## UploadData

The upload data associated with a `UrlRequest`. The upload data can be in the `text/plain`, `application/x-www-form-urlencoded`, or `multipart/form-data` format.The upload data type depends on the content of the `Content-Type` header.If the `Content-Type` header is missing or doesn't include a valid substring indicating the corresponding upload data type, the upload data will be in a binary format.

```
public IUploadData UploadData { get; }
```

## Property Value

IUploadData

The IUploadData instance representing the upload data associated with a `UrlRequest` or `null` if there are no associated upload data.

# Class SendUploadDataResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The response for the SendUploadDataHandler.

```
public sealed class SendUploadDataResponse
```

**Inheritance**

object ← SendUploadDataResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

UploadData

The upload data associated with a `UrlRequest`. The upload data can be in the `text/plain`, `application/x-www-form-urlencoded`, or `multipart/form-data` format.The upload data type depends on the content of the `Content-Type` header.If the `Content-Type` header is missing or doesn't include a valid substring indicating the corresponding upload data type, the upload data will be in a binary format.

## Methods

Continue()

Creates a SendUploadDataResponse that uses the original upload data.

Override(BytesData)

Creates a SendUploadDataResponse that overrides the upload data.

Override(FormData)

Creates a SendUploadDataResponse that overrides the upload data.

Override(MultipartFormData)

Creates a SendUploadDataResponse that overrides the upload data.

Override(TextData)

Creates a SendUploadDataResponse that overrides the upload data.

# Property UploadData

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## UploadData

The upload data associated with a `UrlRequest`. The upload data can be in the `text/plain`, `application/x-www-form-urlencoded`, or `multipart/form-data` format.The upload data type depends on the content of the `Content-Type` header.If the `Content-Type` header is missing or doesn't include a valid substring indicating the corresponding upload data type, the upload data will be in a binary format.

```
public IUploadData UploadData { get; }
```

## Property Value

IUploadData

The IUploadData instance representing the upload data associated with a `UrlRequest` or `null` if there are no associated upload data.

# Method Continue

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Continue()

Creates a SendUploadDataResponse that uses the original upload data.

```
public static SendUploadDataResponse Continue()
```

## Returns

SendUploadDataResponse

The SendUploadDataResponse instance that can be used as a return value in SendUploadDataHandler implementation.

# Method Override

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Override(BytesData)

Creates a SendUploadDataResponse that overrides the upload data.

```
public static SendUploadDataResponse Override(BytesData uploadData)
```

### Parameters

`uploadData` BytesData

ByteData used for overriding

### Returns

SendUploadDataResponse

The SendUploadDataResponse instance that can be used as a return value in SendUploadDataHandler implementation.

## Override(TextData)

Creates a SendUploadDataResponse that overrides the upload data.

```
public static SendUploadDataResponse Override(TextData uploadData)
```

### Parameters

`uploadData` TextData

TextData used for overriding

## Returns

SendUploadDataResponse

The SendUploadDataResponse instance that can be used as a return value in SendUploadDataHandler implementation.

### Override(FormData)

Creates a SendUploadDataResponse that overrides the upload data.

```
public static SendUploadDataResponse Override(FormData uploadData)
```

## Parameters

`uploadData` FormData

FormData used for overriding

## Returns

SendUploadDataResponse

The SendUploadDataResponse instance that can be used as a return value in SendUploadDataHandler implementation.

### Override(MultipartFormData)

Creates a SendUploadDataResponse that overrides the upload data.

```
public static SendUploadDataResponse Override(MultipartFormData uploadData)
```

## Parameters

`uploadData` MultipartFormData

MultipartFormData used for overriding

## Returns

SendUploadDataResponse

The SendUploadDataResponse instance that can be used as a return value in SendUploadDataHandler implementation.

# Class SendUrlRequestParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SendUrlRequestHandler.

```
public sealed class SendUrlRequestParameters : UrlRequestParameters
```

**Inheritance**

object ← NetworkParameters ← UrlRequestParameters ← SendUrlRequestParameters

**Inherited Members**

UrlRequestParameters.UrlRequest , NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Browser

Gets the IBrowser instance that initiated this request.

# Property Browser

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance that initiated this request.

```
public IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance or `null` if this instance has not been determined or not available anymore.

# Class SendUrlRequestResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

A response of the SendUrlRequestHandler.

```
public sealed class SendUrlRequestResponse
```

**Inheritance**

object ← SendUrlRequestResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

UrlToRequest

Gets the new URL to request.

## Methods

Cancel()

Creates a SendUrlRequestResponse that cancels loading process with the Aborted code.

Continue()

Creates a SendUrlRequestResponse that continues loading process with the requested URL.

Override(string)

Creates a SendUrlRequestResponse that overrides the requested URL.

# Property UrlToRequest

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## UrlToRequest

Gets the new URL to request.

```
public string UrlToRequest { get; }
```

## Property Value

string

The new URL to request.

# Method Cancel

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a SendUrlRequestResponse that cancels loading process with the Aborted code.

```
public static SendUrlRequestResponse Cancel()
```

## Returns

SendUrlRequestResponse

The SendUrlRequestResponse instance that can be used as a return value in SendUrlRequestHandler implementation.

# Method Continue

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Continue()

Creates a SendUrlRequestResponse that continues loading process with the requested URL.

```
public static SendUrlRequestResponse Continue()
```

## Returns

SendUrlRequestResponse

The SendUrlRequestResponse instance that can be used as a return value in SendUrlRequestHandler implementation.

# Method Override

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Override(string)

Creates a SendUrlRequestResponse that overrides the requested URL.

```
public static SendUrlRequestResponse Override(string url)
```

## Parameters

`url` string

New URL used for override

## Returns

SendUrlRequestResponse

The SendUrlRequestResponse instance that can be used as a return value in SendUrlRequestHandler implementation.

# Class StartTransactionParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the StartTransactionHandler.

```
public sealed class StartTransactionParameters : UrlRequestParameters
```

**Inheritance**

object ← NetworkParameters ← UrlRequestParameters ← StartTransactionParameters

**Inherited Members**

UrlRequestParameters.UrlRequest , NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Headers

Gets the collection of the of HTTP headers with all its values.

# Property Headers

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Headers

Gets the collection of the of HTTP headers with all its values.

```
public IEnumerable<IHttpHeader> Headers { get; }
```

## Property Value

IEnumerable<IHttpHeader>

The collection of the of HTTP headers with all its values.

# Class StartTransactionResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

A response for the StartTransactionHandler.

```
public sealed class StartTransactionResponse
```

**Inheritance**

object ← StartTransactionResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

HttpHeaders

Gets the collection of the of HTTP headers with all its values.

## Methods

Continue()

Creates a StartTransactionResponse that starts the transaction with the original headers.

OverrideHeaders(IEnumerable<HttpHeader>)

Creates a StartTransactionResponse that overrides headers and starts the transaction

# Property HttpHeaders

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## HttpHeaders

Gets the collection of the of HTTP headers with all its values.

```
public IEnumerable<IHttpHeader> HttpHeaders { get; }
```

## Property Value

IEnumerable<IHttpHeader>

The collection of the of HTTP headers with all its values.

# Method Continue

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Continue()

Creates a StartTransactionResponse that starts the transaction with the original headers.

```
public static StartTransactionResponse Continue()
```

## Returns

StartTransactionResponse

The StartTransactionResponse instance that can be used as a return value in StartTransactionHandler implementation.

# Method OverrideHeaders

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## OverrideHeaders(IEnumerable<HttpHeader>)

Creates a StartTransactionResponse that overrides headers and starts the transaction

```
public static StartTransactionResponse OverrideHeaders(IEnumerable<HttpHeader> httpHeaders)
```

### Parameters

`httpHeaders` IEnumerable<HttpHeader>

HttpHeaders used for overriding

### Returns

StartTransactionResponse

The StartTransactionResponse instance that can be used as a return value in StartTransactionHandler implementation.

# Class UrlRequestJobOptions

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The options needed to create the UrlRequestJob.

```
public sealed class UrlRequestJobOptions
```

**Inheritance**

object ← UrlRequestJobOptions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Headers

Gets or sets the list of the HTTP headers that will be a part of the response.

HttpStatusCode

Gets or sets the HTTP status code of the response.

# Property Headers

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Headers

Gets or sets the list of the HTTP headers that will be a part of the response.

```
public List<HttpHeader> Headers { get; set; }
```

## Property Value

List<HttpHeader>

he list of the HTTP headers that will be a part of the response.

# Property HttpStatusCode

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## HttpStatusCode

Gets or sets the HTTP status code of the response.

```
public HttpStatusCode HttpStatusCode { get; set; }
```

## Property Value

HttpStatusCode

The HTTP status code of the response. The default value is `HttpStatusCode.OK`.

# Class UrlRequestParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The base class for all the INetwork-related handlers parameters containing UrlRequest.

```
public class UrlRequestParameters : NetworkParameters
```

**Inheritance**

object ← NetworkParameters ← UrlRequestParameters

**Derived**

InterceptRequestParameters, ReceiveHeadersParameters, SendUploadDataParameters, SendUrlRequestParameters, StartTransactionParameters

**Inherited Members**

NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

UrlRequest

The URL request received from the Chromium engine.

# Property UrlRequest

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## UrlRequest

The URL request received from the Chromium engine.

```
public UrlRequest UrlRequest { get; }
```

## Property Value

UrlRequest

# Class VerifyCertificateParameters

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

The parameters of the VerifyCertificateHandler.

```
public class VerifyCertificateParameters : NetworkParameters
```

**Inheritance**

object ← NetworkParameters ← VerifyCertificateParameters

**Inherited Members**

NetworkParameters.Network , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Certificate

Gets the SSL certificate to verify.

HostName

Gets the SSL server host name.

VerifyErrors

Gets the verification errors that indicate result of SSL certificate verification by the Chromium engine. If this collection is empty, then default SSL certificate verifier couldn't find any issues with the given certificate, so it's a valid certificate.

# Property Certificate

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Certificate

Gets the SSL certificate to verify.

```
public Certificate Certificate { get; }
```

## Property Value

Certificate

The SSL certificate to verify.

# Property HostName

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## HostName

Gets the SSL server host name.

```
public string HostName { get; }
```

## Property Value

string

The SSL server host name.

# Property VerifyErrors

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## VerifyErrors

Gets the verification errors that indicate result of SSL certificate verification by the Chromium engine. If this collection is empty, then default SSL certificate verifier couldn't find any issues with the given certificate, so it's a valid certificate.

```
public IEnumerable<CertificateVerificationError> VerifyErrors { get; }
```

## Property Value

IEnumerable<CertificateVerificationError>

The verification errors that indicate result of SSL certificate verification by the Chromium engine.

# Class VerifyCertificateResponse

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

A response for the VerifyCertificateHandler

```
public sealed class VerifyCertificateResponse
```

**Inheritance**

object ← VerifyCertificateResponse

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Default()

Creates a VerifyCertificateResponse that lets Chromium decide whether SSL certificate should be accepted or rejected.

Equals(object)

GetHashCode()

Invalid()

Creates a VerifyCertificateResponse that notifies the engine that the given certificate is invalid with the Invalid.

Invalid(IList<CertificateVerificationStatus>)

Creates a VerifyCertificateResponse that notifies the engine that the given certificate is invalid with the specific CertificateVerificationStatus.

Valid()

Creates a VerifyCertificateResponse that marks SSL certificate as valid and must be accepted.

# Method Default

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Default()

Creates a VerifyCertificateResponse that lets Chromium decide whether SSL certificate should be accepted or rejected.

```
public static VerifyCertificateResponse Default()
```

## Returns

VerifyCertificateResponse

The VerifyCertificateResponse instance that can be used as a return value in VerifyCertificateHandler implementation.

# Method Equals

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Method Invalid

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Invalid()

Creates a VerifyCertificateResponse that notifies the engine that the given certificate is invalid with the Invalid.

```
public static VerifyCertificateResponse Invalid()
```

### Returns

VerifyCertificateResponse

The VerifyCertificateResponse instance that can be used as a return value in VerifyCertificateHandler implementation.

## Invalid(IList<CertificateVerificationStatus>)

Creates a VerifyCertificateResponse that notifies the engine that the given certificate is invalid with the specific CertificateVerificationStatus.

```
public static VerifyCertificateResponse Invalid(IList<CertificateVerificationStatus>
certificateVerificationStatuses)
```

### Parameters

`certificateVerificationStatuses` IList<CertificateVerificationStatus>

The list of statuses that explain why the certificate is invalid.

### Returns

VerifyCertificateResponse

The VerifyCertificateResponse instance that can be used as a return value in VerifyCertificateHandler implementation.

## Remarks

If the status list is null or empty the Invalid value will be used.

# Method Valid

Namespace: DotNetBrowser.Net.Handlers

Assembly: DotNetBrowser.dll

## Valid()

Creates a VerifyCertificateResponse that marks SSL certificate as valid and must be accepted.

```
public static VerifyCertificateResponse Valid()
```

## Returns

VerifyCertificateResponse

The VerifyCertificateResponse instance that can be used as a return value in VerifyCertificateHandler implementation.

# Namespace DotNetBrowser.Net.Proxy

## Classes

AutoDetectProxySettings
: With this proxy configuration the connection automatically detects proxy settings.

CustomProxySettings
: Describes a user's proxy settings.

DirectProxySettings
: With this proxy configuration the connection doesn't use a proxy server.

PacProxySettings
: With this proxy configuration the connection uses proxy settings received from proxy auto-config (PAC) file which is located at the specific address.

  In case PAC script fails, a direct connection will be used.

ProxySettings
: The base class for all proxy settings.

SystemProxySettings
: The system proxy settings defined by the operating system.

## Interfaces

IProxy
: The service that allows modifying the proxy configuration for the current engine.

## Enums

ProxyType
: Types of proxy settings.

# Class AutoDetectProxySettings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

With this proxy configuration the connection automatically detects proxy settings.

```
public sealed class AutoDetectProxySettings : ProxySettings
```

### Inheritance

object ← ProxySettings ← AutoDetectProxySettings

### Inherited Members

ProxySettings.Type , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class CustomProxySettings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

Describes a user's proxy settings.

```
public sealed class CustomProxySettings : ProxySettings
```

**Inheritance**

object ← ProxySettings ← CustomProxySettings

**Inherited Members**

ProxySettings.Type , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

CustomProxySettings(string, string)

Constructs a new CustomProxySettings instance with given proxy rules.

## Properties

Exceptions

The set of URLs that should bypass the proxy settings.

Rules

The proxy rules in specified format. The string should be a semicolon-separated list of ordered proxies that apply to a particular URL scheme.

# Constructor CustomProxySettings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## CustomProxySettings(string, string)

Constructs a new CustomProxySettings instance with given proxy rules.

```
public CustomProxySettings(string rules, string exceptions = null)
```

## Parameters

`rules` string

string that represents proxy rules in specified format. The string should be a semicolon-separated list of ordered proxies that apply to a particular URL scheme.

`exceptions` string

string that represents the set of URLs that should bypass the proxy settings.

## Remarks

Examples of the proxy rules string:

- `"http=foopy:80;ftp=foopy2"` use HTTP proxy "foopy:80" for `http://` URLs, and HTTP proxy "foopy2:80" for `%ftp://` URLs.
- `"socks4://foopy"` use SOCKS v4 proxy "foopy:1080" for all URLs.

## Exceptions

ArgumentException

Thrown when `rules` is null, empty or contain only white space.

# Property Exceptions

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## Exceptions

The set of URLs that should bypass the proxy settings.

```
public string Exceptions { get; }
```

## Property Value

string

## Remarks

The format of the exceptions string can be any of the following:

- `[ URL_SCHEME "://" ] HOSTNAME_PATTERN [ ":" <port> ]` Examples: "foobar.com", "*foobar.com", "*.foobar.com", "*foobar.com:99", "https://x.*.y.com:99"
- `"." HOSTNAME_SUFFIX_PATTERN [ ":" PORT ]` Examples: ".google.com", ".com", "http://.google.com"
- `[ SCHEME "://" ] IP_LITERAL [ ":" PORT ]` Examples: "127.0.1", "[0:0::1]", "[::1]", "http://[::1]:99"
- `IP_LITERAL "/" PREFIX_LENGHT_IN_BITS` Examples: "192.168.1.1/16", "fefe:13::abc/33"
- `"<local>"` Match local addresses. The meaning of "<local>" is whether the host matches one of: "127.0.0.1", "::1", "localhost".

If you need to provide several exception rules you can separate them using comma: "*foobar.com,.google.com,<local>".

# Property Rules

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## Rules

The proxy rules in specified format. The string should be a semicolon-separated list of ordered proxies that apply to a particular URL scheme.

```
public string Rules { get; set; }
```

## Property Value

string

## Remarks

Examples of the proxyRules string:

- `"http=foopy:80;ftp=foopy2"` - use HTTP proxy "foopy:80" for `http://` URLs, and HTTP proxy "foopy2:80" for `%ftp://` URLs.
- `"foopy:80"` - use HTTP proxy "foopy:80" for all URLs.
- `"socks4://foopy"` - use SOCKS v4 proxy "foopy:1080" for all URLs.

# Class DirectProxySettings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

With this proxy configuration the connection doesn't use a proxy server.

```
public sealed class DirectProxySettings : ProxySettings
```

### Inheritance

object ← ProxySettings ← DirectProxySettings

### Inherited Members

ProxySettings.Type , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Interface IProxy

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

The service that allows modifying the proxy configuration for the current engine.

```
public interface IProxy : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Engine

Gets the IEngine instance associated with this object.

Profile

Gets the IProfile instance associated with this object.

Settings

Gets or sets the current proxy settings.

# Property Engine

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Profile

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Property Settings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## Settings

Gets or sets the current proxy settings.

```
ProxySettings Settings { get; set; }
```

## Property Value

ProxySettings

The proxy settings used for loading the websites.

# Class PacProxySettings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

With this proxy configuration the connection uses proxy settings received from proxy auto-config (PAC) file which is located at the specific address.

In case PAC script fails, a direct connection will be used.

```
public sealed class PacProxySettings : ProxySettings
```

**Inheritance**

object ← ProxySettings ← PacProxySettings

**Inherited Members**

ProxySettings.Type , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

PacProxySettings(string)

Constructs a new PacProxySettings instance initiated with the pacUrl URL address where PAC file with proxy settings is located. The address of the PAC file must be a valid URL address. The address can contain a path to the PAC file on local file system.

## Properties

PacUrl

The URL address of the PAC file with proxy settings.

# Constructor PacProxySettings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## PacProxySettings(string)

Constructs a new PacProxySettings instance initiated with the pacUrl URL address where PAC file with proxy settings is located. The address of the PAC file must be a valid URL address. The address can contain a path to the PAC file on local file system.

```
public PacProxySettings(string pacUrl)
```

### Parameters

`pacUrl` string

URL address of the PAC file with proxy settings.

### Exceptions

ArgumentException

Thrown when `pacUrl` is null, empty or contain only white space.

# Property PacUrl

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## PacUrl

The URL address of the PAC file with proxy settings.

```
public string PacUrl { get; set; }
```

## Property Value

string

# Class ProxySettings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

The base class for all proxy settings.

```
public abstract class ProxySettings
```

**Inheritance**

object ← ProxySettings

**Derived**

AutoDetectProxySettings, CustomProxySettings, DirectProxySettings, PacProxySettings, SystemProxySettings

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Type

Gets the type of proxy settings.

# Property Type

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

## Type

Gets the type of proxy settings.

```
public ProxyType Type { get; }
```

## Property Value

ProxyType

# Enum ProxyType

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

Types of proxy settings.

```
public enum ProxyType
```

## Fields

`AutoDetect = 1`

The automatically detected proxy settings.

`Custom = 3`

The custom proxy settings.

`Direct = 0`

The direct connection without proxy server.

`Pac = 2`

The proxy settings obtained from the PAC file.

`System = -1`

The proxy settings defined in operating system.

# Class SystemProxySettings

Namespace: DotNetBrowser.Net.Proxy

Assembly: DotNetBrowser.dll

The system proxy settings defined by the operating system.

```
public sealed class SystemProxySettings : ProxySettings
```

### Inheritance

object ← ProxySettings ← SystemProxySettings

### Inherited Members

ProxySettings.Type , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Namespace DotNetBrowser.Passwords

## Classes

PasswordRecord

A record saved in the IPasswordStore.

## Interfaces

IPasswordStore

A service that allows working with PasswordRecord logins and passwords saved in the Chromium password store.

IPasswords

A service that allows managing passwords.

# Interface IPasswordStore

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

A service that allows working with PasswordRecord logins and passwords saved in the Chromium password store.

```
public interface IPasswordStore
```

## Properties

All
Gets all records from the password store including blacklisted ones.

AllNeverSaved
Gets only blacklisted (marked as "never-saved") records from the password store.

AllSaved
Gets only Save saved records from the password store.

Profile
Gets the IProfile instance associated wit this IPasswordStore.

## Methods

Clear()
Clears all records in the password store.

RemoveByUrl(string)
Removes records from the password store associated with this url.

# Property All

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## All

Gets all records from the password store including blacklisted ones.

```
IReadOnlyList<PasswordRecord> All { get; }
```

## Property Value

IReadOnlyList<PasswordRecord>

All records from the password store including blacklisted ones.

## Remarks

The blacklisted entries will have an empty Login property.

## Exceptions

ObjectDisposedException

The IPasswordStore has already been disposed.

# Property AllNeverSaved

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## AllNeverSaved

Gets only blacklisted (marked as "never-saved") records from the password store.

```
IReadOnlyList<PasswordRecord> AllNeverSaved { get; }
```

## Property Value

IReadOnlyList<PasswordRecord>

Only blacklisted (marked as "never-saved") records from the password store.

## Exceptions

ObjectDisposedException

The IPasswordStore has already been disposed.

# Property AllSaved

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## AllSaved

Gets only Save saved records from the password store.

```
IReadOnlyList<PasswordRecord> AllSaved { get; }
```

## Property Value

IReadOnlyList<PasswordRecord>

Only Save saved records from the password store.

## Exceptions

ObjectDisposedException

The IPasswordStore has already been disposed.

# Property Profile

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated wit this IPasswordStore.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated wit this IPasswordStore.

# Method Clear

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## Clear()

Clears all records in the password store.

```
void Clear()
```

## Exceptions

ObjectDisposedException

The IPasswordStore has already been disposed.

# Method RemoveByUrl

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## RemoveByUrl(string)

Removes records from the password store associated with this url.

```
void RemoveByUrl(string url)
```

## Parameters

`url` string

The URL associated with the removed password records.

## Exceptions

ObjectDisposedException

The IPasswordStore has already been disposed.

ArgumentException

The `url` is null, empty, or contains only blank characters.

# Interface IPasswords

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

A service that allows managing passwords.

```
public interface IPasswords : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Browser
: Gets the IBrowser instance associated with this browser.

SavePasswordHandler
: Gets or sets a handler that is used when the user is prompted to save the credentials in the password store

UpdatePasswordHandler
: Gets or sets a handler that is used when the user is prompted to update the password in the password store

# Property Browser

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with this browser.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with this IPasswords instance.

# Property SavePasswordHandler

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## SavePasswordHandler

Gets or sets a handler that is used when the user is prompted to save the credentials in the password store

```
IHandler<SavePasswordParameters, SavePasswordResponse> SavePasswordHandler { get; set; }
```

## Property Value

IHandler<SavePasswordParameters, SavePasswordResponse>

## Remarks

This handler is equivalent to the "Save Password?" bubble in Chromium.

The handler is invoked when the user submits a new web form (a form that has never been submitted before) or submits the web from with a new login value.

Use the Save to save these login and password in the password store. Stored credentials will be shown next time in the suggestions pop-up when focusing the input associated with login or password.

Use the NeverSave to blacklist password forms on this URL. After this, the handler will never be invoked again until you remove this blacklisted record from the password store.

Use the Ignore to ignore the suggestion to save the login and password in the password store. The handler will be invoked again if this or another web form is submitted on the same URL.

The prerequisites for the handler invocation:

- The login and password should not be empty.
- The scheme should be web-safe.The list of web-safe schemes: HTTP, HTTPS, WS, and WSS.
- The profile should not be incognito.
- The valid SSL certificate is used.

- A positive server response on the web form submission.

If an exception occurs inside the handler implementation, the default behavior will be applied - the response NeverSave will be used.

## Exceptions

ObjectDisposedException

The IPasswords has already been disposed.

# Property UpdatePasswordHandler

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## UpdatePasswordHandler

Gets or sets a handler that is used when the user is prompted to update the password in the password store

```
IHandler<UpdatePasswordParameters, UpdatePasswordResponse> UpdatePasswordHandler { get;
set; }
```

## Property Value

IHandler<UpdatePasswordParameters, UpdatePasswordResponse>

## Remarks

This handler is equivalent to the "Update Password?" bubble in Chromium.

The handler can be invoked only if the login and password from the form were previously saved in the password store and now user entered a new password value.

Use the Update method to update the password for this login in the password store . The updated password will be reflected in the suggestions pop-up.

Use the Ignore method to stay with the old password. If you submit the form with the same password value next time the handler will be invoked again.

The prerequisites for the handler invocation:

- The password should not be empty.
- The scheme should be web-safe.The list of web-safe schemes: HTTP, HTTPS, WS, and WSS.
- The IProfile profile should not be incognito.
- The valid SSL certificate is used.
- A positive server response on the web form submission.

If an exception occurs inside the handler implementation, the default behavior will be applied - the response Ignore will be used.

## Exceptions

ObjectDisposedException

The IPasswords has already been disposed.

# Class PasswordRecord

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

A record saved in the IPasswordStore.

```
public sealed class PasswordRecord
```

**Inheritance**

object ← PasswordRecord

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Login

Gets the user's login.

Url

Gets the URL that contains scheme, host and port of the resource where the form is submitted.

## Methods

Equals(object)

GetHashCode()

# Property Login

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## Login

Gets the user's login.

```
public string Login { get; }
```

## Property Value

string

The user's login.

## Remarks

For the blacklisted records the login is empty.

# Property Url

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## Url

Gets the URL that contains scheme, host and port of the resource where the form is submitted.

```
public string Url { get; }
```

## Property Value

string

The URL.

# Method Equals

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Passwords

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Namespace DotNetBrowser.Passwords.Handlers

## Classes

SavePasswordParameters

The parameters of the SavePasswordHandler.

SavePasswordResponse

A response to the SavePasswordHandler.

UpdatePasswordParameters

The parameters of the UpdatePasswordHandler.

UpdatePasswordResponse

A response to the UpdatePasswordHandler.

# Class SavePasswordParameters

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SavePasswordHandler.

```
public sealed class SavePasswordParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← SavePasswordParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Login

Gets the login value passed in the form.

Url

Gets the URL of the resource where the form is located.

# Property Login

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

## Login

Gets the login value passed in the form.

```
public string Login { get; }
```

## Property Value

string

The login value passed in the form.

# Property Url

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the resource where the form is located.

```
public string Url { get; }
```

## Property Value

string

The URL of the resource where the form is located.

# Class SavePasswordResponse

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

A response to the SavePasswordHandler.

```
public sealed class SavePasswordResponse
```

**Inheritance**

object ← SavePasswordResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Ignore

Creates a SavePasswordResponse that notifies the browser to ignore credentials saving.

NeverSave

Creates a SavePasswordResponse that notifies the browser that password forms on this Url URL should be blacklisted(marked as "never-saved") and SavePasswordHandler should not be invoked in the future.

Save

Creates a SavePasswordResponse that notifies the browser to save credentials in the store.

# Field Ignore

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

Creates a SavePasswordResponse that notifies the browser to ignore credentials saving.

```
public static SavePasswordResponse Ignore
```

## Returns

SavePasswordResponse

The SavePasswordResponse instance that can be used as a return value in SavePasswordHandler implementation.

# Field NeverSave

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

Creates a SavePasswordResponse that notifies the browser that password forms on this Url URL should be blacklisted(marked as "never-saved") and SavePasswordHandler should not be invoked in the future.

```
public static SavePasswordResponse NeverSave
```

## Returns

SavePasswordResponse

The SavePasswordResponse instance that can be used as a return value in SavePasswordHandler implementation.

# Field Save

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

Creates a SavePasswordResponse that notifies the browser to save credentials in the store.

```
public static SavePasswordResponse Save
```

## Returns

SavePasswordResponse

The SavePasswordResponse instance that can be used as a return value in SavePasswordHandler implementation.

# Class UpdatePasswordParameters

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

The parameters of the UpdatePasswordHandler.

```
public sealed class UpdatePasswordParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← UpdatePasswordParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Login

Gets the login for which the password is updated.

Url

Gets the URL of the resource where the form is located.

# Property Login

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

## Login

Gets the login for which the password is updated.

```
public string Login { get; }
```

## Property Value

string

The login for which the password is updated.

# Property Url

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the resource where the form is located.

```
public string Url { get; }
```

## Property Value

string

The URL of the resource where the form is located.

# Class UpdatePasswordResponse

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

A response to the UpdatePasswordHandler.

```
public sealed class UpdatePasswordResponse
```

**Inheritance**

object ← UpdatePasswordResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Ignore

Creates a UpdatePasswordResponse that notifies the browser not to update the password for this Login.

Update

Creates a UpdatePasswordResponse that notifies the browser to update the password for this Login login in the password store.

# Field Ignore

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

Creates a UpdatePasswordResponse that notifies the browser not to update the password for this Login.

```
public static UpdatePasswordResponse Ignore
```

## Returns

UpdatePasswordResponse

The UpdatePasswordResponse instance that can be used as a return value in UpdatePasswordHandler implementation.

# Field Update

Namespace: DotNetBrowser.Passwords.Handlers

Assembly: DotNetBrowser.dll

Creates a UpdatePasswordResponse that notifies the browser to update the password for this Login login in the password store.

```
public static UpdatePasswordResponse Update
```

## Returns

UpdatePasswordResponse

The UpdatePasswordResponse instance that can be used as a return value in UpdatePasswordHandler implementation.

# Namespace DotNetBrowser.Permissions

## Interfaces

IPermissions
: A service that allows managing permissions.

## Enums

PermissionType
: Represents the permission types.

# Interface IPermissions

Namespace: DotNetBrowser.Permissions

Assembly: DotNetBrowser.dll

A service that allows managing permissions.

```
public interface IPermissions : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Engine
Gets the IEngine instance associated with this object.

Profile
Gets the IProfile instance associated with this object.

RequestPermissionHandler
Gets or sets a handler that is used when a web page requests a permission, for example to enable geolocation. The permission type and the information about the web page can be obtained from the passed request object.

# Property Engine

Namespace: DotNetBrowser.Permissions

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Profile

Namespace: DotNetBrowser.Permissions

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Property RequestPermissionHandler

Namespace: DotNetBrowser.Permissions

Assembly: DotNetBrowser.dll

## RequestPermissionHandler

Gets or sets a handler that is used when a web page requests a permission, for example to enable geolocation. The permission type and the information about the web page can be obtained from the passed request object.

```
IHandler<RequestPermissionParameters, RequestPermissionResponse> RequestPermissionHandler {
get; set; }
```

## Property Value

IHandler<RequestPermissionParameters, RequestPermissionResponse>

## Remarks

Use the Grant() to grant the requested permission.

Use the Deny() to deny the requested permission.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Deny() will be used.

## Exceptions

ObjectDisposedException

The IPermissions has already been disposed.

# Enum PermissionType

Namespace: DotNetBrowser.Permissions

Assembly: DotNetBrowser.dll

Represents the permission types.

```
public enum PermissionType
```

## Fields

`AudioCapture = 7`

The permission associated with the access to the audio capture devices.

`BackgroundSync = 9`

The permission associated with the usage of Background Sync web API.

`ClipboardReadWrite = 11`

The permission associated with the usage of the asynchronous methods in the Clipboard API.

`ClipboardSanitizedWrite = 12`

The permission associated with the `Clipboard.writeText()` method in the Clipboard API.

`DurableStorage = 6`

The Persistent Storage permission (Durable Storage).

`Geolocation = 3`

The permission associated with the usage of the Geolocation API.

`LocalFonts = 13`

The permission associated with the usage of the Local Font Access API. This permission allows accessing the locally installed fonts and obtaining low-level details about them.

`Midi = 5`

The permission associated with the usage of Web MIDI API.

`MidiSysex = 1`

The permission associated with the usage of Web MIDI API.

`Notifications = 2`

The permission associated with the usage of the Notifications API.

`ProtectedMediaIdentifier = 4`

The protected media identifier permission.

`Sensors = 10`

The permission to use motion and light sensors.

`UnknownPermission = 0`

The unknown permission type.

`VideoCapture = 8`

The permission associated with the access to the video capture devices.

# Namespace DotNetBrowser.Permissions.Handlers

## Classes

RequestPermissionParameters

The parameters of the RequestPermissionHandler.

RequestPermissionResponse

A response to the RequestPermissionHandler.

# Class RequestPermissionParameters

Namespace: DotNetBrowser.Permissions.Handlers

Assembly: DotNetBrowser.dll

The parameters of the RequestPermissionHandler.

```
public sealed class RequestPermissionParameters
```

**Inheritance**

object ← RequestPermissionParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Type

Gets the permission type of the current request.

Url

Gets the URL of the web page that requests permission.

# Property Type

Namespace: DotNetBrowser.Permissions.Handlers

Assembly: DotNetBrowser.dll

## Type

Gets the permission type of the current request.

```
public PermissionType Type { get; }
```

## Property Value

PermissionType

The permission type of the current request.

# Property Url

Namespace: DotNetBrowser.Permissions.Handlers

Assembly: DotNetBrowser.dll

## Url

Gets the URL of the web page that requests permission.

```
public string Url { get; }
```

## Property Value

string

The URL of the web page that requests permission.

# Class RequestPermissionResponse

Namespace: DotNetBrowser.Permissions.Handlers

Assembly: DotNetBrowser.dll

A response to the RequestPermissionHandler.

```
public sealed class RequestPermissionResponse
```

**Inheritance**

object ← RequestPermissionResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Deny()

Creates a RequestPermissionResponse that notifies the engine that permission is denied.

Grant()

Creates a RequestPermissionResponse that notifies the engine that permission is granted.

# Method Deny

Namespace: DotNetBrowser.Permissions.Handlers

Assembly: DotNetBrowser.dll

## Deny()

Creates a RequestPermissionResponse that notifies the engine that permission is denied.

```
public static RequestPermissionResponse Deny()
```

## Returns

RequestPermissionResponse

The RequestPermissionResponse instance that can be used as a return value in RequestPermissionHandler implementation.

# Method Grant

Namespace: DotNetBrowser.Permissions.Handlers

Assembly: DotNetBrowser.dll

## Grant()

Creates a RequestPermissionResponse that notifies the engine that permission is granted.

```
public static RequestPermissionResponse Grant()
```

## Returns

RequestPermissionResponse

The RequestPermissionResponse instance that can be used as a return value in RequestPermissionHandler implementation.

# Namespace DotNetBrowser.Plugins

## Classes

Plugin
: The detailed information about the installed Chromium plugin.

## Interfaces

IPluginSettings
: The settings to configure the available Chromium DotNetBrowser.Plugins.

IPlugins
: The engine service that provides the details about the available Chromium plugins.

## Enums

PluginType
: The plugin types.

# Interface IPluginSettings

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

The settings to configure the available Chromium DotNetBrowser.Plugins.

```
public interface IPluginSettings : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

PdfViewerEnabled

Enables or disables the built-in PDF viewer.

# Property PdfViewerEnabled

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## PdfViewerEnabled

Enables or disables the built-in PDF viewer.

```
bool PdfViewerEnabled { get; set; }
```

## Property Value

bool

`true` if the PDF viewer is enabled, `false` otherwise.

## Remarks

This setting controls how DotNetBrowser handles navigations to web pages embedding PDF documents. When the PDF viewer is enabled, then the PDF document will be displayed in the PDF viewer. Otherwise, the engine will download the PDF document, like any other resource that cannot be rendered in the browser. The download can be configured via the StartDownloadHandler.

Changing this setting affects only subsequent navigations and does not affect the already loaded web pages.

By default, the PDF viewer is enabled.

## Exceptions

ObjectDisposedException

The IPluginSettings has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Interface IPlugins

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

The engine service that provides the details about the available Chromium plugins.

```
public interface IPlugins : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

AllowPluginHandler
: Gets or sets a handler that is used when the engine wants to check whether a specific plugin is allowed or not.

AvailablePlugins
: Gets the collection of the installed and available Chromium plugins.

Engine
: Gets the IEngine instance associated with this object.

Profile
: Gets the IProfile instance associated with this object.

Settings
: Gets the settings of the available Chromium plugins.

# Property AllowPluginHandler

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## AllowPluginHandler

Gets or sets a handler that is used when the engine wants to check whether a specific plugin is allowed or not.

```
IHandler<AllowPluginParameters, AllowPluginResponse> AllowPluginHandler { get; set; }
```

## Property Value

IHandler<AllowPluginParameters, AllowPluginResponse>

## Remarks

Use the Allow() method to allow the plugin.

Use the Deny() method to deny the plugin.

If an exception occurs inside the handler implementation, the default behavior will be applied - the method Allow() will be used.

**Important:** the engine will be blocked until the callback response is sent. It is not allowed to invoke any engine/browser methods in this callback.

## Exceptions

ObjectDisposedException

The IPlugins object has already been disposed.

# Property AvailablePlugins

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## AvailablePlugins

Gets the collection of the installed and available Chromium plugins.

```
IEnumerable<Plugin> AvailablePlugins { get; }
```

## Property Value

IEnumerable<Plugin>

The collection of the installed and available Chromium plugins. The collection will be empty if there are no installed and available plugins.

## Exceptions

ObjectDisposedException

The IPlugins object has already been disposed.

# Property Engine

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Profile

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Property Settings

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## Settings

Gets the settings of the available Chromium plugins.

```
IPluginSettings Settings { get; }
```

## Property Value

IPluginSettings

The IPluginSettings to configure the available Chromium plugins.

# Class Plugin

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

The detailed information about the installed Chromium plugin.

```
public class Plugin
```

**Inheritance**

object ← Plugin

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Description

Gets the plugin description.

MimeTypes

Gets the MIME types supported by this plugin.

Name

Gets the name of the plugin.

Path

Gets the plugin path representation.

PluginType

Gets the plugin type.

Version

Gets the version number of the plugin file.

# Property Description

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## Description

Gets the plugin description.

```
public string Description { get; }
```

## Property Value

string

The plugin description.

# Property MimeTypes

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## MimeTypes

Gets the MIME types supported by this plugin.

```
public IEnumerable<MimeType> MimeTypes { get; }
```

## Property Value

IEnumerable<MimeType>

A collection of the MIME types supported by this plugin.

# Property Name

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## Name

Gets the name of the plugin.

```
public string Name { get; }
```

## Property Value

string

The name of the plugin.

# Property Path

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## Path

Gets the plugin path representation.

```
public string Path { get; }
```

## Property Value

string

The plugin path representation. This can be one of the following:

- the absolute path to the plugin file on the local file system;
- the plugin extension name;
- the resource URL (for example, `chrome-extension://<IDENTIFIER>`).

# Property PluginType

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## PluginType

Gets the plugin type.

```
public PluginType PluginType { get; }
```

## Property Value

PluginType

The plugin type.

# Property Version

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

## Version

Gets the version number of the plugin file.

```
public string Version { get; }
```

## Property Value

string

The version number of the plugin file. This value may be OS-specific.

# Enum PluginType

Namespace: DotNetBrowser.Plugins

Assembly: DotNetBrowser.dll

The plugin types.

```
public enum PluginType
```

## Fields

`BrowserPlugin = 3`

A browser plugin.

`PepperInProcess = 1`

An in-process PPAPI plugin.

`PepperOutOfProcess = 2`

An out-of-process PPAPI plugin.

`Unrecognized = -1`

The plugin type is not recognized.

`Unspecified = 0`

The plugin type is not specified.

# Namespace DotNetBrowser.Plugins.Handlers

## Classes

AllowPluginParameters

The parameters for the AllowPluginHandler.

AllowPluginResponse

A response to the AllowPluginHandler.

# Class AllowPluginParameters

Namespace: DotNetBrowser.Plugins.Handlers

Assembly: DotNetBrowser.dll

The parameters for the AllowPluginHandler.

```
public sealed class AllowPluginParameters
```

**Inheritance**

object ← AllowPluginParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Plugin

Gets the information about the plugin to check.

# Property Plugin

Namespace: DotNetBrowser.Plugins.Handlers

Assembly: DotNetBrowser.dll

## Plugin

Gets the information about the plugin to check.

```
public Plugin Plugin { get; }
```

## Property Value

Plugin

The information about the plugin to check.

# Class AllowPluginResponse

Namespace: DotNetBrowser.Plugins.Handlers

Assembly: DotNetBrowser.dll

A response to the AllowPluginHandler.

```
public sealed class AllowPluginResponse
```

**Inheritance**

object ← AllowPluginResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Allow()

Creates an AllowPluginResponse that notifies the engine that the plugin is allowed to be used.

Deny()

Creates an AllowPluginResponse that notifies the engine that the plugin isn't allowed at the moment.

# Method Allow

Namespace: DotNetBrowser.Plugins.Handlers

Assembly: DotNetBrowser.dll

## Allow()

Creates an AllowPluginResponse that notifies the engine that the plugin is allowed to be used.

```
public static AllowPluginResponse Allow()
```

## Returns

AllowPluginResponse

The AllowPluginResponse instance that can be used as a return value in AllowPluginHandler implementation.

# Method Deny

Namespace: DotNetBrowser.Plugins.Handlers

Assembly: DotNetBrowser.dll

## Deny()

Creates an AllowPluginResponse that notifies the engine that the plugin isn't allowed at the moment.

```
public static AllowPluginResponse Deny()
```

## Returns

AllowPluginResponse

The AllowPluginResponse instance that can be used as a return value in AllowPluginHandler implementation.

# Namespace DotNetBrowser.Print

## Classes

Capabilities
: The capabilities of a printer.

PageMargins
: The page margins used for printing.

PageRange
: The page range to be printed.

PaperSize
: The paper size used for printing.

PdfPrinter
: A printer that allows you to print to PDF.

PdfPrinter<TPrintSettings>
: A printer that allows you to print to PDF.

Printer<TPrintSettings>
: The base class for the printers.

Scaling
: The scaling used for printing.

SystemPrinter
: A local or network printer available in the system.

SystemPrinter<TPrintSettings>
: A local or network printer available in the system.

## Interfaces

IPrintJob<TPrintSettings>
: A printing operation that is currently in-progress.

IPrinters<TPrintSettings, TPdfPrintSettings>
: The collection of the available printers.

PdfPrinter.IHtmlSettings
: The print settings available when printing HTML content on the PDF printer.

PdfPrinter.IPdfSettings
: The print settings available when printing PDF content on the PDF printer.

PdfPrinter.ISettings<T>
: The print settings available when printing HTML or PDF content on the PDF printer.

SystemPrinter.IHtmlSettings
: Print settings available when printing HTML content on a physical printer.

SystemPrinter.IPdfSettings
: Print settings available when printing PDF content on a physical printer.

SystemPrinter.ISettings<T>
: Print settings available when printing HTML or PDF content on a physical printer.

## Enums

ColorModel
: The color models used for printing.

DuplexMode
: The duplex modes used for printing.

Fit
: The content fit used for printing.

Orientation
: The page orientation values used for printing.

PagesPerSheet
: The pages per sheet values that Chromium supports.

PaperSize.Unit
: The measurement unit of the paper size.

# Class Capabilities

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The capabilities of a printer.

```
public sealed class Capabilities
```

**Inheritance**

object ← Capabilities

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

CanCollate

Indicates whether the printer supports collate printing.

ColorModels

Gets the list of the ColorModel values supported by the printer.

DuplexModes

Gets the collection of the DuplexMode values supported by the printer.

MaxCopies

Gets the maximum number of copies that the printer can make.

PaperSizes

Gets the collection of the PaperSize values supported by the printer.

# Property CanCollate

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## CanCollate

Indicates whether the printer supports collate printing.

```
public bool CanCollate { get; }
```

## Property Value

bool

`true` if the printer can collate, `false` otherwise.

# Property ColorModels

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## ColorModels

Gets the list of the ColorModel values supported by the printer.

```
public IReadOnlyList<ColorModel> ColorModels { get; }
```

## Property Value

IReadOnlyList<ColorModel>

The list of the ColorModel values supported by the printer.

# Property DuplexModes

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## DuplexModes

Gets the collection of the DuplexMode values supported by the printer.

```
public IReadOnlyCollection<DuplexMode> DuplexModes { get; }
```

## Property Value

IReadOnlyCollection<DuplexMode>

The collection of the DuplexMode values supported by the printer.

# Property MaxCopies

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## MaxCopies

Gets the maximum number of copies that the printer can make.

```
public int MaxCopies { get; }
```

## Property Value

int

The maximum number of copies that the printer can make.

# Property PaperSizes

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## PaperSizes

Gets the collection of the PaperSize values supported by the printer.

```
public IReadOnlyCollection<PaperSize> PaperSizes { get; }
```

## Property Value

IReadOnlyCollection<PaperSize>

The collection of the PaperSize values supported by the printer.

# Enum ColorModel

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The color models used for printing.

```
public enum ColorModel
```

## Fields

`Black = 7`

BLACK color model.

`BrotherBrScript3Black = 24`

BROTHER_BR_SCRIPT_3_BLACK color model. Used in Brother BRScript3 color printer PPDs.

`BrotherBrScript3Color = 23`

BROTHER_BR_SCRIPT_3_COLOR color model. Used in Brother BRScript3 color printer PPDs.

`BrotherCupsColor = 21`

BROTHER_CUPS_COLOR color model. Used in Brother color laser printer PPDs.

`BrotherCupsMono = 22`

BROTHER_CUPS_MONO color model. Used in Brother color laser printer PPDs.

`Cmy = 4`

CMY color model.

`CmyPlusK = 6`

CMYPlusK represents CMY+K color model.

`Cmyk = 3`

CMYK color model.

`Color = 2`

COLOR color model.

`ColorModeColor = 12`

COLOR_MODE_COLOR color model. Used in Samsung printer PPDs.

`ColorModeMonochrome = 13`

COLOR_MODE_MONOCHROME color model. Used in Samsung printer PPDs.

`Gray = 1`

GRAY color model.

`Grayscale = 8`

GRAYSCALE color model.

`HpColorBlack = 15`

HP_COLOR_BLACK color model. Used in HP color printer PPDs.

`HpColorColor = 14`

HP_COLOR_COLOR color model. Used in HP color printer PPDs.

`Kcmy = 5`

KCMY color model.

`PrintoutModeNormal = 16`

PRINTOUT_MODE_NORMAL color model. Used in Foomatic PPDs.

`PrintoutModeNormalGray = 17`

PRINTOUT_MODE_NORMAL_GRAY color model. Used in Foomatic PPDs.

`ProcessColorModelCmyk = 18`

PROCESS_COLOR_MODEL_CMYK color model. Used in Canon printer PPDs.

`ProcessColorModelGreyscale = 19`

PROCESS_COLOR_MODEL_GREYSCALE color model. Used in Canon printer PPDs.

`ProcessColorModelRgb = 20`

PROCESS_COLOR_MODEL_RGB color model. Used in Canon printer PPDs

`Rgb = 9`

RGB color model.

`Rgb16 = 10`

RGB16 color model.

`Rgba = 11`

RGBA color model.

`Unspecified = 0`

Reserved value.

# Enum DuplexMode

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The duplex modes used for printing.

```
public enum DuplexMode
```

## Fields

`LongEdge = 2`

Two-sided printing, flipping on the long edge.

`ShortEdge = 3`

Two-sided printing, flipping on the short edge.

`Simplex = 1`

One-sided printing.

`Unspecified = 0`

Reserved value.

# Enum Fit

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The content fit used for printing.

```
public enum Fit
```

## Fields

`ToPage = 1`

Fit the content of the page.

`ToPaper = 2`

Fit the content of the paper.

`Unspecified = 0`

Reserved value.

# Interface IPrintJob<TPrintSettings>

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

A printing operation that is currently in-progress.

```
public interface IPrintJob<TPrintSettings> : IAutoDisposable where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The type of print settings that can be applied to this job.

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Settings

Gets the configurable settings of this print job.

## Events

PageCountUpdated

Occurs when the total number of pages to be printed is updated.

PrintCompleted

Occurs when the printing operation is completed.

# Property Settings

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Settings

Gets the configurable settings of this print job.

```
TPrintSettings Settings { get; }
```

## Property Value

TPrintSettings

The configurable settings of this print job.

# Event PageCountUpdated

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Occurs when the total number of pages to be printed is updated.

```
event EventHandler<PageCountUpdatedEventArgs<TPrintSettings>> PageCountUpdated
```

## Returns

EventHandler<PageCountUpdatedEventArgs<TPrintSettings>>

Occurs when the total number of pages to be printed is updated.

## Remarks

The page count is updated each time when the Apply() method is called.

# Event PrintCompleted

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Occurs when the printing operation is completed.

```
event EventHandler<PrintCompletedEventArgs<TPrintSettings>> PrintCompleted
```

## Returns

EventHandler<PrintCompletedEventArgs<TPrintSettings>>

Occurs when the printing operation is completed.

# Interface IPrinters<TPrintSettings, TPdfPrintSettings>

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The collection of the available printers.

```
public interface IPrinters<TPrintSettings, TPdfPrintSettings> :
IReadOnlyList<SystemPrinter<TPrintSettings>>,
IReadOnlyCollection<SystemPrinter<TPrintSettings>>,
IEnumerable<SystemPrinter<TPrintSettings>>, IEnumerable where TPrintSettings : class,
SystemPrinter.ISettings<TPrintSettings> where TPdfPrintSettings : class,
PdfPrinter.ISettings<TPdfPrintSettings>
```

## Type Parameters

`TPrintSettings`

The print settings type available for these printers.

`TPdfPrintSettings`

The PDF print settings type available for PDF printer.

**Inherited Members**

IReadOnlyList<SystemPrinter<TPrintSettings>>.this[int] ,
IReadOnlyCollection<SystemPrinter<TPrintSettings>>.Count ,
IEnumerable<SystemPrinter<TPrintSettings>>.GetEnumerator()

# Properties

Default

Gets the default system printer.

Pdf

Gets the PdfPrinter instance that allows you to print to PDF.

# Property Default

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Default

Gets the default system printer.

```
SystemPrinter<TPrintSettings> Default { get; }
```

## Property Value

SystemPrinter<TPrintSettings>

The default system printer or `null` if there is no default printer configured.

# Property Pdf

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Pdf

Gets the PdfPrinter instance that allows you to print to PDF.

```
PdfPrinter<TPdfPrintSettings> Pdf { get; }
```

## Property Value

PdfPrinter<TPdfPrintSettings>

The PdfPrinter instance that allows you to print to PDF.

# Enum Orientation

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The page orientation values used for printing.

```
public enum Orientation
```

## Fields

`Landscape = 1`

The landscape orientation.

`Portrait = 2`

The portrait orientation.

`Unrecognized = -1`

The orientation is not recognized.

`Unspecified = 0`

Reserved value.

# Class PageMargins

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The page margins used for printing.

```
public sealed class PageMargins
```

**Inheritance**

object ← PageMargins

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

PageMargins(int, int, int, int)

Creates custom margins for the web page in points. One point equals 1/72 of an inch.

## Fields

Default

The default margins for the page.

None

The empty margins for the page.

## Properties

Bottom

Gets the margin on the bottom side of the sheet.

Left

Gets the margin on the left side of the sheet.

Right
Gets the margin on the right side of the sheet.

Top
Gets the margin on the top side of the sheet.

## Methods

Equals(object)

GetHashCode()

# Constructor PageMargins

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## PageMargins(int, int, int, int)

Creates custom margins for the web page in points. One point equals 1/72 of an inch.

```
public PageMargins(int left, int right, int top, int bottom)
```

## Parameters

`left` int

The margin on the left side of the sheet. Cannot be negative.

`right` int

The margin on the right side of the sheet. Cannot be negative.

`top` int

The margin on the top side of the sheet. Cannot be negative.

`bottom` int

The margin on the bottom side of the sheet. Cannot be negative.

## Exceptions

ArgumentException

Any of the parameters is negative.

# Field Default

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The default margins for the page.

```
public static readonly PageMargins Default
```

## Returns

PageMargins

The default margins for the page.

# Field None

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The empty margins for the page.

```
public static readonly PageMargins None
```

## Returns

PageMargins

The empty margins for the page.

# Property Bottom

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Bottom

Gets the margin on the bottom side of the sheet.

```
public int Bottom { get; }
```

## Property Value

int

The margin on the bottom side of the sheet.

# Property Left

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Left

Gets the margin on the left side of the sheet.

```
public int Left { get; }
```

## Property Value

int

The margin on the left side of the sheet.

# Property Right

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Right

Gets the margin on the right side of the sheet.

```
public int Right { get; }
```

## Property Value

int

The margin on the right side of the sheet.

# Property Top

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Top

Gets the margin on the top side of the sheet.

```
public int Top { get; }
```

## Property Value

int

The margin on the top side of the sheet.

# Method Equals

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class PageRange

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The page range to be printed.

```
public sealed class PageRange
```

**Inheritance**

object ← PageRange

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

PageRange(int, int)

Creates a new page range instance.

## Properties

From

Gets the number of the first page within the range.

To

Gets the number of the last page within the range.

## Methods

Equals(object)

GetHashCode()

# Constructor PageRange

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## PageRange(int, int)

Creates a new page range instance.

```
public PageRange(int from, int to)
```

## Parameters

`from` int

The 1-based number of the first page within the range.

`to` int

The 1-based number of the last page within the range.

## Exceptions

ArgumentException

The `from` is greater than `to`.

ArgumentException

The `from` is less than or equal to zero.

# Property From

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## From

Gets the number of the first page within the range.

```
public int From { get; }
```

## Property Value

int

The number of the first page within the range.

# Property To

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## To

Gets the number of the last page within the range.

```
public int To { get; }
```

## Property Value

int

The number of the last page within the range.

# Method Equals

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Enum PagesPerSheet

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The pages per sheet values that Chromium supports.

```
public enum PagesPerSheet
```

## Fields

`Four = 4`

Four pages per sheet.

`Nine = 9`

Nine pages per sheet.

`One = 1`

One page per sheet.

`Six = 6`

Six pages per sheet.

`Sixteen = 16`

Sixteen pages per sheet.

`Two = 2`

Two pages per sheet.

`Unrecognized = -1`

The pages per sheet value is not recognized.

`Unspecified = 0`

Reserved value.

# Class PaperSize

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The paper size used for printing.

```
public sealed class PaperSize
```

**Inheritance**

object ← PaperSize

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

## Fields

Iso2A0
ISO standard sheet media size "2A0".

IsoA0
ISO standard sheet media size "A0".

IsoA0X3
ISO standard sheet media size "A0x3".

IsoA1
ISO standard sheet media size "A1".

IsoA10
ISO standard sheet media size "A10".

IsoA1X3
ISO standard sheet media size "A1x3".

IsoA1X4
ISO standard sheet media size "A1x4".

IsoA2
ISO standard sheet media size "A2".

IsoA2X3
ISO standard sheet media size "A2x3".

IsoA2X4
ISO standard sheet media size "A2x4".

IsoA2X5
ISO standard sheet media size "A2x5".

IsoA3
ISO standard sheet media size "A3".

IsoA3Extra
ISO standard sheet media size "A3 Extra".

IsoA3X3
ISO standard sheet media size "A3x3".

IsoA3X4
ISO standard sheet media size "A3x4".

IsoA3X5
ISO standard sheet media size "A3x5".

IsoA3X6
ISO standard sheet media size "A3x6".

IsoA3X7
ISO standard sheet media size "A3x7".

IsoA4
ISO standard sheet media size "A4".

IsoA4Extra
ISO standard sheet media size "A4 Extra".

IsoA4Tab
ISO standard sheet media size "A4 Tab".

IsoA4X3
ISO standard sheet media size "A4x3".

IsoA4X4
ISO standard sheet media size "A4x4".

IsoA4X5
ISO standard sheet media size "A4x5".

IsoA4X6
ISO standard sheet media size "A4x6".

IsoA4X7
ISO standard sheet media size "A4x7".

IsoA4X8
ISO standard sheet media size "A4x8".

IsoA4X9
ISO standard sheet media size "A4x9".

IsoA5
ISO standard sheet media size "A5".

IsoA5Extra
ISO standard sheet media size "A5 Extra".

IsoA6
ISO standard sheet media size "A6".

IsoA7
ISO standard sheet media size "A7".

IsoA8
ISO standard sheet media size "A8".

IsoA9
ISO standard sheet media size "A9".

IsoB0
ISO standard sheet media size "B0".

IsoB1
ISO standard sheet media size "B1".

IsoB10
ISO standard sheet media size "B10".

IsoB2
ISO standard sheet media size "B2".

IsoB3
ISO standard sheet media size "B3".

IsoB4
ISO standard sheet media size "B4".

IsoB5
ISO standard sheet media size "B5".

IsoB5Extra
ISO standard sheet media size "B5 Extra".

IsoB6
ISO standard sheet media size "B6".

IsoB6C4
ISO standard sheet media size "B6C4".

IsoB7
ISO standard sheet media size "B7".

IsoB8
ISO standard sheet media size "B8".

IsoB9
ISO standard sheet media size "B9".

IsoC0
ISO standard sheet media size "C0".

IsoC1
ISO standard sheet media size "C1".

IsoC10
ISO standard sheet media size "C10".

IsoC2
ISO standard sheet media size "C2".

IsoC3
ISO standard sheet media size "C3".

IsoC4
ISO standard sheet media size "C4".

IsoC5
ISO standard sheet media size "C5".

IsoC6
ISO standard sheet media size "C6".

IsoC6C5
ISO standard sheet media size "C6C5".

IsoC7
ISO standard sheet media size "C7".

IsoC7C6
ISO standard sheet media size "C7C6".

IsoC8
ISO standard sheet media size "C8".

IsoC9
ISO standard sheet media size "C9".

IsoDl
ISO standard sheet media size "DL".

IsoRa0
ISO standard sheet media size "RA0".

IsoRa1
ISO standard sheet media size "RA1".

IsoRa2
ISO standard sheet media size "RA2".

IsoSra0
ISO standard sheet media size "SRA0".

IsoSra1
ISO standard sheet media size "SRA1".

IsoSra2
ISO standard sheet media size "SRA2".

JisB0
Japanese standard sheet media size "B0 (JIS)".

JisB1
Japanese standard sheet media size "B1 (JIS)".

JisB10
Japanese standard sheet media size "B10 (JIS)".

JisB2
Japanese standard sheet media size "B2 (JIS)".

JisB3
Japanese standard sheet media size "B3 (JIS)".

JisB4
Japanese standard sheet media size "B4 (JIS)".

JisB5
Japanese standard sheet media size "B5 (JIS)".

JisB6
Japanese standard sheet media size "B6 (JIS)".

JisB7
Japanese standard sheet media size "B7 (JIS)".

JisB8
Japanese standard sheet media size "B8 (JIS)".

JisB9
Japanese standard sheet media size "B9 (JIS)".

JisExec
Japanese standard sheet media size "Executive (JIS)".

JpnChou2
Japanese standard sheet media size "Choukei 2".

JpnChou3
Japanese standard sheet media size "Choukei 3".

JpnChou4
Japanese standard sheet media size "Choukei 4".

JpnHagaki
Japanese standard sheet media size "Hagaki".

JpnKahu
Japanese standard sheet media size "Kahu Envelope".

JpnKaku2
Japanese standard sheet media size "Kaku 2".

JpnOufuku
Japanese standard sheet media size "Oufuku Hagaki".

JpnYou4
Japanese standard sheet media size "You 4".

Na10X11
North American standard sheet media size "10x11".

Na10X13
North American standard sheet media size "10x13".

Na10X14
North American standard sheet media size "10x14".

Na10X15
North American standard sheet media size "10x15".

Na11X12
North American standard sheet media size "11x12".

Na11X15
North American standard sheet media size "11x15".

Na12X19
North American standard sheet media size "12x19".

Na5X7
North American standard sheet media size "5x7".

Na6X9
North American standard sheet media size "6x9".

Na7X9
North American standard sheet media size "7x9".

Na9X11
North American standard sheet media size "9x11".

NaA2
North American standard sheet media size "A2".

NaArchA
North American standard sheet media size "Arch A".

NaArchB
North American standard sheet media size "Arch B".

NaArchC
North American standard sheet media size "Arch C".

NaArchD
North American standard sheet media size "Arch D".

NaArchE
North American standard sheet media size "Arch E".

NaAsmeF
North American standard sheet media size "ASME F".

NaBPlus
North American standard sheet media size "B-plus".

NaC
North American standard sheet media size "C".

NaC5
North American standard sheet media size "C5".

NaD
North American standard sheet media size "D".

NaE
North American standard sheet media size "E".

NaEdp
North American standard sheet media size "EDP".

NaEurEdp
North American standard sheet media size "European EDP".

NaExecutive
North American standard sheet media size "Executive".

NaF

North American standard sheet media size "F".

NaFanfoldEur

North American standard sheet media size "FanFold European".

NaFanfoldUs

North American standard sheet media size "FanFold US".

NaFoolscap

North American standard sheet media size "FanFold German Legal".

NaGovtLegal

North American standard sheet media size "Government Legal".

NaGovtLetter

North American standard sheet media size "Government Letter".

NaIndex3X5

North American standard sheet media size "Index 3x5".

NaIndex4X6

North American standard sheet media size "Index 4x6".

NaIndex4X6Ext

North American standard sheet media size "Index 4x6 ext".

NaIndex5X8

North American standard sheet media size "Index 5x8".

NaInvoice

North American standard sheet media size "Invoice".

NaLedger

North American standard sheet media size "Tabloid".

NaLegal

North American standard sheet media size "Legal".

NaLegalExtra

North American standard sheet media size "Legal extra".

NaLetter

North American standard sheet media size "Letter".

NaLetterExtra
North American standard sheet media size "Letter extra".

NaLetterPlus
North American standard sheet media size "Letter plus".

NaMonarch
North American standard sheet media size "Monarch".

NaNumber10
North American standard sheet media size "Envelope #10".

NaNumber11
North American standard sheet media size "Envelope #11".

NaNumber12
North American standard sheet media size "Envelope #12".

NaNumber14
North American standard sheet media size "Envelope #14".

NaNumber9
North American standard sheet media size "Envelope #9".

NaPersonal
North American standard sheet media size "Personal".

NaQuarto
North American standard sheet media size "Quarto".

NaSuperA
North American standard sheet media size "Super A".

NaSuperB
North American standard sheet media size "Super B".

NaWideFormat
North American standard sheet media size "Wide format".

OmDaiPaKai
Chinese standard sheet media size "Dai-pa-kai".

OmFolio
Other metric standard sheet media size "Folio".

OmFolioSp
Other metric standard sheet media size "Folio SP".

OmInvite
Other metric standard sheet media size "Invite Envelope".

OmItalian
Other metric standard sheet media size "Italian Envelope".

OmJuuroKuKai
Chinese standard sheet media size "Juuro-ku-kai".

OmLargePhoto
Other metric standard sheet media size "Large photo".

OmPaKai
Chinese standard sheet media size "Pa-kai".

OmPostfix
Other metric standard sheet media size "Postfix Envelope".

OmSmallPhoto
Other metric standard sheet media size "Small photo".

Prc1
Chinese standard sheet media size "prc1 Envelope".

Prc10
Chinese standard sheet media size "prc10 Envelope".

Prc16K
Chinese standard sheet media size "prc 16k".

Prc2
Chinese standard sheet media size "prc2 Envelope".

Prc3
Chinese standard sheet media size "prc3 Envelope".

Prc32K
Chinese standard sheet media size "prc 32k".

Prc4
Chinese standard sheet media size "prc4 Envelope".

Prc5
Chinese standard sheet media size "prc5 Envelope".

Prc6
Chinese standard sheet media size "prc6 Envelope".

Prc7
Chinese standard sheet media size "prc7 Envelope".

Prc8
Chinese standard sheet media size "prc8 Envelope".

Roc16K
Chinese standard sheet media size "ROC 16K".

Roc8K
Chinese standard sheet media size "ROC 8K".

## Properties

Height
Gets the paper height in microns.

Name
Gets the paper size name.

Width
Gets the paper width in microns.

## Methods

Equals(object)

GetHashCode()

Of(float, float, Unit)
Creates a new paper size setting.

Of(float, float, Unit, string)
Creates a new paper size setting.

ToString()

# Field Iso2A0

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "2A0".

```
public static readonly PaperSize Iso2A0
```

## Returns

PaperSize

ISO standard sheet media size "2A0".

# Field IsoA0

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A0".

```
public static readonly PaperSize IsoA0
```

## Returns

PaperSize

ISO standard sheet media size "A0".

# Field IsoA0X3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A0x3".

```
public static readonly PaperSize IsoA0X3
```

## Returns

PaperSize

ISO standard sheet media size "A0x3".

# Field IsoA1

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A1".

```
public static readonly PaperSize IsoA1
```

## Returns

PaperSize

ISO standard sheet media size "A1".

# Field IsoA10

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A10".

```
public static readonly PaperSize IsoA10
```

## Returns

PaperSize

ISO standard sheet media size "A10".

# Field IsoA1X3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A1x3".

```
public static readonly PaperSize IsoA1X3
```

## Returns

PaperSize

ISO standard sheet media size "A1x3".

# Field IsoA1X4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A1x4".

```
public static readonly PaperSize IsoA1X4
```

## Returns

PaperSize

ISO standard sheet media size "A1x4".

# Field IsoA2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A2".

```
public static readonly PaperSize IsoA2
```

## Returns

PaperSize

ISO standard sheet media size "A2".

# Field IsoA2X3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A2x3".

```
public static readonly PaperSize IsoA2X3
```

## Returns

PaperSize

ISO standard sheet media size "A2x3".

# Field IsoA2X4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A2x4".

```
public static readonly PaperSize IsoA2X4
```

## Returns

PaperSize

ISO standard sheet media size "A2x4".

# Field IsoA2X5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A2x5".

```
public static readonly PaperSize IsoA2X5
```

## Returns

PaperSize

ISO standard sheet media size "A2x5".

# Field IsoA3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A3".

```
public static readonly PaperSize IsoA3
```

## Returns

PaperSize

ISO standard sheet media size "A3".

# Field IsoA3Extra

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A3 Extra".

```
public static readonly PaperSize IsoA3Extra
```

## Returns

PaperSize

ISO standard sheet media size "A3 Extra".

# Field IsoA3X3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A3x3".

```
public static readonly PaperSize IsoA3X3
```

## Returns

PaperSize

ISO standard sheet media size "A3x3".

# Field IsoA3X4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A3x4".

```
public static readonly PaperSize IsoA3X4
```

## Returns

PaperSize

ISO standard sheet media size "A3x4".

# Field IsoA3X5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A3x5".

```
public static readonly PaperSize IsoA3X5
```

## Returns

PaperSize

ISO standard sheet media size "A3x5".

# Field IsoA3X6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A3x6".

```
public static readonly PaperSize IsoA3X6
```

## Returns

PaperSize

ISO standard sheet media size "A3x6".

# Field IsoA3X7

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A3x7".

```
public static readonly PaperSize IsoA3X7
```

## Returns

PaperSize

ISO standard sheet media size "A3x7".

# Field IsoA4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4".

```
public static readonly PaperSize IsoA4
```

## Returns

PaperSize

ISO standard sheet media size "A4".

# Field IsoA4Extra

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4 Extra".

```
public static readonly PaperSize IsoA4Extra
```

## Returns

PaperSize

ISO standard sheet media size "A4 Extra".

# Field IsoA4Tab

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4 Tab".

```
public static readonly PaperSize IsoA4Tab
```

## Returns

PaperSize

ISO standard sheet media size "A4 Tab".

# Field IsoA4X3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4x3".

```
public static readonly PaperSize IsoA4X3
```

## Returns

PaperSize

ISO standard sheet media size "A4x3".

# Field IsoA4X4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4x4".

```
public static readonly PaperSize IsoA4X4
```

## Returns

PaperSize

ISO standard sheet media size "A4x4".

# Field IsoA4X5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4x5".

```
public static readonly PaperSize IsoA4X5
```

## Returns

PaperSize

ISO standard sheet media size "A4x5".

# Field IsoA4X6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4x6".

```
public static readonly PaperSize IsoA4X6
```

## Returns

PaperSize

ISO standard sheet media size "A4x6".

# Field IsoA4X7

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4x7".

```
public static readonly PaperSize IsoA4X7
```

## Returns

PaperSize

ISO standard sheet media size "A4x7".

# Field IsoA4X8

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4x8".

```
public static readonly PaperSize IsoA4X8
```

## Returns

PaperSize

ISO standard sheet media size "A4x8".

# Field IsoA4X9

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A4x9".

```
public static readonly PaperSize IsoA4X9
```

## Returns

PaperSize

ISO standard sheet media size "A4x9".

# Field IsoA5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A5".

```
public static readonly PaperSize IsoA5
```

## Returns

PaperSize

ISO standard sheet media size "A5".

# Field IsoA5Extra

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A5 Extra".

```
public static readonly PaperSize IsoA5Extra
```

## Returns

PaperSize

ISO standard sheet media size "A5 Extra".

# Field IsoA6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A6".

```
public static readonly PaperSize IsoA6
```

## Returns

PaperSize

ISO standard sheet media size "A6".

# Field IsoA7

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A7".

```
public static readonly PaperSize IsoA7
```

## Returns

PaperSize

ISO standard sheet media size "A7".

# Field IsoA8

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A8".

```
public static readonly PaperSize IsoA8
```

## Returns

PaperSize

ISO standard sheet media size "A8".

# Field IsoA9

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "A9".

```
public static readonly PaperSize IsoA9
```

## Returns

PaperSize

ISO standard sheet media size "A9".

# Field IsoB0

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B0".

```
public static readonly PaperSize IsoB0
```

## Returns

PaperSize

ISO standard sheet media size "B0".

# Field IsoB1

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B1".

```
public static readonly PaperSize IsoB1
```

## Returns

PaperSize

ISO standard sheet media size "B1".

# Field IsoB10

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B10".

```
public static readonly PaperSize IsoB10
```

## Returns

PaperSize

ISO standard sheet media size "B10".

# Field IsoB2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B2".

```
public static readonly PaperSize IsoB2
```

## Returns

PaperSize

ISO standard sheet media size "B2".

# Field IsoB3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B3".

```
public static readonly PaperSize IsoB3
```

## Returns

PaperSize

ISO standard sheet media size "B3".

# Field IsoB4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B4".

```
public static readonly PaperSize IsoB4
```

## Returns

PaperSize

ISO standard sheet media size "B4".

# Field IsoB5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B5".

```
public static readonly PaperSize IsoB5
```

## Returns

PaperSize

ISO standard sheet media size "B5".

# Field IsoB5Extra

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B5 Extra".

```
public static readonly PaperSize IsoB5Extra
```

## Returns

PaperSize

ISO standard sheet media size "B5 Extra".

# Field IsoB6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B6".

```
public static readonly PaperSize IsoB6
```

## Returns

PaperSize

ISO standard sheet media size "B6".

# Field IsoB6C4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B6C4".

```
public static readonly PaperSize IsoB6C4
```

## Returns

PaperSize

ISO standard sheet media size "B6C4".

# Field IsoB7

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B7".

```
public static readonly PaperSize IsoB7
```

## Returns

PaperSize

ISO standard sheet media size "B7".

# Field IsoB8

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B8".

```
public static readonly PaperSize IsoB8
```

## Returns

PaperSize

ISO standard sheet media size "B8".

# Field IsoB9

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "B9".

```
public static readonly PaperSize IsoB9
```

## Returns

PaperSize

ISO standard sheet media size "B9".

# Field IsoC0

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C0".

```
public static readonly PaperSize IsoC0
```

## Returns

PaperSize

ISO standard sheet media size "C0".

# Field IsoC1

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C1".

```
public static readonly PaperSize IsoC1
```

## Returns

PaperSize

ISO standard sheet media size "C1".

# Field IsoC10

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C10".

```
public static readonly PaperSize IsoC10
```

## Returns

PaperSize

ISO standard sheet media size "C10".

# Field IsoC2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C2".

```
public static readonly PaperSize IsoC2
```

## Returns

PaperSize

ISO standard sheet media size "C2".

# Field IsoC3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C3".

```
public static readonly PaperSize IsoC3
```

## Returns

PaperSize

ISO standard sheet media size "C3".

# Field IsoC4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C4".

```
public static readonly PaperSize IsoC4
```

## Returns

PaperSize

ISO standard sheet media size "C4".

# Field IsoC5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C5".

```
public static readonly PaperSize IsoC5
```

## Returns

PaperSize

ISO standard sheet media size "C5".

# Field IsoC6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C6".

```
public static readonly PaperSize IsoC6
```

## Returns

PaperSize

ISO standard sheet media size "C6".

# Field IsoC6C5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C6C5".

```
public static readonly PaperSize IsoC6C5
```

## Returns

PaperSize

ISO standard sheet media size "C6C5".

# Field IsoC7

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C7".

```
public static readonly PaperSize IsoC7
```

## Returns

PaperSize

ISO standard sheet media size "C7".

# Field IsoC7C6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C7C6".

```
public static readonly PaperSize IsoC7C6
```

## Returns

PaperSize

ISO standard sheet media size "C7C6".

# Field IsoC8

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C8".

```
public static readonly PaperSize IsoC8
```

## Returns

PaperSize

ISO standard sheet media size "C8".

# Field IsoC9

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "C9".

```
public static readonly PaperSize IsoC9
```

## Returns

PaperSize

ISO standard sheet media size "C9".

# Field IsoDl

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "DL".

```
public static readonly PaperSize IsoDl
```

## Returns

PaperSize

ISO standard sheet media size "DL".

# Field IsoRa0

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "RA0".

```
public static readonly PaperSize IsoRa0
```

## Returns

PaperSize

ISO standard sheet media size "RA0".

# Field IsoRa1

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "RA1".

```
public static readonly PaperSize IsoRa1
```

## Returns

PaperSize

ISO standard sheet media size "RA1".

# Field IsoRa2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "RA2".

```
public static readonly PaperSize IsoRa2
```

## Returns

PaperSize

ISO standard sheet media size "RA2".

# Field IsoSra0

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "SRA0".

```
public static readonly PaperSize IsoSra0
```

## Returns

PaperSize

ISO standard sheet media size "SRA0".

# Field IsoSra1

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "SRA1".

```
public static readonly PaperSize IsoSra1
```

## Returns

PaperSize

ISO standard sheet media size "SRA1".

# Field IsoSra2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

ISO standard sheet media size "SRA2".

```
public static readonly PaperSize IsoSra2
```

## Returns

PaperSize

ISO standard sheet media size "SRA2".

# Field JisB0

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B0 (JIS)".

```
public static readonly PaperSize JisB0
```

## Returns

PaperSize

Japanese standard sheet media size "B0 (JIS)".

# Field JisB1

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B1 (JIS)".

```
public static readonly PaperSize JisB1
```

## Returns

PaperSize

Japanese standard sheet media size "B1 (JIS)".

# Field JisB10

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B10 (JIS)".

```
public static readonly PaperSize JisB10
```

## Returns

PaperSize

Japanese standard sheet media size "B10 (JIS)".

# Field JisB2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B2 (JIS)".

```
public static readonly PaperSize JisB2
```

## Returns

PaperSize

Japanese standard sheet media size "B2 (JIS)".

# Field JisB3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B3 (JIS)".

```
public static readonly PaperSize JisB3
```

## Returns

PaperSize

Japanese standard sheet media size "B3 (JIS)".

# Field JisB4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B4 (JIS)".

```
public static readonly PaperSize JisB4
```

## Returns

PaperSize

Japanese standard sheet media size "B4 (JIS)".

# Field JisB5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B5 (JIS)".

```
public static readonly PaperSize JisB5
```

## Returns

PaperSize

Japanese standard sheet media size "B5 (JIS)".

# Field JisB6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B6 (JIS)".

```
public static readonly PaperSize JisB6
```

## Returns

PaperSize

Japanese standard sheet media size "B6 (JIS)".

# Field JisB7

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B7 (JIS)".

```
public static readonly PaperSize JisB7
```

## Returns

PaperSize

Japanese standard sheet media size "B7 (JIS)".

# Field JisB8

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B8 (JIS)".

```
public static readonly PaperSize JisB8
```

## Returns

PaperSize

Japanese standard sheet media size "B8 (JIS)".

# Field JisB9

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "B9 (JIS)".

```
public static readonly PaperSize JisB9
```

## Returns

PaperSize

Japanese standard sheet media size "B9 (JIS)".

# Field JisExec

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "Executive (JIS)".

```
public static readonly PaperSize JisExec
```

## Returns

PaperSize

Japanese standard sheet media size "Executive (JIS)".

# Field JpnChou2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "Choukei 2".

```
public static readonly PaperSize JpnChou2
```

## Returns

PaperSize

Japanese standard sheet media size "Choukei 2".

# Field JpnChou3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "Choukei 3".

```
public static readonly PaperSize JpnChou3
```

## Returns

PaperSize

Japanese standard sheet media size "Choukei 3".

# Field JpnChou4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "Choukei 4".

```
public static readonly PaperSize JpnChou4
```

## Returns

PaperSize

Japanese standard sheet media size "Choukei 4".

# Field JpnHagaki

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "Hagaki".

```
public static readonly PaperSize JpnHagaki
```

## Returns

PaperSize

Japanese standard sheet media size "Hagaki".

# Field JpnKahu

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "Kahu Envelope".

```
public static readonly PaperSize JpnKahu
```

## Returns

PaperSize

Japanese standard sheet media size "Kahu Envelope".

# Field JpnKaku2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "Kaku 2".

```
public static readonly PaperSize JpnKaku2
```

## Returns

PaperSize

Japanese standard sheet media size "Kaku 2".

# Field JpnOufuku

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "Oufuku Hagaki".

```
public static readonly PaperSize JpnOufuku
```

## Returns

PaperSize

Japanese standard sheet media size "Oufuku Hagaki".

# Field JpnYou4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Japanese standard sheet media size "You 4".

```
public static readonly PaperSize JpnYou4
```

## Returns

PaperSize

Japanese standard sheet media size "You 4".

# Field Na10X11

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "10x11".

```
public static readonly PaperSize Na10X11
```

## Returns

PaperSize

North American standard sheet media size "10x11".

# Field Na10X13

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "10x13".

```
public static readonly PaperSize Na10X13
```

## Returns

PaperSize

North American standard sheet media size "10x13".

# Field Na10X14

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "10x14".

```
public static readonly PaperSize Na10X14
```

## Returns

PaperSize

North American standard sheet media size "10x14".

# Field Na10X15

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "10x15".

```
public static readonly PaperSize Na10X15
```

## Returns

PaperSize

North American standard sheet media size "10x15".

# Field Na11X12

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "11x12".

```
public static readonly PaperSize Na11X12
```

## Returns

PaperSize

North American standard sheet media size "11x12".

# Field Na11X15

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "11x15".

```
public static readonly PaperSize Na11X15
```

## Returns

PaperSize

North American standard sheet media size "11x15".

# Field Na12X19

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "12x19".

```
public static readonly PaperSize Na12X19
```

## Returns

PaperSize

North American standard sheet media size "12x19".

# Field Na5X7

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "5x7".

```
public static readonly PaperSize Na5X7
```

## Returns

PaperSize

North American standard sheet media size "5x7".

# Field Na6X9

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "6x9".

```
public static readonly PaperSize Na6X9
```

## Returns

PaperSize

North American standard sheet media size "6x9".

# Field Na7X9

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "7x9".

```
public static readonly PaperSize Na7X9
```

## Returns

PaperSize

North American standard sheet media size "7x9".

# Field Na9X11

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "9x11".

```
public static readonly PaperSize Na9X11
```

## Returns

PaperSize

North American standard sheet media size "9x11".

# Field NaA2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "A2".

```
public static readonly PaperSize NaA2
```

## Returns

PaperSize

North American standard sheet media size "A2".

# Field NaArchA

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Arch A".

```
public static readonly PaperSize NaArchA
```

## Returns

PaperSize

North American standard sheet media size "Arch A".

# Field NaArchB

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Arch B".

```
public static readonly PaperSize NaArchB
```

## Returns

PaperSize

North American standard sheet media size "Arch B".

# Field NaArchC

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Arch C".

```
public static readonly PaperSize NaArchC
```

## Returns

PaperSize

North American standard sheet media size "Arch C".

# Field NaArchD

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Arch D".

```
public static readonly PaperSize NaArchD
```

## Returns

PaperSize

North American standard sheet media size "Arch D".

# Field NaArchE

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Arch E".

```
public static readonly PaperSize NaArchE
```

## Returns

PaperSize

North American standard sheet media size "Arch E".

# Field NaAsmeF

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "ASME F".

```
public static readonly PaperSize NaAsmeF
```

## Returns

PaperSize

North American standard sheet media size "ASME F".

# Field NaBPlus

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "B-plus".

```
public static readonly PaperSize NaBPlus
```

## Returns

PaperSize

North American standard sheet media size "B-plus".

# Field NaC

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "C".

```
public static readonly PaperSize NaC
```

## Returns

PaperSize

North American standard sheet media size "C".

# Field NaC5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "C5".

```
public static readonly PaperSize NaC5
```

## Returns

PaperSize

North American standard sheet media size "C5".

# Field NaD

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "D".

```
public static readonly PaperSize NaD
```

## Returns

PaperSize

North American standard sheet media size "D".

# Field NaE

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "E".

```
public static readonly PaperSize NaE
```

## Returns

PaperSize

North American standard sheet media size "E".

# Field NaEdp

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "EDP".

```
public static readonly PaperSize NaEdp
```

## Returns

PaperSize

North American standard sheet media size "EDP".

# Field NaEurEdp

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "European EDP".

```
public static readonly PaperSize NaEurEdp
```

## Returns

PaperSize

North American standard sheet media size "European EDP".

# Field NaExecutive

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Executive".

```
public static readonly PaperSize NaExecutive
```

## Returns

PaperSize

North American standard sheet media size "Executive".

# Field NaF

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "F".

```
public static readonly PaperSize NaF
```

## Returns

PaperSize

North American standard sheet media size "F".

# Field NaFanfoldEur

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "FanFold European".

```
public static readonly PaperSize NaFanfoldEur
```

## Returns

PaperSize

North American standard sheet media size "FanFold European".

# Field NaFanfoldUs

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "FanFold US".

```
public static readonly PaperSize NaFanfoldUs
```

## Returns

PaperSize

North American standard sheet media size "FanFold US".

# Field NaFoolscap

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "FanFold German Legal".

```
public static readonly PaperSize NaFoolscap
```

## Returns

PaperSize

North American standard sheet media size "FanFold German Legal".

# Field NaGovtLegal

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Government Legal".

```
public static readonly PaperSize NaGovtLegal
```

## Returns

PaperSize

North American standard sheet media size "Government Legal".

# Field NaGovtLetter

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Government Letter".

```
public static readonly PaperSize NaGovtLetter
```

## Returns

PaperSize

North American standard sheet media size "Government Letter".

# Field NaIndex3X5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Index 3x5".

```
public static readonly PaperSize NaIndex3X5
```

## Returns

PaperSize

North American standard sheet media size "Index 3x5".

# Field NaIndex4X6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Index 4x6".

```
public static readonly PaperSize NaIndex4X6
```

## Returns

PaperSize

North American standard sheet media size "Index 4x6".

# Field NaIndex4X6Ext

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Index 4x6 ext".

```
public static readonly PaperSize NaIndex4X6Ext
```

## Returns

PaperSize

North American standard sheet media size "Index 4x6 ext".

# Field NaIndex5X8

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Index 5x8".

```
public static readonly PaperSize NaIndex5X8
```

## Returns

PaperSize

North American standard sheet media size "Index 5x8".

# Field NaInvoice

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Invoice".

```
public static readonly PaperSize NaInvoice
```

## Returns

PaperSize

North American standard sheet media size "Invoice".

# Field NaLedger

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Tabloid".

```
public static readonly PaperSize NaLedger
```

## Returns

PaperSize

North American standard sheet media size "Tabloid".

# Field NaLegal

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Legal".

```
public static readonly PaperSize NaLegal
```

## Returns

PaperSize

North American standard sheet media size "Legal".

# Field NaLegalExtra

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Legal extra".

```
public static readonly PaperSize NaLegalExtra
```

## Returns

PaperSize

North American standard sheet media size "Legal extra".

# Field NaLetter

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Letter".

```
public static readonly PaperSize NaLetter
```

## Returns

PaperSize

North American standard sheet media size "Letter".

# Field NaLetterExtra

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Letter extra".

```
public static readonly PaperSize NaLetterExtra
```

## Returns

PaperSize

North American standard sheet media size "Letter extra".

# Field NaLetterPlus

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Letter plus".

```
public static readonly PaperSize NaLetterPlus
```

## Returns

PaperSize

North American standard sheet media size "Letter plus".

# Field NaMonarch

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Monarch".

```
public static readonly PaperSize NaMonarch
```

## Returns

PaperSize

North American standard sheet media size "Monarch".

# Field NaNumber10

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Envelope #10".

```
public static readonly PaperSize NaNumber10
```

## Returns

PaperSize

North American standard sheet media size "Envelope #10".

# Field NaNumber11

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Envelope #11".

```
public static readonly PaperSize NaNumber11
```

## Returns

PaperSize

North American standard sheet media size "Envelope #11".

# Field NaNumber12

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Envelope #12".

```
public static readonly PaperSize NaNumber12
```

## Returns

PaperSize

North American standard sheet media size "Envelope #12".

# Field NaNumber14

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Envelope #14".

```
public static readonly PaperSize NaNumber14
```

## Returns

PaperSize

North American standard sheet media size "Envelope #14".

# Field NaNumber9

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Envelope #9".

```
public static readonly PaperSize NaNumber9
```

## Returns

PaperSize

North American standard sheet media size "Envelope #9".

# Field NaPersonal

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Personal".

```
public static readonly PaperSize NaPersonal
```

## Returns

PaperSize

North American standard sheet media size "Personal".

# Field NaQuarto

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Quarto".

```
public static readonly PaperSize NaQuarto
```

## Returns

PaperSize

North American standard sheet media size "Quarto".

# Field NaSuperA

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Super A".

```
public static readonly PaperSize NaSuperA
```

## Returns

PaperSize

North American standard sheet media size "Super A".

# Field NaSuperB

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Super B".

```
public static readonly PaperSize NaSuperB
```

## Returns

PaperSize

North American standard sheet media size "Super B".

# Field NaWideFormat

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

North American standard sheet media size "Wide format".

```
public static readonly PaperSize NaWideFormat
```

## Returns

PaperSize

North American standard sheet media size "Wide format".

# Field OmDaiPaKai

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "Dai-pa-kai".

```
public static readonly PaperSize OmDaiPaKai
```

## Returns

PaperSize

Chinese standard sheet media size "Dai-pa-kai".

# Field OmFolio

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Other metric standard sheet media size "Folio".

```
public static readonly PaperSize OmFolio
```

## Returns

PaperSize

Other metric standard sheet media size "Folio".

# Field OmFolioSp

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Other metric standard sheet media size "Folio SP".

```
public static readonly PaperSize OmFolioSp
```

## Returns

PaperSize

Other metric standard sheet media size "Folio SP".

# Field OmInvite

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Other metric standard sheet media size "Invite Envelope".

```
public static readonly PaperSize OmInvite
```

## Returns

PaperSize

Other metric standard sheet media size "Invite Envelope".

# Field OmItalian

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Other metric standard sheet media size "Italian Envelope".

```
public static readonly PaperSize OmItalian
```

## Returns

PaperSize

Other metric standard sheet media size "Italian Envelope".

# Field OmJuuroKuKai

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "Juuro-ku-kai".

```
public static readonly PaperSize OmJuuroKuKai
```

## Returns

PaperSize

Chinese standard sheet media size "Juuro-ku-kai".

# Field OmLargePhoto

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Other metric standard sheet media size "Large photo".

```
public static readonly PaperSize OmLargePhoto
```

## Returns

PaperSize

Other metric standard sheet media size "Large photo".

# Field OmPaKai

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "Pa-kai".

```
public static readonly PaperSize OmPaKai
```

## Returns

PaperSize

Chinese standard sheet media size "Pa-kai".

# Field OmPostfix

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Other metric standard sheet media size "Postfix Envelope".

```
public static readonly PaperSize OmPostfix
```

## Returns

PaperSize

Other metric standard sheet media size "Postfix Envelope".

# Field OmSmallPhoto

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Other metric standard sheet media size "Small photo".

```
public static readonly PaperSize OmSmallPhoto
```

## Returns

PaperSize

Other metric standard sheet media size "Small photo".

# Field Prc1

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc1 Envelope".

```
public static readonly PaperSize Prc1
```

## Returns

PaperSize

Chinese standard sheet media size "prc1 Envelope".

# Field Prc10

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc10 Envelope".

```
public static readonly PaperSize Prc10
```

## Returns

PaperSize

Chinese standard sheet media size "prc10 Envelope".

# Field Prc16K

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc 16k".

```
public static readonly PaperSize Prc16K
```

## Returns

PaperSize

Chinese standard sheet media size "prc 16k".

# Field Prc2

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc2 Envelope".

```
public static readonly PaperSize Prc2
```

## Returns

PaperSize

Chinese standard sheet media size "prc2 Envelope".

# Field Prc3

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc3 Envelope".

```
public static readonly PaperSize Prc3
```

## Returns

PaperSize

Chinese standard sheet media size "prc3 Envelope".

# Field Prc32K

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc 32k".

```
public static readonly PaperSize Prc32K
```

## Returns

PaperSize

Chinese standard sheet media size "prc 32k".

# Field Prc4

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc4 Envelope".

```
public static readonly PaperSize Prc4
```

## Returns

PaperSize

Chinese standard sheet media size "prc4 Envelope".

# Field Prc5

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc5 Envelope".

```
public static readonly PaperSize Prc5
```

## Returns

PaperSize

Chinese standard sheet media size "prc5 Envelope".

# Field Prc6

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc6 Envelope".

```
public static readonly PaperSize Prc6
```

## Returns

PaperSize

Chinese standard sheet media size "prc6 Envelope".

# Field Prc7

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc7 Envelope".

```
public static readonly PaperSize Prc7
```

## Returns

PaperSize

Chinese standard sheet media size "prc7 Envelope".

# Field Prc8

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "prc8 Envelope".

```
public static readonly PaperSize Prc8
```

## Returns

PaperSize

Chinese standard sheet media size "prc8 Envelope".

# Field Roc16K

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "ROC 16K".

```
public static readonly PaperSize Roc16K
```

## Returns

PaperSize

Chinese standard sheet media size "ROC 16K".

# Field Roc8K

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Chinese standard sheet media size "ROC 8K".

```
public static readonly PaperSize Roc8K
```

## Returns

PaperSize

Chinese standard sheet media size "ROC 8K".

# Property Height

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Height

Gets the paper height in microns.

```
public int Height { get; }
```

## Property Value

int

The paper height in microns.

# Property Name

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Name

Gets the paper size name.

```
public string Name { get; }
```

## Property Value

string

The paper size name. Can be null or empty.

# Property Width

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Width

Gets the paper width in microns.

```
public int Width { get; }
```

## Property Value

int

The paper width in microns.

# Method Equals

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Method Of

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Of(float, float, Unit, string)

Creates a new paper size setting.

```
public static PaperSize Of(float width, float height, PaperSize.Unit unit, string name)
```

## Parameters

`width` float

The paper size width.

`height` float

The paper size height.

`unit` PaperSize.Unit

The `width` and `height` measurement unit.

`name` string

The name of the paper size.

## Returns

PaperSize

The paper size used for printing.

## Exceptions

ArgumentException

The `width` is negative.

ArgumentException

The `height` is negative.

ArgumentException

The `name` is null , empty or contains only white space.

## Of(float, float, Unit)

Creates a new paper size setting.

```
public static PaperSize Of(float width, float height, PaperSize.Unit unit)
```

### Parameters

`width` float

The paper size width.

`height` float

The paper size height.

`unit` PaperSize.Unit

The `width` and `height` measurement unit.

### Returns

PaperSize

The paper size used for printing.

### Exceptions

ArgumentException

The `width` is negative.

ArgumentException

The `height` is negative.

# Method ToString

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## ToString()

```
public override string ToString()
```

## Returns

string

# Enum PaperSize.Unit

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The measurement unit of the paper size.

```
public enum PaperSize.Unit
```

## Fields

`Inches = 1`

The paper size is measured in inches.

`Microns = 2`

The paper size is measured in microns.

`Millimeters = 0`

The paper size is measured in millimeters.

# Class PdfPrinter

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

A printer that allows you to print to PDF.

```
public static class PdfPrinter
```

### Inheritance

object ← PdfPrinter

### Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Interface PdfPrinter.IHtmlSettings

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The print settings available when printing HTML content on the PDF printer.

```
public interface PdfPrinter.IHtmlSettings : PdfPrinter.ISettings<PdfPrinter.IHtmlSettings>,
IPrintSettings, IPageRanges<PdfPrinter.IHtmlSettings>,
IPagesPerSheet<PdfPrinter.IHtmlSettings>, IPdfFilePath<PdfPrinter.IHtmlSettings>,
IScaling<PdfPrinter.IHtmlSettings>, IPageMargins<PdfPrinter.IHtmlSettings>,
IPrintHeaderFooter<PdfPrinter.IHtmlSettings>, IOrientation<PdfPrinter.IHtmlSettings>,
IPrintBackgrounds<PdfPrinter.IHtmlSettings>, IPrintSelectionOnly<PdfPrinter.IHtmlSettings>,
IPaperSize<PdfPrinter.IHtmlSettings>, IHeaderTemplate<PdfPrinter.IHtmlSettings>,
IFooterTemplate<PdfPrinter.IHtmlSettings>
```

**Inherited Members**

IPrintSettings.Apply() , IPageRanges<PdfPrinter.IHtmlSettings>.PageRanges , IPagesPerSheet<PdfPrinter.IHtmlSettings>.PagesPerSheet , IPdfFilePath<PdfPrinter.IHtmlSettings>.PdfFilePath , IScaling<PdfPrinter.IHtmlSettings>.Scaling , IPageMargins<PdfPrinter.IHtmlSettings>.PageMargins , IPrintHeaderFooter<PdfPrinter.IHtmlSettings>.PrintingHeaderFooterEnabled , IOrientation<PdfPrinter.IHtmlSettings>.Orientation , IPrintBackgrounds<PdfPrinter.IHtmlSettings>.PrintingBackgroundsEnabled , IPrintSelectionOnly<PdfPrinter.IHtmlSettings>.PrintingSelectionOnlyEnabled , IPaperSize<PdfPrinter.IHtmlSettings>.PaperSize , IHeaderTemplate<PdfPrinter.IHtmlSettings>.HeaderTemplate , IFooterTemplate<PdfPrinter.IHtmlSettings>.FooterTemplate

**Extension Methods**

PrintSettingsExtensions.Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>) , FooterTemplateExtensions.SetFooterTemplate<TPrintSettings>(IFooterTemplate<TPrintSettings>, string) , HeaderTemplateExtensions.SetHeaderTemplate<TPrintSettings>(IHeaderTemplate<TPrintSettings>, string) , OrientationExtensions.SetOrientation<TPrintSettings>(IOrientation<TPrintSettings>, Orientation) , PageMarginsExtensions.SetPageMargins<TPrintSettings>(IPageMargins<TPrintSettings>, PageMargins) , PageRangesExtensions.SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>) ,

PagesPerSheetExtensions.SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet) ,
PaperSizeExtensions.SetPaperSize<TPrintSettings>(IPaperSize<TPrintSettings>, PaperSize) ,
PdfFilePathExtensions.SetPdfFilePath<TPrintSettings>(IPdfFilePath<TPrintSettings>, string) ,
PrintBackgroundsExtensions.DisablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>) ,
PrintBackgroundsExtensions.EnablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>) ,
PrintHeaderFooterExtensions.DisablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>) ,
PrintHeaderFooterExtensions.EnablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>) ,
PrintSelectionOnlyExtensions.DisablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>) ,
PrintSelectionOnlyExtensions.EnablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>) ,
ScalingExtensions.SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling)

# Interface PdfPrinter.IPdfSettings

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The print settings available when printing PDF content on the PDF printer.

```
public interface PdfPrinter.IPdfSettings : PdfPrinter.ISettings<PdfPrinter.IPdfSettings>,
IPrintSettings, IPageRanges<PdfPrinter.IPdfSettings>,
IPagesPerSheet<PdfPrinter.IPdfSettings>, IPdfFilePath<PdfPrinter.IPdfSettings>,
IScaling<PdfPrinter.IPdfSettings>
```

**Inherited Members**

IPrintSettings.Apply() , IPageRanges<PdfPrinter.IPdfSettings>.PageRanges , IPagesPerSheet<PdfPrinter.IPdfSettings>.PagesPerSheet , IPdfFilePath<PdfPrinter.IPdfSettings>.PdfFilePath , IScaling<PdfPrinter.IPdfSettings>.Scaling

**Extension Methods**

PrintSettingsExtensions.Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>) , PageRangesExtensions.SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>) , PagesPerSheetExtensions.SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet) , PdfFilePathExtensions.SetPdfFilePath<TPrintSettings>(IPdfFilePath<TPrintSettings>, string) , ScalingExtensions.SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling)

# Interface PdfPrinter.ISettings<T>

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The print settings available when printing HTML or PDF content on the PDF printer.

```
public interface PdfPrinter.ISettings<out T> : IPrintSettings, IPageRanges<T>,
IPagesPerSheet<T>, IPdfFilePath<T>, IScaling<T> where T : class, IPrintSettings
```

## Type Parameters

`T`

The specific print settings type.

**Inherited Members**

IPrintSettings.Apply() , IPageRanges<T>.PageRanges , IPagesPerSheet<T>.PagesPerSheet , IPdfFilePath<T>.PdfFilePath , IScaling<T>.Scaling

**Extension Methods**

PrintSettingsExtensions.Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>) , PageRangesExtensions.SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>) , PagesPerSheetExtensions.SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet) , PdfFilePathExtensions.SetPdfFilePath<TPrintSettings>(IPdfFilePath<TPrintSettings>, string) , ScalingExtensions.SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling)

# Class PdfPrinter<TPrintSettings>

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

A printer that allows you to print to PDF.

```
public sealed class PdfPrinter<TPrintSettings> : Printer<TPrintSettings>, IAutoDisposable
where TPrintSettings : class, PdfPrinter.ISettings<TPrintSettings>
```

## Type Parameters

`TPrintSettings`

The print settings type that can be applied to the print job.

**Inheritance**

object ← Printer<TPrintSettings> ← PdfPrinter<TPrintSettings>

**Implements**

IAutoDisposable

**Inherited Members**

Printer<TPrintSettings>.Capabilities , Printer<TPrintSettings>.DeviceName , Printer<TPrintSettings>.IsDefault , Printer<TPrintSettings>.PrintJob , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class Printer<TPrintSettings>

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The base class for the printers.

```
public abstract class Printer<TPrintSettings> : IAutoDisposable where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The print settings type available for this printer.

**Inheritance**

object ← Printer<TPrintSettings>

**Implements**

IAutoDisposable

**Derived**

PdfPrinter<TPrintSettings>, SystemPrinter<TPrintSettings>

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Properties

Capabilities

Gets the capabilities of this printer.

DeviceName

Gets the device name of this printer.

IsDefault

Indicates whether this printer is the default system printer.

PrintJob

Gets the current print job. This print job can be used to configure print settings.

# Property Capabilities

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Capabilities

Gets the capabilities of this printer.

```
public Capabilities Capabilities { get; }
```

## Property Value

Capabilities

The capabilities of this printer.

# Property DeviceName

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## DeviceName

Gets the device name of this printer.

```
public string DeviceName { get; }
```

## Property Value

string

The device name of this printer.

# Property IsDefault

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## IsDefault

Indicates whether this printer is the default system printer.

```
public bool IsDefault { get; }
```

## Property Value

bool

`true` if this printer is the default system printer, `false` otherwise.

# Property PrintJob

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## PrintJob

Gets the current print job. This print job can be used to configure print settings.

```
public IPrintJob<TPrintSettings> PrintJob { get; }
```

## Property Value

IPrintJob<TPrintSettings>

The current print job.

# Class Scaling

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The scaling used for printing.

```
public sealed class Scaling
```

**Inheritance**

object ← Scaling

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Default

The default scaling for the printing content.

## Methods

Custom(int)

Creates a custom scaling which scales the content according to the given scale factor.

Equals(object)

GetHashCode()

# Field Default

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

The default scaling for the printing content.

```
public static readonly Scaling Default
```

## Returns

Scaling

The default scaling for the printing content.

# Method Custom

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Custom(int)

Creates a custom scaling which scales the content according to the given scale factor.

```
public static Scaling Custom(int scaleFactor)
```

## Parameters

`scaleFactor` int

The scale factor to apply to the content, where

```
100
```

is a normal size. Should be in the range from

```
10
```

to

```
200
```

.

## Returns

Scaling

The scaling used for printing.

## Exceptions

[ArgumentOutOfRangeException]

The `scaleFactor` is out of range. Should be in the range from

```
10
```

to

```
200
```

.

# Method Equals

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Class SystemPrinter

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

A local or network printer available in the system.

```
public static class SystemPrinter
```

### Inheritance

object ← SystemPrinter

### Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Interface SystemPrinter.IHtmlSettings

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Print settings available when printing HTML content on a physical printer.

```
public interface SystemPrinter.IHtmlSettings :
SystemPrinter.ISettings<SystemPrinter.IHtmlSettings>, IPrintSettings,
ICollate<SystemPrinter.IHtmlSettings>, IColorModel<SystemPrinter.IHtmlSettings>,
ICopies<SystemPrinter.IHtmlSettings>, IDuplexMode<SystemPrinter.IHtmlSettings>,
IPageRanges<SystemPrinter.IHtmlSettings>, IPagesPerSheet<SystemPrinter.IHtmlSettings>,
IPaperSize<SystemPrinter.IHtmlSettings>, IScaling<SystemPrinter.IHtmlSettings>,
IPageMargins<SystemPrinter.IHtmlSettings>, IPrintHeaderFooter<SystemPrinter.IHtmlSettings>,
IOrientation<SystemPrinter.IHtmlSettings>, IPrintBackgrounds<SystemPrinter.IHtmlSettings>,
IPrintSelectionOnly<SystemPrinter.IHtmlSettings>,
IHeaderTemplate<SystemPrinter.IHtmlSettings>, IFooterTemplate<SystemPrinter.IHtmlSettings>
```

### Inherited Members

IPrintSettings.Apply() , ICollate<SystemPrinter.IHtmlSettings>.CollatePrintingEnabled , IColorModel<SystemPrinter.IHtmlSettings>.ColorModel , ICopies<SystemPrinter.IHtmlSettings>.Copies , IDuplexMode<SystemPrinter.IHtmlSettings>.DuplexMode , IPageRanges<SystemPrinter.IHtmlSettings>.PageRanges , IPagesPerSheet<SystemPrinter.IHtmlSettings>.PagesPerSheet , IPaperSize<SystemPrinter.IHtmlSettings>.PaperSize , IScaling<SystemPrinter.IHtmlSettings>.Scaling , IPageMargins<SystemPrinter.IHtmlSettings>.PageMargins , IPrintHeaderFooter<SystemPrinter.IHtmlSettings>.PrintingHeaderFooterEnabled , IOrientation<SystemPrinter.IHtmlSettings>.Orientation , IPrintBackgrounds<SystemPrinter.IHtmlSettings>.PrintingBackgroundsEnabled , IPrintSelectionOnly<SystemPrinter.IHtmlSettings>.PrintingSelectionOnlyEnabled , IHeaderTemplate<SystemPrinter.IHtmlSettings>.HeaderTemplate , IFooterTemplate<SystemPrinter.IHtmlSettings>.FooterTemplate

### Extension Methods

CollateExtensions.DisableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>) , CollateExtensions.EnableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>) , ColorModelExtensions.SetColorModel<TPrintSettings>(IColorModel<TPrintSettings>, ColorModel) , CopiesExtensions.SetCopies<TPrintSettings>(ICopies<TPrintSettings>, int) , DuplexModeExtensions.SetDuplexMode<TPrintSettings>(IDuplexMode<TPrintSettings>, DuplexMode) ,

FooterTemplateExtensions.SetFooterTemplate<TPrintSettings>(IFooterTemplate<TPrintSettings>, string) ,
HeaderTemplateExtensions.SetHeaderTemplate<TPrintSettings>(IHeaderTemplate<TPrintSettings>, string) ,
OrientationExtensions.SetOrientation<TPrintSettings>(IOrientation<TPrintSettings>, Orientation) ,
PageMarginsExtensions.SetPageMargins<TPrintSettings>(IPageMargins<TPrintSettings>, PageMargins) ,
PageRangesExtensions.SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>) ,
PagesPerSheetExtensions.SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet) ,
PaperSizeExtensions.SetPaperSize<TPrintSettings>(IPaperSize<TPrintSettings>, PaperSize) ,
PrintBackgroundsExtensions.DisablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>) ,
PrintBackgroundsExtensions.EnablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>) ,
PrintHeaderFooterExtensions.DisablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>) ,
PrintHeaderFooterExtensions.EnablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>) ,
PrintSelectionOnlyExtensions.DisablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>) ,
PrintSelectionOnlyExtensions.EnablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>) ,
ScalingExtensions.SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling) ,
PrintSettingsExtensions.Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>)

# Interface SystemPrinter.IPdfSettings

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Print settings available when printing PDF content on a physical printer.

```
public interface SystemPrinter.IPdfSettings :
SystemPrinter.ISettings<SystemPrinter.IPdfSettings>, IPrintSettings,
ICollate<SystemPrinter.IPdfSettings>, IColorModel<SystemPrinter.IPdfSettings>,
ICopies<SystemPrinter.IPdfSettings>, IDuplexMode<SystemPrinter.IPdfSettings>,
IPageRanges<SystemPrinter.IPdfSettings>, IPagesPerSheet<SystemPrinter.IPdfSettings>,
IPaperSize<SystemPrinter.IPdfSettings>, IScaling<SystemPrinter.IPdfSettings>,
IFit<SystemPrinter.IPdfSettings>
```

### Inherited Members

IPrintSettings.Apply() , ICollate<SystemPrinter.IPdfSettings>.CollatePrintingEnabled , IColorModel<SystemPrinter.IPdfSettings>.ColorModel , ICopies<SystemPrinter.IPdfSettings>.Copies , IDuplexMode<SystemPrinter.IPdfSettings>.DuplexMode , IPageRanges<SystemPrinter.IPdfSettings>.PageRanges , IPagesPerSheet<SystemPrinter.IPdfSettings>.PagesPerSheet , IPaperSize<SystemPrinter.IPdfSettings>.PaperSize , IScaling<SystemPrinter.IPdfSettings>.Scaling , IFit<SystemPrinter.IPdfSettings>.Fit

### Extension Methods

CollateExtensions.DisableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>) , CollateExtensions.EnableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>) , ColorModelExtensions.SetColorModel<TPrintSettings>(IColorModel<TPrintSettings>, ColorModel) , CopiesExtensions.SetCopies<TPrintSettings>(ICopies<TPrintSettings>, int) , DuplexModeExtensions.SetDuplexMode<TPrintSettings>(IDuplexMode<TPrintSettings>, DuplexMode) , FitExtensions.SetFit<TPrintSettings>(IFit<TPrintSettings>, Fit) , PageRangesExtensions.SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>) , PagesPerSheetExtensions.SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet) , PaperSizeExtensions.SetPaperSize<TPrintSettings>(IPaperSize<TPrintSettings>, PaperSize) , ScalingExtensions.SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling) , PrintSettingsExtensions.Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>)

# Interface SystemPrinter.ISettings<T>

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

Print settings available when printing HTML or PDF content on a physical printer.

```
public interface SystemPrinter.ISettings<out T> : IPrintSettings, ICollate<T>,
ICColorModel<T>, ICopies<T>, IDuplexMode<T>, IPageRanges<T>, IPagesPerSheet<T>,
IPaperSize<T>, IScaling<T> where T : class, IPrintSettings
```

## Type Parameters

`T`

The specific print settings type.

**Inherited Members**

IPrintSettings.Apply() , ICollate<T>.CollatePrintingEnabled , IColorModel<T>.ColorModel , ICopies<T>.Copies , IDuplexMode<T>.DuplexMode , IPageRanges<T>.PageRanges , IPagesPerSheet<T>.PagesPerSheet , IPaperSize<T>.PaperSize , IScaling<T>.Scaling

**Extension Methods**

CollateExtensions.DisableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>) , CollateExtensions.EnableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>) , ColorModelExtensions.SetColorModel<TPrintSettings>(IColorModel<TPrintSettings>, ColorModel) , CopiesExtensions.SetCopies<TPrintSettings>(ICopies<TPrintSettings>, int) , DuplexModeExtensions.SetDuplexMode<TPrintSettings>(IDuplexMode<TPrintSettings>, DuplexMode) , PageRangesExtensions.SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>) , PagesPerSheetExtensions.SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet) , PaperSizeExtensions.SetPaperSize<TPrintSettings>(IPaperSize<TPrintSettings>, PaperSize) , ScalingExtensions.SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling) , PrintSettingsExtensions.Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>)

# Class SystemPrinter<TPrintSettings>

Namespace: DotNetBrowser.Print

Assembly: DotNetBrowser.dll

A local or network printer available in the system.

```
public sealed class SystemPrinter<TPrintSettings> : Printer<TPrintSettings>,
IAutoDisposable where TPrintSettings : class, SystemPrinter.ISettings<TPrintSettings>
```

## Type Parameters

`TPrintSettings`

### Inheritance

object ← Printer<TPrintSettings> ← SystemPrinter<TPrintSettings>

### Implements

IAutoDisposable

### Inherited Members

Printer<TPrintSettings>.Capabilities , Printer<TPrintSettings>.DeviceName , Printer<TPrintSettings>.IsDefault , Printer<TPrintSettings>.PrintJob , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Namespace DotNetBrowser.Print.Events

## Classes

PageCountUpdatedEventArgs<TPrintSettings>
: Event arguments for the PageCountUpdated event.

PrintCompletedEventArgs<TPrintSettings>
: Event arguments for the PrintCompleted event.

# Class PageCountUpdatedEventArgs<TPrintSettings>

Namespace: DotNetBrowser.Print.Events

Assembly: DotNetBrowser.dll

Event arguments for the PageCountUpdated event.

```
public sealed class PageCountUpdatedEventArgs<TPrintSettings> : EventArgs where
TPrintSettings : class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The print settings type.

**Inheritance**

object ← EventArgs ← PageCountUpdatedEventArgs<TPrintSettings>

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

PageCount

Gets the total number of pages to be printed.

PrintJob

Gets the print job associated with this event.

# Property PageCount

Namespace: DotNetBrowser.Print.Events

Assembly: DotNetBrowser.dll

## PageCount

Gets the total number of pages to be printed.

```
public int PageCount { get; }
```

## Property Value

int

The total number of pages to be printed.

# Property PrintJob

Namespace: DotNetBrowser.Print.Events

Assembly: DotNetBrowser.dll

## PrintJob

Gets the print job associated with this event.

```
public IPrintJob<TPrintSettings> PrintJob { get; }
```

## Property Value

IPrintJob<TPrintSettings>

The print job associated with this event.

# Class PrintCompletedEventArgs<TPrintSettings>

Namespace: DotNetBrowser.Print.Events

Assembly: DotNetBrowser.dll

Event arguments for the PrintCompleted event.

```
public sealed class PrintCompletedEventArgs<TPrintSettings> : EventArgs where
TPrintSettings : class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The print settings type.

**Inheritance**

object ← EventArgs ← PrintCompletedEventArgs<TPrintSettings>

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

IsCompletedSuccessfully

Indicates whether the printing is completed successfully.

PrintJob

Gets the print job associated with this event.

# Property IsCompletedSuccessfully

Namespace: DotNetBrowser.Print.Events

Assembly: DotNetBrowser.dll

## IsCompletedSuccessfully

Indicates whether the printing is completed successfully.

```
public bool IsCompletedSuccessfully { get; }
```

## Property Value

bool

`true` if the printing is completed successfully, `false` otherwise.

# Property PrintJob

Namespace: DotNetBrowser.Print.Events

Assembly: DotNetBrowser.dll

## PrintJob

Gets the print job associated with this event.

```
public IPrintJob<TPrintSettings> PrintJob { get; }
```

## Property Value

IPrintJob<TPrintSettings>

The print job associated with this event.

# Namespace DotNetBrowser.Print.Handlers

## Classes

PrintContentParameters<TPrintSettings, TPdfPrintSettings>
: The base class for content printing handlers parameters.

PrintHtmlContentParameters
: The parameters of the PrintHtmlContentHandler.

PrintHtmlContentResponse
: A response for the PrintHtmlContentHandler.

PrintPdfContentParameters
: The parameters of the PrintPdfContentHandler.

PrintPdfContentResponse
: A response for the PrintPdfContentHandler.

# Class PrintContentParameters<TPrintSettings, TPdfPrintSettings>

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

The base class for content printing handlers parameters.

```
public class PrintContentParameters<TPrintSettings, TPdfPrintSettings> where TPrintSettings
: class, SystemPrinter.ISettings<TPrintSettings> where TPdfPrintSettings : class,
PdfPrinter.ISettings<TPdfPrintSettings>
```

## Type Parameters

`TPrintSettings`

The system print settings type.

`TPdfPrintSettings`

The PDF print settings type.

**Inheritance**

object ← PrintContentParameters<TPrintSettings, TPdfPrintSettings>

**Derived**

PrintHtmlContentParameters, PrintPdfContentParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

# Properties

Browser

Gets the browser associated with the handler.

Printers

Gets the collection of printers available in this handler.

# Property Browser

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

## Browser

Gets the browser associated with the handler.

```
public IBrowser Browser { get; }
```

## Property Value

IBrowser

The browser associated with the handler.

# Property Printers

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

## Printers

Gets the collection of printers available in this handler.

```
public IPrinters<TPrintSettings, TPdfPrintSettings> Printers { get; }
```

## Property Value

IPrinters<TPrintSettings, TPdfPrintSettings>

The collection of printers available in this handler.

# Class PrintHtmlContentParameters

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

The parameters of the PrintHtmlContentHandler.

```
public sealed class PrintHtmlContentParameters :
PrintContentParameters<SystemPrinter.IHtmlSettings, PdfPrinter.IHtmlSettings>
```

### Inheritance

object ← PrintContentParameters<SystemPrinter.IHtmlSettings, PdfPrinter.IHtmlSettings> ← PrintHtmlContentParameters

### Inherited Members

PrintContentParameters<SystemPrinter.IHtmlSettings, PdfPrinter.IHtmlSettings>.Browser , PrintContentParameters<SystemPrinter.IHtmlSettings, PdfPrinter.IHtmlSettings>.Printers , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class PrintHtmlContentResponse

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

A response for the PrintHtmlContentHandler.

```
public class PrintHtmlContentResponse
```

**Inheritance**

object ← PrintHtmlContentResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a PrintHtmlContentResponse instance that tells the Chromium engine to cancel printing.

Print(PdfPrinter<IHtmlSettings>)

Creates a PrintHtmlContentResponse instance that tells the Chromium engine to proceed with the passed printer.

Print(SystemPrinter<IHtmlSettings>)

Creates a PrintHtmlContentResponse instance that tells the Chromium engine to proceed with the passed printer.

# Method Cancel

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a PrintHtmlContentResponse instance that tells the Chromium engine to cancel printing.

```
public static PrintHtmlContentResponse Cancel()
```

## Returns

PrintHtmlContentResponse

The PrintHtmlContentResponse instance that can be used as a return value in PrintHtmlContentHandler implementation.

# Method Print

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

## Print(SystemPrinter<IHtmlSettings>)

Creates a PrintHtmlContentResponse instance that tells the Chromium engine to proceed with the passed printer.

```
public static PrintHtmlContentResponse Print(SystemPrinter<SystemPrinter.IHtmlSettings>
printer)
```

### Parameters

`printer` SystemPrinter<SystemPrinter.IHtmlSettings>

The printer to use.

### Returns

PrintHtmlContentResponse

The PrintHtmlContentResponse instance that can be used as a return value in PrintHtmlContentHandler implementation.

## Print(PdfPrinter<IHtmlSettings>)

Creates a PrintHtmlContentResponse instance that tells the Chromium engine to proceed with the passed printer.

```
public static PrintHtmlContentResponse Print(PdfPrinter<PdfPrinter.IHtmlSettings> printer)
```

### Parameters

`printer` PdfPrinter<PdfPrinter.IHtmlSettings>

The printer to use.

## Returns

PrintHtmlContentResponse

The PrintHtmlContentResponse instance that can be used as a return value in PrintHtmlContentHandler implementation.

# Class PrintPdfContentParameters

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

The parameters of the PrintPdfContentHandler.

```
public sealed class PrintPdfContentParameters :
PrintContentParameters<SystemPrinter.IPdfSettings, PdfPrinter.IPdfSettings>
```

### Inheritance

object ← PrintContentParameters<SystemPrinter.IPdfSettings, PdfPrinter.IPdfSettings> ← PrintPdfContentParameters

### Inherited Members

PrintContentParameters<SystemPrinter.IPdfSettings, PdfPrinter.IPdfSettings>.Browser , PrintContentParameters<SystemPrinter.IPdfSettings, PdfPrinter.IPdfSettings>.Printers , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

# Class PrintPdfContentResponse

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

A response for the PrintPdfContentHandler.

```
public class PrintPdfContentResponse
```

**Inheritance**

object ← PrintPdfContentResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Cancel()

Creates a PrintPdfContentResponse instance that tells the Chromium engine to cancel printing.

Print(PdfPrinter<IPdfSettings>)

Creates a PrintPdfContentResponse instance that tells the Chromium engine to proceed with the passed printer.

Print(SystemPrinter<IPdfSettings>)

Creates a PrintPdfContentResponse instance that tells the Chromium engine to proceed with the passed printer.

# Method Cancel

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

## Cancel()

Creates a PrintPdfContentResponse instance that tells the Chromium engine to cancel printing.

```
public static PrintPdfContentResponse Cancel()
```

## Returns

PrintPdfContentResponse

The PrintPdfContentResponse instance that can be used as a return value in PrintPdfContentHandler implementation.

# Method Print

Namespace: DotNetBrowser.Print.Handlers

Assembly: DotNetBrowser.dll

## Print(SystemPrinter<IPdfSettings>)

Creates a PrintPdfContentResponse instance that tells the Chromium engine to proceed with the passed printer.

```
public static PrintPdfContentResponse Print(SystemPrinter<SystemPrinter.IPdfSettings>
printer)
```

### Parameters

`printer` SystemPrinter<SystemPrinter.IPdfSettings>

The printer to use.

### Returns

PrintPdfContentResponse

The PrintPdfContentResponse instance that can be used as a return value in PrintPdfContentHandler implementation.

## Print(PdfPrinter<IPdfSettings>)

Creates a PrintPdfContentResponse instance that tells the Chromium engine to proceed with the passed printer.

```
public static PrintPdfContentResponse Print(PdfPrinter<PdfPrinter.IPdfSettings> printer)
```

### Parameters

`printer` PdfPrinter<PdfPrinter.IPdfSettings>

The printer to use.

## Returns

PrintPdfContentResponse

The PrintPdfContentResponse instance that can be used as a return value in PrintPdfContentHandler implementation.

# Namespace DotNetBrowser.Print.Settings

## Classes

CollateExtensions
Extension methods for ICollate<TPrintSettings> interface.

ColorModelExtensions
Extension methods for IColorModel<TPrintSettings> interface.

CopiesExtensions
Extension methods for ICopies<TPrintSettings> interface.

DuplexModeExtensions
Extension methods for IDuplexMode<TPrintSettings> interface.

FitExtensions
Extension methods for IFit<TPrintSettings> interface.

FooterTemplateExtensions
Extension methods for IFooterTemplate<TPrintSettings> interface.

HeaderTemplateExtensions
Extension methods for IHeaderTemplate<TPrintSettings> interface.

OrientationExtensions
Extension methods for IOrientation<TPrintSettings> interface.

PageMarginsExtensions
Extension methods for IPageMargins<TPrintSettings> interface.

PageRangesExtensions
Extension methods for IPageRanges<TPrintSettings> interface.

PagesPerSheetExtensions
Extension methods for IPagesPerSheet<TPrintSettings> interface.

PaperSizeExtensions
Extension methods for IPaperSize<TPrintSettings> interface.

PdfFilePathExtensions
Extension methods for IPdfFilePath<TPrintSettings> interface.

PrintBackgroundsExtensions

Extension methods for IPrintBackgrounds<TPrintSettings> interface.

PrintHeaderFooterExtensions
Extension methods for IPrintHeaderFooter<TPrintSettings> interface.

PrintSelectionOnlyExtensions
Extension methods for IPrintSelectionOnly<TPrintSettings> interface.

PrintSettingsExtensions
Extension methods for IPrintSettings interface.

ScalingExtensions
Extension methods for IScaling<TPrintSettings> interface.

## Interfaces

ICollate<TPrintSettings>
Allows configuring collate printing.

IColorModel<TPrintSettings>
Allows configuring the color model for printing.

ICopies<TPrintSettings>
Allows configuring the number of copies to print.

IDuplexMode<TPrintSettings>
Allows configuring the duplex mode for printing.

IFit<TPrintSettings>
Allows configuring the content fit for printing.

IFooterTemplate<TPrintSettings>
Allows configuring the HTML template for the print footer.

IHeaderTemplate<TPrintSettings>
Allows configuring the HTML template for the print header.

IOrientation<TPrintSettings>
Allows configuring the page orientation.

IPageMargins<TPrintSettings>
Allows configuring the page margins.

IPageRanges<TPrintSettings>
Allows configuring the page ranges for printing.

IPagesPerSheet<TPrintSettings>
Allows configuring the number of pages per sheet.

IPaperSize<TPrintSettings>
Allows configuring the paper size for printing.

IPdfFilePath<TPrintSettings>
Allows configuring the destination PDF file path.

IPrintBackgrounds<TPrintSettings>
Allows configuring printing background graphics.

IPrintHeaderFooter<TPrintSettings>
Allows configuring printing headers and footers.

IPrintSelectionOnly<TPrintSettings>
Allows configuring printing only the selected content.

IPrintSettings
The common interface for the concrete print job settings to implement.

IScaling<TPrintSettings>
Allows configuring the scaling for printing.

# Class CollateExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for ICollate<TPrintSettings> interface.

```
public static class CollateExtensions
```

**Inheritance**

object ← CollateExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

DisableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>)

Disables collate printing.

EnableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>)

Enables collate printing.

# Method DisableCollatePrinting

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## DisableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>)

Disables collate printing.

```
public static TPrintSettings DisableCollatePrinting<TPrintSettings>(this
ICollate<TPrintSettings> collate) where TPrintSettings : class, IPrintSettings
```

## Parameters

`collate` ICollate<TPrintSettings>

The ICollate<TPrintSettings> implementation.

## Returns

TPrintSettings

The current print settings instance.

## Type Parameters

`TPrintSettings`

The specific print settings type.

## Exceptions

ArgumentException

Collate is not supported by the printer.

# Method EnableCollatePrinting

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## EnableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>)

Enables collate printing.

```
public static TPrintSettings EnableCollatePrinting<TPrintSettings>(this
ICollate<TPrintSettings> obj) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` ICollate<TPrintSettings>

The ICollate<TPrintSettings> implementation.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

Collate is not supported by the printer.

# Class ColorModelExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IColorModel<TPrintSettings> interface.

```
public static class ColorModelExtensions
```

**Inheritance**

object ← ColorModelExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetColorModel<TPrintSettings>(IColorModel<TPrintSettings>, ColorModel)

Sets the color model used for printing.

# Method SetColorModel

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetColorModel<TPrintSettings>(IColorModel<TPrintSettings>, ColorModel)

Sets the color model used for printing.

```
public static TPrintSettings SetColorModel<TPrintSettings>(this IColorModel<TPrintSettings>
obj, ColorModel colorModel) where TPrintSettings : class, IPrintSettings
```

## Parameters

`obj` IColorModel<TPrintSettings>

The IColorModel<TPrintSettings> implementation.

`colorModel` ColorModel

The color model used for printing.

## Returns

TPrintSettings

The current print settings instance.

## Type Parameters

`TPrintSettings`

The specific print settings type.

## Exceptions

ArgumentException

The specified color model is not supported by the printer.

# Class CopiesExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for ICopies<TPrintSettings> interface.

```
public static class CopiesExtensions
```

**Inheritance**

object ← CopiesExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetCopies<TPrintSettings>(ICopies<TPrintSettings>, int)

Sets the number of copies to print.

# Method SetCopies

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetCopies<TPrintSettings>(ICopies<TPrintSettings>, int)

Sets the number of copies to print.

```
public static TPrintSettings SetCopies<TPrintSettings>(this ICopies<TPrintSettings> obj,
int copies) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` ICopies<TPrintSettings>

The ICopies<TPrintSettings> implementation.

`copies` int

The number of copies to print. Cannot be zero or negative.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

The specified copies number is less than or equal to zero.

ArgumentException

The specified copies number is not supported by the printer.

# Class DuplexModeExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IDuplexMode<TPrintSettings> interface.

```
public static class DuplexModeExtensions
```

**Inheritance**

object ← DuplexModeExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetDuplexMode<TPrintSettings>(IDuplexMode<TPrintSettings>, DuplexMode)

Sets the duplex mode.

# Method SetDuplexMode

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetDuplexMode<TPrintSettings>(IDuplexMode<TPrintSettings>, DuplexMode)

Sets the duplex mode.

```
public static TPrintSettings SetDuplexMode<TPrintSettings>(this IDuplexMode<TPrintSettings>
obj, DuplexMode duplexMode) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IDuplexMode<TPrintSettings>

The IDuplexMode<TPrintSettings> implementation.

`duplexMode` DuplexMode

The duplex mode used for printing.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

The specified duplex mode is not supported by the printer.

# Class FitExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IFit<TPrintSettings> interface.

```
public static class FitExtensions
```

**Inheritance**

object ← FitExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetFit<TPrintSettings>(IFit<TPrintSettings>, Fit)

Sets the content fit for printing.

# Method SetFit

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetFit<TPrintSettings>(IFit<TPrintSettings>, Fit)

Sets the content fit for printing.

```
public static TPrintSettings SetFit<TPrintSettings>(this IFit<TPrintSettings> obj, Fit fit)
where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IFit<TPrintSettings>

The IFit<TPrintSettings> implementation.

`fit` Fit

The content fit used for printing.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentNullException

The specified fit is null.

ArgumentException

The specified fit is not supported by the printer.

# Class FooterTemplateExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IFooterTemplate<TPrintSettings> interface.

```
public static class FooterTemplateExtensions
```

**Inheritance**

object ← FooterTemplateExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetFooterTemplate<TPrintSettings>(IFooterTemplate<TPrintSettings>, string)

Sets the HTML to be displayed in the footer.

# Method SetFooterTemplate

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetFooterTemplate<TPrintSettings>(IFooterTemplate<TPrintSettings>, string)

Sets the HTML to be displayed in the footer.

```
public static TPrintSettings SetFooterTemplate<TPrintSettings>(this
IFooterTemplate<TPrintSettings> obj, string footerTemplate) where TPrintSettings :
class, IPrintSettings
```

## Parameters

`obj` IFooterTemplate<TPrintSettings>

The IFooterTemplate<TPrintSettings> implementation.

`footerTemplate` string

The HTML to be displayed in the footer. Cannot be null.

## Returns

TPrintSettings

The current print settings instance.

## Type Parameters

`TPrintSettings`

The specific print settings type.

## Exceptions

ArgumentNullException

The specified header template is null.

# Class HeaderTemplateExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IHeaderTemplate<TPrintSettings> interface.

```
public static class HeaderTemplateExtensions
```

**Inheritance**

object ← HeaderTemplateExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetHeaderTemplate<TPrintSettings>(IHeaderTemplate<TPrintSettings>, string)

Sets the HTML to be displayed in the header.

# Method SetHeaderTemplate

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetHeaderTemplate<TPrintSettings>(IHeaderTemplate<TPrintSettings>, string)

Sets the HTML to be displayed in the header.

```
public static TPrintSettings SetHeaderTemplate<TPrintSettings>(this
IHeaderTemplate<TPrintSettings> obj, string headerTemplate) where TPrintSettings :
class, IPrintSettings
```

### Parameters

`obj` IHeaderTemplate<TPrintSettings>

The IHeaderTemplate<TPrintSettings> implementation.

`headerTemplate` string

The HTML to be displayed in the header. Cannot be null.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentNullException

The specified header template is null.

# Interface ICollate<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring collate printing.

```
public interface ICollate<out TPrintSettings> where TPrintSettings : class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
CollateExtensions.DisableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>) ,
CollateExtensions.EnableCollatePrinting<TPrintSettings>(ICollate<TPrintSettings>)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support collate printing.

## Properties

CollatePrintingEnabled

Enables or disables collate printing.

# Property CollatePrintingEnabled

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## CollatePrintingEnabled

Enables or disables collate printing.

```
bool CollatePrintingEnabled { get; set; }
```

## Property Value

bool

`true` if collate printing is enabled, `false` otherwise.

## Exceptions

ArgumentException

Collate is not supported by the printer.

# Interface IColorModel<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the color model for printing.

```
public interface IColorModel<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**

ColorModelExtensions.SetColorModel<TPrintSettings>(IColorModel<TPrintSettings>, ColorModel)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support color models.

## Properties

ColorModel

Gets or sets the color model.

# Property ColorModel

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## ColorModel

Gets or sets the color model.

```
ColorModel ColorModel { get; set; }
```

## Property Value

ColorModel

The color model used for printing.

## Exceptions

ArgumentException

The specified color model is not supported by the printer.

# Interface ICopies<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the number of copies to print.

```
public interface ICopies<out TPrintSettings> where TPrintSettings : class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**

CopiesExtensions.SetCopies<TPrintSettings>(ICopies<TPrintSettings>, int)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support specifying copies.

## Properties

Copies

Gets or sets the number of copies to print.

# Property Copies

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## Copies

Gets or sets the number of copies to print.

```
int Copies { get; set; }
```

## Property Value

int

The number of copies to print. Cannot be zero or negative.

## Exceptions

ArgumentException

The specified copies number is less than or equal to zero.

ArgumentException

The specified copies number is not supported by the printer.

# Interface IDuplexMode<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the duplex mode for printing.

```
public interface IDuplexMode<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
DuplexModeExtensions.SetDuplexMode<TPrintSettings>(IDuplexMode<TPrintSettings>, DuplexMode)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support specifying duplex mode.

## Properties

DuplexMode

Gets or sets the duplex mode.

# Property DuplexMode

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## DuplexMode

Gets or sets the duplex mode.

```
DuplexMode DuplexMode { get; set; }
```

## Property Value

DuplexMode

The duplex mode used for printing.

## Exceptions

ArgumentException

The specified duplex mode is not supported by the printer.

# Interface IFit<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the content fit for printing.

```
public interface IFit<out TPrintSettings> where TPrintSettings : class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
FitExtensions.SetFit<TPrintSettings>(IFit<TPrintSettings>, Fit)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring the content fit for printing.

## Properties

Fit

Gets or sets the content fit for printing.

# Property Fit

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## Fit

Gets or sets the content fit for printing.

```
Fit Fit { get; set; }
```

## Property Value

Fit

The fit used for printing.

## Exceptions

ArgumentNullException

The specified fit is null.

ArgumentException

The specified fit is not supported by the printer.

# Interface IFooterTemplate<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the HTML template for the print footer.

```
public interface IFooterTemplate<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**

FooterTemplateExtensions.SetFooterTemplate<TPrintSettings>(IFooterTemplate<TPrintSettings>, string)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring the footer for printing.

## Properties

FooterTemplate

Gets or sets the HTML to be displayed in the footer.

# Property FooterTemplate

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## FooterTemplate

Gets or sets the HTML to be displayed in the footer.

```
string FooterTemplate { get; set; }
```

## Property Value

string

The footer template. Cannot be null.

## Remarks

Apply the following CSS classes to insert printing metadata into the template. These classes don't affect the visual appearance of the elements.

- `date`: the formatted print date;
- `title`: the document title;
- `url`: the document location;
- `pageNumber`: the current page number;
- `totalPages`: the total page count in the document;

For example,

```

```

would generate a span containing a title.
The programming, audio/video, and frame tags are not supported. Images are supported only with Base64 content.

## Exceptions

ArgumentNullException

The specified footer template is null.

# Interface IHeaderTemplate<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the HTML template for the print header.

```
public interface IHeaderTemplate<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**

HeaderTemplateExtensions.SetHeaderTemplate<TPrintSettings>(IHeaderTemplate<TPrintSettings>, string)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring the header for printing.

## Properties

HeaderTemplate

Gets or sets the HTML to be displayed in the header.

# Property HeaderTemplate

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## HeaderTemplate

Gets or sets the HTML to be displayed in the header.

```
string HeaderTemplate { get; set; }
```

## Property Value

string

The header template. Cannot be null.

## Remarks

Apply the following CSS classes to insert printing metadata into the template. These classes don't affect the visual appearance of the elements.

- `date`: the formatted print date;
- `title`: the document title;
- `url`: the document location;
- `pageNumber`: the current page number;
- `totalPages`: the total page count in the document;

For example,

```

```

would generate a span containing a title.
The programming, audio/video, and frame tags are not supported. Images are supported only with Base64 content.

## Exceptions

ArgumentNullException

The specified header is null.

# Interface IOrientation<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the page orientation.

```
public interface IOrientation<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**

OrientationExtensions.SetOrientation<TPrintSettings>(IOrientation<TPrintSettings>, Orientation)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring the page orientation.

## Properties

Orientation

Gets or sets the page orientation.

# Property Orientation

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## Orientation

Gets or sets the page orientation.

```
Orientation Orientation { get; set; }
```

## Property Value

Orientation

The page orientation.

## Exceptions

ArgumentException

The specified page orientation is not supported by the printer.

# Interface IPageMargins<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the page margins.

```
public interface IPageMargins<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**

PageMarginsExtensions.SetPageMargins<TPrintSettings>(IPageMargins<TPrintSettings>, PageMargins)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring the page margins.

## Properties

PageMargins

Gets or sets the page margins for printing.

# Property PageMargins

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## PageMargins

Gets or sets the page margins for printing.

```
PageMargins PageMargins { get; set; }
```

## Property Value

PageMargins

The page margins for printing.

## Exceptions

ArgumentException

The specified page margins is null.

ArgumentException

The specified page margins is not supported by the printer.

# Interface IPageRanges<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the page ranges for printing.

```
public interface IPageRanges<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**

PageRangesExtensions.SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support specifying page ranges.

## Properties

PageRanges

Gets or sets the collection of the page ranges to print.

# Property PageRanges

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## PageRanges

Gets or sets the collection of the page ranges to print.

```
IReadOnlyCollection<PageRange> PageRanges { get; set; }
```

## Property Value

IReadOnlyCollection<PageRange>

The collection of the page ranges to print.

## Exceptions

ArgumentException

The collection is null, empty, or contains null objects.

# Interface IPagesPerSheet<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the number of pages per sheet.

```
public interface IPagesPerSheet<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
PagesPerSheetExtensions.SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring the number of pages per sheet.

## Properties

PagesPerSheet

Gets or sets the number of pages per sheet.

# Property PagesPerSheet

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## PagesPerSheet

Gets or sets the number of pages per sheet.

```
PagesPerSheet PagesPerSheet { get; set; }
```

## Property Value

PagesPerSheet

The number of pages per sheet.

## Exceptions

ArgumentException

The specified number of pages per sheet is not supported by the printer.

# Interface IPaperSize<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the paper size for printing.

```
public interface IPaperSize<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**

PaperSizeExtensions.SetPaperSize<TPrintSettings>(IPaperSize<TPrintSettings>, PaperSize)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring paper size.

## Properties

PaperSize

Gets or sets the paper size for printing.

# Property PaperSize

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## PaperSize

Gets or sets the paper size for printing.

```
PaperSize PaperSize { get; set; }
```

## Property Value

PaperSize

The paper size used for printing.

## Exceptions

ArgumentException

The specified paper size is null.

# Interface IPdfFilePath<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the destination PDF file path.

```
public interface IPdfFilePath<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
PdfFilePathExtensions.SetPdfFilePath<TPrintSettings>(IPdfFilePath<TPrintSettings>, string)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring the destination PDF file path.

## Properties

PdfFilePath
Gets or sets the full PDF file path.

# Property PdfFilePath

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## PdfFilePath

Gets or sets the full PDF file path.

```
string PdfFilePath { get; set; }
```

## Property Value

string

The full PDF file path. Cannot be null, empty, or contain only white space.

## Exceptions

ArgumentException

The specified file path is null, empty, or contain only white space.

ArgumentException

The parent directory for the PDF file does not exist and cannot be created.

# Interface IPrintBackgrounds<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring printing background graphics.

```
public interface IPrintBackgrounds<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
PrintBackgroundsExtensions.DisablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>) ,
PrintBackgroundsExtensions.EnablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring printing background graphics.

## Properties

PrintingBackgroundsEnabled

Enables or disables printing background graphics.

# Property PrintingBackgroundsEnabled

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## PrintingBackgroundsEnabled

Enables or disables printing background graphics.

```
bool PrintingBackgroundsEnabled { get; set; }
```

## Property Value

bool

`true` if printing background graphics is enabled, `false` otherwise.

## Exceptions

ArgumentException

The background graphics printing is not supported by the printer.

# Interface IPrintHeaderFooter<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring printing headers and footers.

```
public interface IPrintHeaderFooter<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
PrintHeaderFooterExtensions.DisablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>) ,
PrintHeaderFooterExtensions.EnablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring printing headers and footers.

## Properties

PrintingHeaderFooterEnabled

Enables or disables printing headers and footer.

# Property PrintingHeaderFooterEnabled

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## PrintingHeaderFooterEnabled

Enables or disables printing headers and footer.

```
bool PrintingHeaderFooterEnabled { get; set; }
```

## Property Value

bool

`true` if printing headers and footer is enabled, `false` otherwise.

## Exceptions

ArgumentException

The headers and footer printing is not supported by the printer.

# Interface IPrintSelectionOnly<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring printing only the selected content.

```
public interface IPrintSelectionOnly<out TPrintSettings> where TPrintSettings :
class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
PrintSelectionOnlyExtensions.DisablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>) ,
PrintSelectionOnlyExtensions.EnablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring printing only the selected content.

## Properties

PrintingSelectionOnlyEnabled

Enables or disables printing only the selected content.

# Property PrintingSelectionOnlyEnabled

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## PrintingSelectionOnlyEnabled

Enables or disables printing only the selected content.

```
bool PrintingSelectionOnlyEnabled { get; set; }
```

## Property Value

bool

`true` if printing only the selected content is enabled, `false` otherwise.

## Exceptions

ArgumentException

Only the selected content printing is not supported by the printer.

# Interface IPrintSettings

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

The common interface for the concrete print job settings to implement.

```
public interface IPrintSettings
```

**Extension Methods**

PrintSettingsExtensions.Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>)

## Methods

Apply()

Applies the configured print settings. You should call this method to regenerate the internal print preview using the configured settings and update the total page count to be printed.

# Method Apply

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## Apply()

Applies the configured print settings. You should call this method to regenerate the internal print preview using the configured settings and update the total page count to be printed.

```
void Apply()
```

## Remarks

This method is invoked automatically when you choose to proceed in the PrintHtmlContentHandler or PrintPdfContentHandler. If the settings cannot be applied at this point, the printing operation is canceled.

## Exceptions

ObjectDisposedException

The IPrintJob<TPrintSettings> has already been disposed.

InvalidOperationException

The print settings were not applied properly.

# Interface IScaling<TPrintSettings>

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Allows configuring the scaling for printing.

```
public interface IScaling<out TPrintSettings> where TPrintSettings : class, IPrintSettings
```

## Type Parameters

`TPrintSettings`

The specific print settings type.

**Extension Methods**
ScalingExtensions.SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling)

## Remarks

This interface is implemented by specific IPrintSettings implementations that support configuring the scaling for printing.

## Properties

Scaling

Gets or sets the scaling for printing.

# Property Scaling

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## Scaling

Gets or sets the scaling for printing.

```
Scaling Scaling { get; set; }
```

## Property Value

Scaling

The scaling used for printing.

## Exceptions

ArgumentNullException

The specified scaling is null.

ArgumentException

The specified scaling is not supported by the printer.

# Class OrientationExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IOrientation<TPrintSettings> interface.

```
public static class OrientationExtensions
```

**Inheritance**

object ← OrientationExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetOrientation<TPrintSettings>(IOrientation<TPrintSettings>, Orientation)

Sets the page orientation.

# Method SetOrientation

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetOrientation<TPrintSettings>(IOrientation<TPrintSettings>, Orientation)

Sets the page orientation.

```
public static TPrintSettings SetOrientation<TPrintSettings>(this
IOrientation<TPrintSettings> obj, Orientation orientation) where TPrintSettings :
class, IPrintSettings
```

## Parameters

`obj` IOrientation<TPrintSettings>

The IOrientation<TPrintSettings> implementation.

`orientation` Orientation

The page orientation.

## Returns

TPrintSettings

The current print settings instance.

## Type Parameters

`TPrintSettings`

The specific print settings type.

## Exceptions

ArgumentException

The specified page orientation is not supported by the printer.

# Class PageMarginsExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPageMargins<TPrintSettings> interface.

```
public static class PageMarginsExtensions
```

**Inheritance**

object ← PageMarginsExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetPageMargins<TPrintSettings>(IPageMargins<TPrintSettings>, PageMargins)

Configures the page margins for printing

# Method SetPageMargins

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetPageMargins<TPrintSettings>(IPageMargins<TPrintSettings>, PageMargins)

Configures the page margins for printing

```
public static TPrintSettings SetPageMargins<TPrintSettings>(this
IPageMargins<TPrintSettings> obj, PageMargins pageMargins) where TPrintSettings :
class, IPrintSettings
```

## Parameters

`obj` IPageMargins<TPrintSettings>

The IPageMargins<TPrintSettings> implementation.

`pageMargins` PageMargins

The page margins for printing.

## Returns

TPrintSettings

The current print settings instance.

## Type Parameters

`TPrintSettings`

The specific print settings type.

## Exceptions

ArgumentException

The specified page margins is null.

ArgumentException

The specified page margins is not supported by the printer.

# Class PageRangesExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPageRanges<TPrintSettings> interface.

```
public static class PageRangesExtensions
```

**Inheritance**

object ← PageRangesExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>)

Sets the collection of the page ranges to print.

# Method SetPageRanges

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetPageRanges<TPrintSettings>(IPageRanges<TPrintSettings>, IReadOnlyCollection<PageRange>)

Sets the collection of the page ranges to print.

```
public static TPrintSettings SetPageRanges<TPrintSettings>(this IPageRanges<TPrintSettings>
obj, IReadOnlyCollection<PageRange> pageRanges) where TPrintSettings :
class, IPrintSettings
```

### Parameters

`obj` IPageRanges<TPrintSettings>

The IPageRanges<TPrintSettings> implementation.

`pageRanges` IReadOnlyCollection<PageRange>

The collection of the page ranges to print.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

The collection is null, empty, or contains null objects.

# Class PagesPerSheetExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPagesPerSheet<TPrintSettings> interface.

```
public static class PagesPerSheetExtensions
```

**Inheritance**

object ← PagesPerSheetExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet)

Sets the number of pages per sheet.

# Method SetPagesPerSheet

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetPagesPerSheet<TPrintSettings>(IPagesPerSheet<TPrintSettings>, PagesPerSheet)

Sets the number of pages per sheet.

```
public static TPrintSettings SetPagesPerSheet<TPrintSettings>(this
IPagesPerSheet<TPrintSettings> obj, PagesPerSheet pagesPerSheet) where TPrintSettings :
class, IPrintSettings
```

### Parameters

`obj` IPagesPerSheet<TPrintSettings>

The IPagesPerSheet<TPrintSettings> implementation.

`pagesPerSheet` PagesPerSheet

The new number of pages per sheet.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

The specified number of pages per sheet is not supported by the printer.

# Class PaperSizeExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPaperSize<TPrintSettings> interface.

```
public static class PaperSizeExtensions
```

**Inheritance**

object ← PaperSizeExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetPaperSize<TPrintSettings>(IPaperSize<TPrintSettings>, PaperSize)

Sets the paper size for printing.

# Method SetPaperSize

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetPaperSize<TPrintSettings>(IPaperSize<TPrintSettings>, PaperSize)

Sets the paper size for printing.

```
public static TPrintSettings SetPaperSize<TPrintSettings>(this IPaperSize<TPrintSettings>
obj, PaperSize paperSize) where TPrintSettings : class, IPrintSettings
```

## Parameters

`obj` IPaperSize<TPrintSettings>

The IPaperSize<TPrintSettings> implementation.

`paperSize` PaperSize

The paper size used for printing.

## Returns

TPrintSettings

The current print settings instance.

## Type Parameters

`TPrintSettings`

The specific print settings type.

## Exceptions

ArgumentException

The specified paper size is null.

# Class PdfFilePathExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPdfFilePath<TPrintSettings> interface.

```
public static class PdfFilePathExtensions
```

**Inheritance**

object ← PdfFilePathExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetPdfFilePath<TPrintSettings>(IPdfFilePath<TPrintSettings>, string)

Sets the full PDF file path.

# Method SetPdfFilePath

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetPdfFilePath<TPrintSettings>(IPdfFilePath<TPrintSettings>, string)

Sets the full PDF file path.

```
public static TPrintSettings SetPdfFilePath<TPrintSettings>(this
IPdfFilePath<TPrintSettings> obj, string pdfFilePath) where TPrintSettings :
class, IPrintSettings
```

## Parameters

`obj` IPdfFilePath<TPrintSettings>

The IPdfFilePath<TPrintSettings> implementation.

`pdfFilePath` string

The full PDF file path. Cannot be null, empty, or contain only white space.

## Returns

TPrintSettings

The current print settings instance.

## Type Parameters

`TPrintSettings`

The specific print settings type.

## Exceptions

ArgumentException

The specified file path is null, empty, or contain only white space.

ArgumentException

The parent directory for the PDF file does not exist and cannot be created.

# Class PrintBackgroundsExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPrintBackgrounds<TPrintSettings> interface.

```
public static class PrintBackgroundsExtensions
```

**Inheritance**

object ← PrintBackgroundsExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

DisablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>)

Disables printing background graphics.

EnablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>)

Enables printing background graphics.

# Method DisablePrintingBackgrounds

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## DisablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>)

Disables printing background graphics.

```
public static TPrintSettings DisablePrintingBackgrounds<TPrintSettings>(this
IPrintBackgrounds<TPrintSettings> obj) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IPrintBackgrounds<TPrintSettings>

The IPrintBackgrounds<TPrintSettings> implementation.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

The background graphics printing is not supported by the printer.

# Method EnablePrintingBackgrounds

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## EnablePrintingBackgrounds<TPrintSettings>(IPrintBackgrounds<TPrintSettings>)

Enables printing background graphics.

```
public static TPrintSettings EnablePrintingBackgrounds<TPrintSettings>(this
IPrintBackgrounds<TPrintSettings> obj) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IPrintBackgrounds<TPrintSettings>

The IPrintBackgrounds<TPrintSettings> implementation.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

The background graphics printing is not supported by the printer.

# Class PrintHeaderFooterExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPrintHeaderFooter<TPrintSettings> interface.

```
public static class PrintHeaderFooterExtensions
```

**Inheritance**

object ← PrintHeaderFooterExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

DisablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>)

Disables printing headers and footers.

EnablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>)

Enables printing headers and footers.

# Method DisablePrintingHeaderFooter

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## DisablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>)

Disables printing headers and footers.

```
public static TPrintSettings DisablePrintingHeaderFooter<TPrintSettings>(this
IPrintHeaderFooter<TPrintSettings> obj) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IPrintHeaderFooter<TPrintSettings>

The IPrintHeaderFooter<TPrintSettings> implementation.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

The headers and footer printing is not supported by the printer.

# Method EnablePrintingHeaderFooter

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## EnablePrintingHeaderFooter<TPrintSettings>(IPrintHeaderFooter<TPrintSettings>)

Enables printing headers and footers.

```
public static TPrintSettings EnablePrintingHeaderFooter<TPrintSettings>(this
IPrintHeaderFooter<TPrintSettings> obj) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IPrintHeaderFooter<TPrintSettings>

The IPrintHeaderFooter<TPrintSettings> implementation.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

The headers and footer printing is not supported by the printer.

# Class PrintSelectionOnlyExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPrintSelectionOnly<TPrintSettings> interface.

```
public static class PrintSelectionOnlyExtensions
```

**Inheritance**

object ← PrintSelectionOnlyExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

DisablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>)

Disables printing only the selected content.

EnablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>)

Enables printing only the selected content.

# Method DisablePrintingSelectionOnly

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## DisablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>)

Disables printing only the selected content.

```
public static TPrintSettings DisablePrintingSelectionOnly<TPrintSettings>(this
IPrintSelectionOnly<TPrintSettings> obj) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IPrintSelectionOnly<TPrintSettings>

The IPrintSelectionOnly<TPrintSettings> implementation.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

Only the selected content printing is not supported by the printer.

# Method EnablePrintingSelectionOnly

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## EnablePrintingSelectionOnly<TPrintSettings>(IPrintSelectionOnly<TPrintSettings>)

Enables printing only the selected content.

```
public static TPrintSettings EnablePrintingSelectionOnly<TPrintSettings>(this
IPrintSelectionOnly<TPrintSettings> obj) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IPrintSelectionOnly<TPrintSettings>

The IPrintSelectionOnly<TPrintSettings> implementation.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentException

Only the selected content printing is not supported by the printer.

# Class PrintSettingsExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IPrintSettings interface.

```
public static class PrintSettingsExtensions
```

**Inheritance**

object ← PrintSettingsExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>)

Performs the action with the print settings and then applies the changes.

# Method Apply

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## Apply<TPrintSettings>(TPrintSettings, Action<TPrintSettings>)

Performs the action with the print settings and then applies the changes.

```
public static void Apply<TPrintSettings>(this TPrintSettings printSettings,
Action<TPrintSettings> action) where TPrintSettings : class, IPrintSettings
```

### Parameters

`printSettings` TPrintSettings

The print settings to work with. Cannot be null.

`action` Action<TPrintSettings>

The action to perform with these print settings. Cannot be null.

### Type Parameters

`TPrintSettings`

The type of print settings that can be applied.

### Exceptions

ObjectDisposedException

The IPrintJob<TPrintSettings> has already been disposed.

InvalidOperationException

The print settings were not applied properly.

# Class ScalingExtensions

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

Extension methods for IScaling<TPrintSettings> interface.

```
public static class ScalingExtensions
```

**Inheritance**

object ← ScalingExtensions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Methods

SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling)

Sets the scaling for printing.

# Method SetScaling

Namespace: DotNetBrowser.Print.Settings

Assembly: DotNetBrowser.dll

## SetScaling<TPrintSettings>(IScaling<TPrintSettings>, Scaling)

Sets the scaling for printing.

```
public static TPrintSettings SetScaling<TPrintSettings>(this IScaling<TPrintSettings> obj,
Scaling scaling) where TPrintSettings : class, IPrintSettings
```

### Parameters

`obj` IScaling<TPrintSettings>

The IScaling<TPrintSettings> implementation.

`scaling` Scaling

The scaling used for printing.

### Returns

TPrintSettings

The current print settings instance.

### Type Parameters

`TPrintSettings`

The specific print settings type.

### Exceptions

ArgumentNullException

The specified scaling is null.

ArgumentException

The specified scaling is not supported by the printer.

# Namespace DotNetBrowser.Profile

## Interfaces

IProfile
: The Chromium profile.

IProfilePreferences
: The preferences of a IProfile.

IProfiles
: The collection of the Chromium profiles.

## Enums

CookieControlsMode
: The Chromium third-party cookies block mode.

ProfileType
: The Chromium profile type.

# Enum CookieControlsMode

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

The Chromium third-party cookies block mode.

```
public enum CookieControlsMode
```

## Fields

`BlockThirdParty = 2`

Sites can't use user cookies to see user browsing activity across different sites, for example, to personalize ads. Features on some sites may not work.

`IncognitoOnly = 3`

While in Incognito, sites can't use user cookies to see user browsing activity across sites, even related sites. The user browsing activity isn't used for things like personalizing ads. Features on some sites may not work.

`Off = 1`

Sites can use cookies to see the user browsing activity across different sites, for example, to personalize ads.

`Undefined = 0`

Reserved value.

# Interface IProfile

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

The Chromium profile.

```
public interface IProfile : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

CookieStore
Gets the cookie service that allows managing cookies.

CreditCardStore
Gets the service that allows working with the credit card store.

Downloads
Gets the service that allows managing downloads.

Engine
Gets the IEngine instance associated with this object.

Extensions
Gets the service that allows managing extensions.

HttpAuthCache
Gets the HTTP Authentication cache service, which is responsible for storing HTTP authentication identities and challenge info

HttpCache
Gets the HTTP cache service.

IsDefault
Indicates whether this profile is default.

MediaCasting

Gets the service that allows working with media casting.

Name
Gets the profile name.

Network
Gets the service that allows working with network.

PasswordStore
Gets the service that allows working with the password store.

Path
Gets the profile path.

Permissions
Gets the service that allows managing permissions.

Plugins
Gets the service that allows configuring plugins.

Preferences
Gets the profile preferences.

Proxy
Gets the service that allows working with proxy.

SpellChecker
Gets the service that allows working with spell checking functionality.

Type
Gets the profile type.

UserDataProfileStore
Gets the service that allows working with the user date profile store.

ZoomLevels
Gets the service that allows working with zoom.

## Methods

CreateBrowser()
Creates a new IBrowser instance with the initial "about:blank" web page.

# Property CookieStore

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## CookieStore

Gets the cookie service that allows managing cookies.

```
ICookieStore CookieStore { get; }
```

## Property Value

ICookieStore

The cookie service that allows managing cookies.

# Property CreditCardStore

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## CreditCardStore

Gets the service that allows working with the credit card store.

```
ICreditCardStore CreditCardStore { get; }
```

## Property Value

ICreditCardStore

The service that allows working with the credit card store.

# Property Downloads

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Downloads

Gets the service that allows managing downloads.

```
IDownloads Downloads { get; }
```

## Property Value

IDownloads

The service that allows managing downloads.

# Property Engine

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Extensions

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Extensions

Gets the service that allows managing extensions.

```
IExtensions Extensions { get; }
```

## Property Value

IExtensions

The service that allows managing extensions.

# Property HttpAuthCache

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## HttpAuthCache

Gets the HTTP Authentication cache service, which is responsible for storing HTTP authentication identities and challenge info

```
IHttpAuthCache HttpAuthCache { get; }
```

## Property Value

IHttpAuthCache

The HTTP Authentication cache service.

# Property HttpCache

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## HttpCache

Gets the HTTP cache service.

```
IHttpCache HttpCache { get; }
```

## Property Value

IHttpCache

The HTTP cache service.

# Property IsDefault

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## IsDefault

Indicates whether this profile is default.

```
bool IsDefault { get; }
```

## Property Value

bool

`true` if this profile is default, `false` otherwise.

# Property MediaCasting

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## MediaCasting

Gets the service that allows working with media casting.

```
IMediaCasting MediaCasting { get; }
```

## Property Value

IMediaCasting

The service that allows working with media casting.

# Property Name

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Name

Gets the profile name.

```
string Name { get; }
```

## Property Value

string

The profile name configured during its creation.

# Property Network

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Network

Gets the service that allows working with network.

```
INetwork Network { get; }
```

## Property Value

INetwork

The service that allows working with network.

# Property PasswordStore

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## PasswordStore

Gets the service that allows working with the password store.

```
IPasswordStore PasswordStore { get; }
```

## Property Value

IPasswordStore

The service that allows working with the password store.

# Property Path

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Path

Gets the profile path.

```
string Path { get; }
```

## Property Value

string

The path that is used to store all the profile data.

# Property Permissions

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Permissions

Gets the service that allows managing permissions.

```
IPermissions Permissions { get; }
```

## Property Value

IPermissions

The service that allows managing permissions.

# Property Plugins

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Plugins

Gets the service that allows configuring plugins.

```
IPlugins Plugins { get; }
```

## Property Value

IPlugins

The service that allows configuring plugins.

# Property Preferences

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Preferences

Gets the profile preferences.

```
IProfilePreferences Preferences { get; }
```

## Property Value

IProfilePreferences

The preferences for this profile.

# Property Proxy

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Proxy

Gets the service that allows working with proxy.

```
IProxy Proxy { get; }
```

## Property Value

IProxy

The service that allows working with proxy.

# Property SpellChecker

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## SpellChecker

Gets the service that allows working with spell checking functionality.

```
ISpellChecker SpellChecker { get; }
```

## Property Value

ISpellChecker

The service that allows working with spell checking functionality.

# Property Type

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Type

Gets the profile type.

```
ProfileType Type { get; }
```

## Property Value

ProfileType

The profile type - regular or incognito.

# Property UserDataProfileStore

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## UserDataProfileStore

Gets the service that allows working with the user date profile store.

```
IUserDataProfileStore UserDataProfileStore { get; }
```

## Property Value

IUserDataProfileStore

The service that allows working with the user date profile store.

# Property ZoomLevels

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## ZoomLevels

Gets the service that allows working with zoom.

```
IZoomLevels ZoomLevels { get; }
```

## Property Value

IZoomLevels

The service that allows working with zoom.

# Method CreateBrowser

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## CreateBrowser()

Creates a new IBrowser instance with the initial "about:blank" web page.

```
IBrowser CreateBrowser()
```

## Returns

IBrowser

a new IBrowser instance.

# Interface IProfilePreferences

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

The preferences of a IProfile.

```
public interface IProfilePreferences : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

AutofillEnabled
: Enables or disables the web form autofill and displaying of suggestions pop-ups. By default, the autofill is enabled.

CaretBrowsingEnabled
: Enables or disables the caret browsing. By default, the caret browsing is disabled.

Engine
: Gets the IEngine instance associated with this object.

NetworkPredictionEnabled
: Enables or disables the network prediction. By default, the network prediction is enabled.

Profile
: Gets the IProfile instance associated with this object.

ThirdPartyCookieMode
: Gets or sets the third-party cookies blocking mode.

# Property AutofillEnabled

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## AutofillEnabled

Enables or disables the web form autofill and displaying of suggestions pop-ups. By default, the autofill is enabled.

```
bool AutofillEnabled { get; set; }
```

## Property Value

bool

`true` if the web form autofill is enabled for the simple `<input>` values, `false` otherwise.

## Remarks

**Important:** the autofill for the IPasswordStore passwords is always enabled except the cases when a site has an invalid SSL certificate or the scheme is not registered as web-safe.The list of web-safe schemes: HTTP, HTTPS, WS, and WSS.

## Exceptions

ObjectDisposedException

The IProfilePreferences instance has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property CaretBrowsingEnabled

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## CaretBrowsingEnabled

Enables or disables the caret browsing. By default, the caret browsing is disabled.

```
bool CaretBrowsingEnabled { get; set; }
```

## Property Value

bool

`true` if the caret browsing is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The IProfilePreferences instance has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Engine

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property NetworkPredictionEnabled

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## NetworkPredictionEnabled

Enables or disables the network prediction. By default, the network prediction is enabled.

```
bool NetworkPredictionEnabled { get; set; }
```

## Property Value

bool

`true` if the network prediction is enabled, `false` otherwise.

## Remarks

Network prediction is the built-in feature of Chromium that allows you to open the most visited websites faster. This is done by DNS prefetching, TCP and SSL preconnection, prerendering of web pages, and resource prefetching.

## Exceptions

ObjectDisposedException

The IProfilePreferences instance has already been disposed.

ConnectionClosedException

The connection to the Chromium engine is closed.

# Property Profile

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Property ThirdPartyCookieMode

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## ThirdPartyCookieMode

Gets or sets the third-party cookies blocking mode.

```
CookieControlsMode ThirdPartyCookieMode { get; set; }
```

## Property Value

CookieControlsMode

# Interface IProfiles

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

The collection of the Chromium profiles.

```
public interface IProfiles : IEnumerable<IProfile>, IEnumerable, IAutoDisposable
```

**Inherited Members**

IEnumerable<IProfile>.GetEnumerator() , IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Count

Gets the current number of profiles.

Default

Gets the default profile used by the Chromium engine.

Engine

Gets the IEngine instance associated with this object.

## Methods

Create(string, ProfileType)

Creates a new Chromium profile of the specified `profileType` with the specified `name`.

Remove(IProfile)

Removes the existing Chromium profile.

# Property Count

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Count

Gets the current number of profiles.

```
int Count { get; }
```

## Property Value

int

The number of profiles.

# Property Default

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Default

Gets the default profile used by the Chromium engine.

```
IProfile Default { get; }
```

## Property Value

IProfile

The default profile used by the Chromium engine.

# Property Engine

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Method Create

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Create(string, ProfileType)

Creates a new Chromium profile of the specified `profileType` with the specified `name`.

```
IProfile Create(string name, ProfileType profileType = ProfileType.Regular)
```

## Parameters

`name` string

The profile name. Cannot be null or empty.

`profileType` ProfileType

The profile type. If omitted, the regular profile will be created.

## Returns

IProfile

The new Chromium profile.

# Method Remove

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

## Remove(IProfile)

Removes the existing Chromium profile.

```
bool Remove(IProfile profile)
```

### Parameters

`profile` IProfile

The Chromium profile to remove

### Returns

bool

`true` if the profile was removed, `false` otherwise.

### Remarks

When the profile is removed, the corresponding IBrowser instances are disposed. The downloads in progress are canceled, and the profile directory is scheduled for deletion.

### Exceptions

ArgumentException

The profile belongs to a different IEngine instance.

# Enum ProfileType

Namespace: DotNetBrowser.Profile

Assembly: DotNetBrowser.dll

The Chromium profile type.

```
public enum ProfileType
```

## Fields

`Incognito = 1`

The incognito Chromium profile.

`Regular = 0`

The regular Chromium profile.

# Namespace DotNetBrowser.Search

## Classes

FindOptions
: The find text options.

FindResult
: A result of the text search.

## Interfaces

ITextFinder
: Allows finding text on the loaded web page.

## Enums

Direction
: The search direction.

StopFindAction
: The stop find actions.

# Enum Direction

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

The search direction.

```
public enum Direction
```

## Fields

`Backward = 2`

Search backward.

`Forward = 1`

Search forward.

# Class FindOptions

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

The find text options.

```
public class FindOptions
```

**Inheritance**

object ← FindOptions

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

FindOptions(bool, Direction)

Initializes a new instance of FindOptions with the specified `matchCase` and `direction`.

## Properties

Direction

Gets or sets the search direction.

MatchCase

Enables or disables the case-sensitive search.

# Constructor FindOptions

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

## FindOptions(bool, Direction)

Initializes a new instance of FindOptions with the specified `matchCase` and `direction`.

```
public FindOptions(bool matchCase = false, Direction direction = Direction.Forward)
```

## Parameters

`matchCase` bool

Indicates whether the search is case-sensitive.

`direction` Direction

The search direction.

# Property Direction

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

## Direction

Gets or sets the search direction.

```
public Direction Direction { get; set; }
```

## Property Value

Direction

The search direction.

# Property MatchCase

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

## MatchCase

Enables or disables the case-sensitive search.

```
public bool MatchCase { get; set; }
```

## Property Value

bool

`true` if the case-sensitive search is enabled, `false` otherwise.

# Class FindResult

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

A result of the text search.

```
public class FindResult
```

**Inheritance**

object ← FindResult

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

NumberOfMatches

Gets the number of matches.

SelectedMatch

Gets the index of the currently selected match.

# Property NumberOfMatches

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

## NumberOfMatches

Gets the number of matches.

```
public int NumberOfMatches { get; }
```

## Property Value

int

The number of the matches found at the moment or -1 when no matches found.

# Property SelectedMatch

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

## SelectedMatch

Gets the index of the currently selected match.

```
public int SelectedMatch { get; }
```

## Property Value

int

The index of the currently selected match or -1 when there's no matches.

# Interface ITextFinder

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

Allows finding text on the loaded web page.

```
public interface ITextFinder : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Browser
: Gets the IBrowser instance associated with this object.

## Methods

Find(string, FindOptions, IHandler<FindResultReceivedParameters>)
: Performs search of the given `searchText` with the given `options`, highlights all matches and selects the first match on the currently loaded web page.

StopFinding(StopFindAction)
: Stops finding and clears the highlighting of the found matches.

# Property Browser

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with this object.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with this object.

# Method Find

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

## Find(string, FindOptions, IHandler<FindResultReceivedParameters>)

Performs search of the given `searchText` with the given `options`, highlights all matches and selects the first match on the currently loaded web page.

```
Task<FindResult> Find(string searchText, FindOptions options = null,
IHandler<FindResultReceivedParameters> intermediateResultsHandler = null)
```

### Parameters

`searchText` string

the text to search. Cannot be null or empty.

`options` FindOptions

the options such as direction and match case.

`intermediateResultsHandler` IHandler<FindResultReceivedParameters>

the handler which is called when some search results are received.

### Returns

Task<FindResult>

a Task which completes when the search is finished. The result of the task is the last find result received from Chromium.

### Remarks

The search is performed only through a visible content on the loaded web page. If some text is presented on the web page, but due to CSS rules it's not visible, the text finder will not check this content

during search. Also, it doesn't find text on the web pages with an empty size, so make sure that the size of the browser instance where the text search is performed isn't empty.

## Exceptions

ObjectDisposedException

The ITextFinder has already been disposed.

# Method StopFinding

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

## StopFinding(StopFindAction)

Stops finding and clears the highlighting of the found matches.

```
void StopFinding(StopFindAction action = StopFindAction.ClearSelection)
```

## Parameters

`action` StopFindAction

The action that will be applied to the selected match.

## Exceptions

ObjectDisposedException

The ITextFinder has already been disposed.

# Enum StopFindAction

Namespace: DotNetBrowser.Search

Assembly: DotNetBrowser.dll

The stop find actions.

```
public enum StopFindAction
```

## Fields

`ClearSelection = 1`

Stops the current search and clears the selection.

`KeepSelection = 2`

Stops the current search but leave the selection.

# Namespace DotNetBrowser.Search.Handlers

## Classes

FindResultReceivedParameters

The parameters of the handler that is passed to the Find(string, FindOptions, IHandler<FindResultReceivedParameters>) method and called when a find result is received.

# Class FindResultReceivedParameters

Namespace: DotNetBrowser.Search.Handlers

Assembly: DotNetBrowser.dll

The parameters of the handler that is passed to the Find(string, FindOptions, IHandler<FindResultReceivedParameters>) method and called when a find result is received.

```
public sealed class FindResultReceivedParameters
```

**Inheritance**

object ← FindResultReceivedParameters

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object)

## Properties

FindResult

Gets the find result.

IsSearchFinished

Indicates whether the searching process was finished.

TextFinder

Gets the ITextFinder instance associated with this object.

## Methods

ToString()

Represent object as string

# Property FindResult

Namespace: DotNetBrowser.Search.Handlers

Assembly: DotNetBrowser.dll

## FindResult

Gets the find result.

```
public FindResult FindResult { get; }
```

## Property Value

FindResult

The find result.

# Property IsSearchFinished

Namespace: DotNetBrowser.Search.Handlers

Assembly: DotNetBrowser.dll

## IsSearchFinished

Indicates whether the searching process was finished.

```
public bool IsSearchFinished { get; }
```

## Property Value

bool

`true` if the search was finished, `false` otherwise.

# Property TextFinder

Namespace: DotNetBrowser.Search.Handlers

Assembly: DotNetBrowser.dll

## TextFinder

Gets the ITextFinder instance associated with this object.

```
public ITextFinder TextFinder { get; }
```

## Property Value

ITextFinder

The ITextFinder instance associated with this object.

# Method ToString

Namespace: DotNetBrowser.Search.Handlers

Assembly: DotNetBrowser.dll

## ToString()

Represent object as string

```
public override string ToString()
```

## Returns

string

a string that represents the object

# Namespace DotNetBrowser.SpellCheck

## Classes

Language
: The language for which Chromium can perform spell checking.

LanguageNotAvailableException
: Thrown when the engine fails to configure the spellchecker to use the language passed to the Add(Language) method.

SpellCheckMenu
: A spell check menu in the context menu.

SpellCheckingResult
: The spell checking result that contains the bounds of a mis-spelled substring of the checked text.

## Interfaces

ILanguages
: The collection of the languages used for spell checking.

ISpellCheckDictionary
: Provides functionality for working with a spell check dictionary.

ISpellChecker
: Represents an engine service that provides functionality for configuring spell checking.

# Interface ILanguages

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

The collection of the languages used for spell checking.

```
public interface ILanguages
```

## Properties

All

Gets the list of all the languages currently used for spell checking.

## Methods

Add(Language)

Adds the language to the list of the languages for which spell checking is performed.

Remove(Language)

Removes the language from the list of the languages for which spell checking is performed.

# Property All

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## All

Gets the list of all the languages currently used for spell checking.

```
IReadOnlyList<Language> All { get; }
```

## Property Value

IReadOnlyList<Language>

The list of the languages used for spell checking.

## Remarks

On Linux and Windows the dictionaries for spell checking are downloaded programmatically and stored in the user data directory.

On macOS returns the languages for spell checking configured in the system settings.

## Exceptions

ObjectDisposedException

The ISpellChecker has already been disposed.

# Method Add

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Add(Language)

Adds the language to the list of the languages for which spell checking is performed.

```
Task Add(Language language)
```

## Parameters

`language` Language

The language to use for spell checking.

## Returns

Task

The task that can be used to wait until the dictionary is loaded from the user data directory. The task will fail with LanguageNotAvailableException if Chromium fails to download the dictionary for the given language (Windows and Linux), or the dictionary is not available in the system (macOS).

## Remarks

On Windows and Linux, this method loads the dictionary for the given language and blocks the current thread execution until the dictionary is loaded from the user data directory. If the dictionary does not exist in the user data directory, the engine will download it from the remote server.

On macOS, this method checks whether the specified dictionaries are present in the system, and throws a LanguageNotAvailableException if not.

## Exceptions

ArgumentNullException

The language is null.

ObjectDisposedException

The ISpellChecker has already been disposed.

# Method Remove

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Remove(Language)

Removes the language from the list of the languages for which spell checking is performed.

```
void Remove(Language language)
```

## Parameters

`language` Language

## Remarks

If all languages are removed, Chromium performs no spell checking. To enable spell checking, add a language using the Add(Language) method.

On macOS, this method does nothing because spellcheck languages are configured in the OS settings.

## Exceptions

ObjectDisposedException

The ISpellChecker has already been disposed.

# Interface ISpellCheckDictionary

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Provides functionality for working with a spell check dictionary.

```
public interface ISpellCheckDictionary : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Words

Gets the words in the dictionary.

## Methods

Add(string)

Adds the specified `word` to the dictionary and schedules a write to disk.

Contains(string)

Checks if the dictionary contains the specified `word`.

Remove(string)

Removes the specified `word` from the dictionary.

# Property Words

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Words

Gets the words in the dictionary.

```
IEnumerable<string> Words { get; }
```

## Property Value

IEnumerable<string>

A collection of the words in the dictionary. This collection will be empty if the dictionary does not have any words.

## Exceptions

ObjectDisposedException

The ISpellCheckDictionary has already been disposed.

# Method Add

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Add(string)

Adds the specified `word` to the dictionary and schedules a write to disk.

```
bool Add(string word)
```

### Parameters

`word` string

The word to add to the dictionary. It must be UTF8, between 1 and 99 bytes long, and without leading or trailing ASCII whitespace.

### Returns

bool

`true` if the word is valid and not a duplicate.

### Exceptions

ArgumentException

The `word` is null, empty, or contains only white space.

ObjectDisposedException

The ISpellCheckDictionary has already been disposed.

# Method Contains

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Contains(string)

Checks if the dictionary contains the specified `word`.

```
bool Contains(string word)
```

## Parameters

`word` string

The word to check. It must be UTF8, between 1 and 99 bytes long, and without leading or trailing ASCII whitespace.

## Returns

bool

`true` if the word is present in the dictionary, `false` otherwise.

## Exceptions

ArgumentException

The `word` is null, empty, or contains only white space.

ObjectDisposedException

The ISpellCheckDictionary has already been disposed.

# Method Remove

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Remove(string)

Removes the specified `word` from the dictionary.

```
bool Remove(string word)
```

## Parameters

`word` string

The word to remove from the dictionary. It must be UTF8, between 1 and 99 bytes long, and without leading or trailing ASCII whitespace.

## Returns

bool

`true` if the word was found and removed successfully, `false` otherwise.

## Exceptions

ArgumentException

The `word` is null, empty, or contains only white space.

ObjectDisposedException

The ISpellCheckDictionary has already been disposed.

# Interface ISpellChecker

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Represents an engine service that provides functionality for configuring spell checking.

```
public interface ISpellChecker : IAutoDisposable
```

**Inherited Members**

IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

CustomDictionary

Gets the custom dictionary.

Enabled

Enables or disables spell checking. By default, spell checking is enabled.

Engine

The IEngine instance associated with this object.

Languages

Gets the list of the languages used for spell checking.

# Property CustomDictionary

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## CustomDictionary

Gets the custom dictionary.

```
ISpellCheckDictionary CustomDictionary { get; }
```

## Property Value

ISpellCheckDictionary

The custom dictionary.

# Property Enabled

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Enabled

Enables or disables spell checking. By default, spell checking is enabled.

```
bool Enabled { get; set; }
```

## Property Value

bool

`true` if spell checking is enabled, `false` otherwise.

## Exceptions

ObjectDisposedException

The ISpellChecker has already been disposed.

# Property Engine

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Engine

The IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

# Property Languages

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Languages

Gets the list of the languages used for spell checking.

```
ILanguages Languages { get; }
```

## Property Value

ILanguages

The list of the languages used for spell checking.

## Remarks

On Linux and Windows the dictionaries for spell checking are downloaded programmatically and stored in the user data directory.

On macOS returns the languages for spell checking configured in the system settings.

## Exceptions

ObjectDisposedException

The ISpellChecker has already been disposed.

# Class Language

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

The language for which Chromium can perform spell checking.

```
public sealed class Language : TypedEnum<string>
```

**Inheritance**

object ← TypedEnum<string> ← Language

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Bulgarian

Bulgarian language ("bg")

Catalan

Catalan language ("ca")

Croatian

Croatian language ("hr")

Czech

Czech language ("cs")

Danish

Danish language ("da")

Default

Default language (English).

Dutch

Dutch language ("nl")

EnglishUk

English (UK) language ("en-gb")

EnglishUs
English (US) language ("en-us")

Estonian
Estonian language ("et")

French
French (France) language ("fr")

German
German language ("de")

Greek
Greek language ("el")

Hebrew
Hebrew language ("he")

Hindi
Hindi language ("hi")

Hungarian
Hungarian language ("hu")

Indonesian
Indonesian language ("id")

Italian
Italian language ("it")

Korean
Korean language ("ko")

Latvian
Latvian language ("lv")

Lithuanian
Lithuanian language ("lt")

Norwegian
Norwegian language ("nb")

Persian

Persian language ("fa")

Polish
Polish language ("pl")

PortugueseBrazil
Portuguese (Brazil) language ("pt-br")

PortuguesePortugal
Portuguese (Portugal) language ("pt-pt")

Romanian
Romanian language ("ro")

Russian
Russian language ("ru")

Serbian
Serbian language ("sr")

Slovak
Slovak language ("sk")

Slovenian
Slovenian language ("sl")

SpanishSpain
Spanish language (Spain) ("es")

Swedish
Swedish language ("sv")

Turkish
Turkish language ("tr")

Ukrainian
Ukrainian language ("uk")

Vietnamese
Vietnamese language ("vi")

## Properties

Code
: Gets the language code.

Region
: Gets the region code. Empty if the region is not specified.

## Methods

Create(string)
: Initializes the Language instance with the given Code.

Create(string, string)
: Initializes the Language instance with the given Code and Region.

FromCulture(CultureInfo)
: Initializes the Language instance from the specified culture information.

FromString(string)
: Returns the Language instance by its string representation.

## Operators

explicit operator Language(CultureInfo)
: Returns the Language instance by the specified culture information.

explicit operator Language(string)
: Returns the Language instance by its string representation.

# Field Bulgarian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Bulgarian language ("bg")

```
public static readonly Language Bulgarian
```

## Returns

Language

Bulgarian language ("bg")

# Field Catalan

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Catalan language ("ca")

```
public static readonly Language Catalan
```

## Returns

Language

Catalan language ("ca")

# Field Croatian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Croatian language ("hr")

```
public static readonly Language Croatian
```

## Returns

Language

Croatian language ("hr")

# Field Czech

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Czech language ("cs")

```
public static readonly Language Czech
```

## Returns

Language

Czech language ("cs")

# Field Danish

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Danish language ("da")

```
public static readonly Language Danish
```

## Returns

Language

Danish language ("da")

# Field Default

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Default language (English).

```
public static readonly Language Default
```

## Returns

Language

Default language (English).

# Field Dutch

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Dutch language ("nl")

```
public static readonly Language Dutch
```

## Returns

Language

Dutch language ("nl")

# Field EnglishUk

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

English (UK) language ("en-gb")

```
public static readonly Language EnglishUk
```

## Returns

Language

English (UK) language ("en-gb")

# Field EnglishUs

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

English (US) language ("en-us")

```
public static readonly Language EnglishUs
```

## Returns

Language

English (US) language ("en-us")

# Field Estonian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Estonian language ("et")

```
public static readonly Language Estonian
```

## Returns

Language

Estonian language ("et")

# Field French

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

French (France) language ("fr")

```
public static readonly Language French
```

## Returns

Language

French (France) language ("fr")

# Field German

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

German language ("de")

```
public static readonly Language German
```

## Returns

Language

German language ("de")

# Field Greek

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Greek language ("el")

```
public static readonly Language Greek
```

## Returns

Language

Greek language ("el")

# Field Hebrew

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Hebrew language ("he")

```
public static readonly Language Hebrew
```

## Returns

Language

Hebrew language ("he")

# Field Hindi

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Hindi language ("hi")

```
public static readonly Language Hindi
```

## Returns

Language

Hindi language ("hi")

# Field Hungarian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Hungarian language ("hu")

```
public static readonly Language Hungarian
```

## Returns

Language

Hungarian language ("hu")

# Field Indonesian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Indonesian language ("id")

```
public static readonly Language Indonesian
```

## Returns

Language

Indonesian language ("id")

# Field Italian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Italian language ("it")

```
public static readonly Language Italian
```

## Returns

Language

Italian language ("it")

# Field Korean

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Korean language ("ko")

```
public static readonly Language Korean
```

## Returns

Language

Korean language ("ko")

# Field Latvian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Latvian language ("lv")

```
public static readonly Language Latvian
```

## Returns

Language

Latvian language ("lv")

# Field Lithuanian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Lithuanian language ("lt")

```
public static readonly Language Lithuanian
```

## Returns

Language

Lithuanian language ("lt")

# Field Norwegian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Norwegian language ("nb")

```
public static readonly Language Norwegian
```

## Returns

Language

Norwegian language ("nb")

# Field Persian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Persian language ("fa")

```
public static readonly Language Persian
```

## Returns

Language

Persian language ("fa")

# Field Polish

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Polish language ("pl")

```
public static readonly Language Polish
```

## Returns

Language

Polish language ("pl")

# Field PortugueseBrazil

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Portuguese (Brazil) language ("pt-br")

```
public static readonly Language PortugueseBrazil
```

## Returns

Language

Portuguese (Brazil) language ("pt-br")

# Field PortuguesePortugal

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Portuguese (Portugal) language ("pt-pt")

```
public static readonly Language PortuguesePortugal
```

## Returns

Language

Portuguese (Portugal) language ("pt-pt")

# Field Romanian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Romanian language ("ro")

```
public static readonly Language Romanian
```

## Returns

Language

Romanian language ("ro")

# Field Russian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Russian language ("ru")

```
public static readonly Language Russian
```

## Returns

Language

Russian language ("ru")

# Field Serbian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Serbian language ("sr")

```
public static readonly Language Serbian
```

## Returns

Language

Serbian language ("sr")

# Field Slovak

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Slovak language ("sk")

```
public static readonly Language Slovak
```

## Returns

Language

Slovak language ("sk")

# Field Slovenian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Slovenian language ("sl")

```
public static readonly Language Slovenian
```

## Returns

Language

Slovenian language ("sl")

# Field SpanishSpain

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Spanish language (Spain) ("es")

```
public static readonly Language SpanishSpain
```

## Returns

Language

Spanish language (Spain) ("es")

# Field Swedish

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Swedish language ("sv")

```
public static readonly Language Swedish
```

## Returns

Language

Swedish language ("sv")

# Field Turkish

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Turkish language ("tr")

```
public static readonly Language Turkish
```

## Returns

Language

Turkish language ("tr")

# Field Ukrainian

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Ukrainian language ("uk")

```
public static readonly Language Ukrainian
```

## Returns

Language

Ukrainian language ("uk")

# Field Vietnamese

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Vietnamese language ("vi")

```
public static readonly Language Vietnamese
```

## Returns

Language

Vietnamese language ("vi")

# Property Code

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Code

Gets the language code.

```
public string Code { get; }
```

## Property Value

string

The language code.

# Property Region

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Region

Gets the region code. Empty if the region is not specified.

```
public string Region { get; }
```

## Property Value

string

The region code.

# Method Create

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Create(string, string)

Initializes the Language instance with the given Code and Region.

```
public static Language Create(string code, string region)
```

### Parameters

`code` string

The language code.

`region` string

The language code.

### Returns

Language

The Language instance.

## Create(string)

Initializes the Language instance with the given Code.

```
public static Language Create(string code)
```

### Parameters

`code` string

The language code.

## Returns

Language

The Language instance.

# Method FromCulture

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## FromCulture(CultureInfo)

Initializes the Language instance from the specified culture information.

```
public static Language FromCulture(CultureInfo cultureInfo)
```

## Parameters

`cultureInfo` CultureInfo

The culture information.

## Returns

Language

The Language instance.

# Method FromString

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## FromString(string)

Returns the Language instance by its string representation.

```
public static Language FromString(string value)
```

## Parameters

`value` string

The language string representation

## Returns

Language

The Language instance that corresponds to the provided string representation.

# Operator explicit operator

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## explicit operator Language(CultureInfo)

Returns the Language instance by the specified culture information.

```
public static explicit operator Language(CultureInfo cultureInfo)
```

### Parameters

`cultureInfo` CultureInfo

The culture information.

### Returns

Language

The Language instance that corresponds to the specified culture information.

## explicit operator Language(string)

Returns the Language instance by its string representation.

```
public static explicit operator Language(string value)
```

### Parameters

`value` string

The language string representation

## Returns

Language

The Language instance that corresponds to the provided string representation.

# Class LanguageNotAvailableException

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

Thrown when the engine fails to configure the spellchecker to use the language passed to the Add(Language) method.

```
public sealed class LanguageNotAvailableException : Exception, ISerializable
```

**Inheritance**

object ← Exception ← LanguageNotAvailableException

**Implements**

ISerializable

**Inherited Members**

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.ReferenceEquals(object, object)

## Properties

Language

Gets the language for which the dictionary is not available.

# Property Language

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Language

Gets the language for which the dictionary is not available.

```
public Language Language { get; }
```

## Property Value

Language

The language for which the dictionary is not available.

# Class SpellCheckMenu

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

A spell check menu in the context menu.

```
public sealed class SpellCheckMenu
```

**Inheritance**

object ← SpellCheckMenu

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

AddToDictionaryMenuItemText

Gets the localized text of the "Add to Dictionary" menu item.

DictionarySuggestions

Gets a collection of the suggested replacements for a misspelled word under the cursor.

MisspelledWord

Gets the misspelled word under the cursor, if any.

# Property AddToDictionaryMenuItemText

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## AddToDictionaryMenuItemText

Gets the localized text of the "Add to Dictionary" menu item.

```
public string AddToDictionaryMenuItemText { get; }
```

## Property Value

string

The localized text of the "Add to Dictionary" menu item.

# Property DictionarySuggestions

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## DictionarySuggestions

Gets a collection of the suggested replacements for a misspelled word under the cursor.

```
public IEnumerable<string> DictionarySuggestions { get; }
```

## Property Value

IEnumerable<string>

The suggested replacements for a misspelled word under the cursor. Can be empty.

# Property MisspelledWord

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## MisspelledWord

Gets the misspelled word under the cursor, if any.

```
public string MisspelledWord { get; }
```

## Property Value

string

The misspelled word under the cursor, if any.

# Class SpellCheckingResult

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

The spell checking result that contains the bounds of a mis-spelled substring of the checked text.

```
public sealed class SpellCheckingResult
```

**Inheritance**

object ← SpellCheckingResult

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Length

Gets the length of the mis-spelled word in the checked text.

Location

Gets the location of the first symbol in the mis-spelled word in the checked text that is considered as mis-spelled by the spell checker.

# Property Length

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Length

Gets the length of the mis-spelled word in the checked text.

```
public uint Length { get; }
```

## Property Value

uint

The length of the mis-spelled word in the checked text.

# Property Location

Namespace: DotNetBrowser.SpellCheck

Assembly: DotNetBrowser.dll

## Location

Gets the location of the first symbol in the mis-spelled word in the checked text that is considered as mis-spelled by the spell checker.

```
public uint Location { get; }
```

## Property Value

uint

The location of the first symbol in the mis-spelled word in the checked text that is considered as mis-spelled by the spell checker.

# Namespace DotNetBrowser.Ui

## Classes

Bitmap

Represents a binary image that consists of an image size and a byte array that contains the pre-multiplied image pixels in the BGRA format.

Color

A numeric model of an RGB color. The components of the color instance are presented in the arithmetic notation. This means that each component accepts any fractional value from 0 to 1.

FontSize

The browser font size.

Language

The supported user interface languages.

# Class Bitmap

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Represents a binary image that consists of an image size and a byte array that contains the pre-multiplied image pixels in the BGRA format.

```
public sealed class Bitmap
```

**Inheritance**

object ← Bitmap

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

**Extension Methods**

BitmapExtensions.ToSkBitmap(Bitmap) , BitmapExtensions.ToSkImage(Bitmap) , BitmapExtensions.ToUiBitmap(Bitmap)

## Properties

Pixels

Gets a read-only byte array that contains the pre-multiplied pixels of the binary image. Each pixel allocates 4 bytes in the BGRA format.

Size

Gets the size of this bitmap instance.

# Property Pixels

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Pixels

Gets a read-only byte array that contains the pre-multiplied pixels of the binary image. Each pixel allocates 4 bytes in the BGRA format.

```
public IReadOnlyList<byte> Pixels { get; }
```

## Property Value

IReadOnlyList<byte>

A read-only byte array that contains the pre-multiplied pixels of the binary image.

# Property Size

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Size

Gets the size of this bitmap instance.

```
public Size Size { get; }
```

## Property Value

Size

The size of this bitmap instance. The size can be empty.

# Class Color

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

A numeric model of an RGB color. The components of the color instance are presented in the arithmetic notation. This means that each component accepts any fractional value from 0 to 1.

```
public sealed class Color : IFormattable
```

**Inheritance**

object ← Color

**Implements**

IFormattable

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

## Remarks

If all the components except `alpha` are at zero and the `alpha` is at 1, the result is black. If all are at 1, the result is the brightest representable white.

**Important:** the component values out of the `0..1` range are not allowed and should not be used.

## Constructors

Color(float, float, float, float)

Initializes a new Color instance with the specified channel values.

## Properties

Alpha

Gets the opacity channel value in the `0..1` range. When the value is 1, the color is 100% opaque. When 0, the color is 100% transparent.

Blue
: Gets the blue channel value in the `0..1` range.

Green
: Gets the green channel value in the `0..1` range.

Red
: Gets the red channel value in the `0..1` range.

## Methods

Equals(object)

GetHashCode()

ToHexRGB()
: Creates a string representation for this color in the `RGB` hex format.

ToHexRGBA()
: Creates a string representation for this color in the `RGBA` hex format.

ToString()

ToString(string)
: Returns a string representation of the value of this Color instance, according to the provided format specifier.

ToString(string, IFormatProvider)

# Constructor Color

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Color(float, float, float, float)

Initializes a new Color instance with the specified channel values.

```
public Color(float red, float green, float blue, float alpha = 1)
```

## Parameters

`red` float

The red channel value.

`green` float

The green channel value.

`blue` float

The blue channel value.

`alpha` float

The alpha channel value.

## Exceptions

ArgumentOutOfRangeException

thrown when any of the parameters is outside the `0..1` range.

# Property Alpha

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Alpha

Gets the opacity channel value in the `0..1` range. When the value is 1, the color is 100% opaque. When 0, the color is 100% transparent.

```
public float Alpha { get; }
```

## Property Value

float

The opacity channel value in the `0..1` range.

# Property Blue

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Blue

Gets the blue channel value in the `0..1` range.

```
public float Blue { get; }
```

## Property Value

float

The blue channel value in the `0..1` range.

# Property Green

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Green

Gets the green channel value in the `0..1` range.

```
public float Green { get; }
```

## Property Value

float

The green channel value in the `0..1` range.

# Property Red

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Red

Gets the red channel value in the `0..1` range.

```
public float Red { get; }
```

## Property Value

float

The red channel value in the `0..1` range.

# Method Equals

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Method ToHexRGB

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## ToHexRGB()

Creates a string representation for this color in the `RGB` hex format.

```
public string ToHexRGB()
```

## Returns

string

The string representation for this color in the `RGB` hex format.

# Method ToHexRGBA

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## ToHexRGBA()

Creates a string representation for this color in the `RGBA` hex format.

```
public string ToHexRGBA()
```

## Returns

string

The string representation for this color in the `RGBA` hex format.

# Method ToString

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## ToString(string, IFormatProvider)

```
public string ToString(string format, IFormatProvider formatProvider)
```

### Parameters

`format` string

`formatProvider` IFormatProvider

### Returns

string

### Remarks

The supported format strings are:

- **G** - General string representation: `Red: <value>, Green: <value>, Blue: <value>, Alpha: <value>`
- **RGB** - Hexademical representation (with alpha omitted): `RRGGBB`.
- **RGBA** - Hexademical representation: `RRGGBBAA`.

## ToString()

```
public override string ToString()
```

### Returns

string

## ToString(string)

Returns a string representation of the value of this Color instance, according to the provided format specifier.

```
public string ToString(string format)
```

## Parameters

`format` string

A single format specifier that indicates how to format the value of this Color. The format parameter can be "G", "F", "RGB", or "RGBA". If format is null or an empty string (""), "G" is used.

## Returns

string

The value of this Color, represented in the specified format.

## Remarks

The supported format strings are:

- **G -** General string representation: `Red: <value>, Green: <value>, Blue: <value>, Alpha: <value>`
- **RGB -** Hexademical representation (with alpha omitted): `RRGGBB`.
- **RGBA -** Hexademical representation: `RRGGBBAA`.

# Class FontSize

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

The browser font size.

```
public sealed class FontSize
```

**Inheritance**

object ← FontSize

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Constructors

FontSize(int)

Creates a new font size.

## Properties

Value

Gets the font size in pixels.

## Methods

Equals(object)

GetHashCode()

## Operators

implicit operator int(FontSize)

Converts the font size to integer.

implicit operator FontSize(int)

Converts an integer to the font size.

# Constructor FontSize

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## FontSize(int)

Creates a new font size.

```
public FontSize(int value)
```

## Parameters

`value` int

The font size in pixels. Cannot be zero or negative.

## Exceptions

ArgumentException

The font size in pixels is zero or negative.

# Property Value

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Value

Gets the font size in pixels.

```
public int Value { get; }
```

## Property Value

int

The font size in pixels.

# Method Equals

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Operator implicit operator

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## implicit operator int(FontSize)

Converts the font size to integer.

```
public static implicit operator int(FontSize size)
```

### Parameters

`size` FontSize

The font size to convert.

### Returns

int

## implicit operator FontSize(int)

Converts an integer to the font size.

```
public static implicit operator FontSize(int value)
```

### Parameters

`value` int

The font size in pixels. Cannot be zero or negative.

## Returns

FontSize

## Exceptions

ArgumentException

The font size in pixels is zero or negative.

# Class Language

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

The supported user interface languages.

```
public sealed class Language : TypedEnum<string>
```

**Inheritance**

object ← TypedEnum<string> ← Language

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Amharic
Amharic language ("am")

Arabic
Arabic language ("ar")

Bengali
Bengali language ("bn")

Bulgarian
Bulgarian language ("bg")

Catalan
Catalan language ("ca")

Chinese
Chinese language ("zh-cn")

ChineseTaiwan
Chinese (Taiwan) language ("zh-tw")

Czech

Czech language ("cs")

Danish
Danish language ("da")

Default
Default language (English).

Dutch
Dutch language ("nl")

EnglishUk
English (UK) language ("en-gb")

EnglishUs
English (US) language ("en-us")

Estonian
Estonian language ("et")

Filipino
Filipino language ("fil")

Finnish
Finnish language ("fi")

French
French (France) language ("fr")

German
German language ("de")

Greek
Greek language ("el")

Gujarati
Gujarati language ("gu")

Hebrew
Hebrew language ("he")

Hindi
Hindi language ("hi")

Hungarian

Hungarian language ("hu")

Indonesian
Indonesian language ("id")

Italian
Italian language ("it")

Japanese
Japanese language ("ja")

Kannada
Kannada language ("kn")

Korean
Korean language ("ko")

Latvian
Latvian language ("lv")

Lithuanian
Lithuanian language ("lt")

Malay
Malay language ("ms")

Malayalam
Malayalam language ("ml")

Marathi
Marathi language ("mr")

Norwegian
Norwegian language ("nb")

Persian
Persian language ("fa")

Polish
Polish language ("pl")

PortugueseBrazil
Portuguese (Brazil) language ("pt-br")

PortuguesePortugal

Portuguese (Portugal) language ("pt-pt")

Romanian
Romanian language ("ro")

Russian
Russian language ("ru")

Serbian
Serbian language ("sr")

Slovak
Slovak language ("sk")

Slovenian
Slovenian language ("sl")

SpanishLatinAmerica
Spanish (Latin America) language ("es-419")

SpanishSpain
Spanish language (Spain) ("es")

Swahili
Swahili language ("sw")

Swedish
Swedish language ("sv")

Tamil
Tamil language ("ta")

Telugu
Telugu language ("te")

Thai
Thai language ("th")

Turkish
Turkish language ("tr")

Ukrainian
Ukrainian language ("uk")

Vietnamese

Vietnamese language ("vi")

## Methods

FromCulture(CultureInfo)
Initializes the Language instance from the specified culture information.

FromString(string)
Returns the Language instance by its string representation.

## Operators

explicit operator Language(CultureInfo)
Returns the Language instance by the specified culture information.

explicit operator Language(string)
Returns the Language instance by its string representation.

# Field Amharic

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Amharic language ("am")

```
public static readonly Language Amharic
```

## Returns

Language

Amharic language ("am")

# Field Arabic

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Arabic language ("ar")

```
public static readonly Language Arabic
```

## Returns

Language

Arabic language ("ar")

# Field Bengali

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Bengali language ("bn")

```
public static readonly Language Bengali
```

## Returns

Language

Bengali language ("bn")

# Field Bulgarian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Bulgarian language ("bg")

```
public static readonly Language Bulgarian
```

## Returns

Language

Bulgarian language ("bg")

# Field Catalan

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Catalan language ("ca")

```
public static readonly Language Catalan
```

## Returns

Language

Catalan language ("ca")

# Field Chinese

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Chinese language ("zh-cn")

```
public static readonly Language Chinese
```

## Returns

Language

Chinese language ("zh-cn")

# Field ChineseTaiwan

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Chinese (Taiwan) language ("zh-tw")

```
public static readonly Language ChineseTaiwan
```

## Returns

Language

Chinese (Taiwan) language ("zh-tw")

# Field Czech

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Czech language ("cs")

```
public static readonly Language Czech
```

## Returns

Language

Czech language ("cs")

# Field Danish

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Danish language ("da")

```
public static readonly Language Danish
```

## Returns

Language

Danish language ("da")

# Field Default

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Default language (English).

```
public static readonly Language Default
```

## Returns

Language

Default language (English).

# Field Dutch

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Dutch language ("nl")

```
public static readonly Language Dutch
```

## Returns

Language

Dutch language ("nl")

# Field EnglishUk

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

English (UK) language ("en-gb")

```
public static readonly Language EnglishUk
```

## Returns

Language

English (UK) language ("en-gb")

# Field EnglishUs

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

English (US) language ("en-us")

```
public static readonly Language EnglishUs
```

## Returns

Language

English (US) language ("en-us")

# Field Estonian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Estonian language ("et")

```
public static readonly Language Estonian
```

## Returns

Language

Estonian language ("et")

# Field Filipino

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Filipino language ("fil")

```
public static readonly Language Filipino
```

## Returns

Language

Filipino language ("fil")

# Field Finnish

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Finnish language ("fi")

```
public static readonly Language Finnish
```

## Returns

Language

Finnish language ("fi")

# Field French

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

French (France) language ("fr")

```
public static readonly Language French
```

## Returns

Language

French (France) language ("fr")

# Field German

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

German language ("de")

```
public static readonly Language German
```

## Returns

Language

German language ("de")

# Field Greek

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Greek language ("el")

```
public static readonly Language Greek
```

## Returns

Language

Greek language ("el")

# Field Gujarati

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Gujarati language ("gu")

```
public static readonly Language Gujarati
```

## Returns

Language

Gujarati language ("gu")

# Field Hebrew

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Hebrew language ("he")

```
public static readonly Language Hebrew
```

## Returns

Language

Hebrew language ("he")

# Field Hindi

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Hindi language ("hi")

```
public static readonly Language Hindi
```

## Returns

Language

Hindi language ("hi")

# Field Hungarian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Hungarian language ("hu")

```
public static readonly Language Hungarian
```

## Returns

Language

Hungarian language ("hu")

# Field Indonesian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Indonesian language ("id")

```
public static readonly Language Indonesian
```

## Returns

Language

Indonesian language ("id")

# Field Italian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Italian language ("it")

```
public static readonly Language Italian
```

## Returns

Language

Italian language ("it")

# Field Japanese

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Japanese language ("ja")

```
public static readonly Language Japanese
```

## Returns

Language

Japanese language ("ja")

# Field Kannada

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Kannada language ("kn")

```
public static readonly Language Kannada
```

## Returns

Language

Kannada language ("kn")

# Field Korean

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Korean language ("ko")

```
public static readonly Language Korean
```

## Returns

Language

Korean language ("ko")

# Field Latvian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Latvian language ("lv")

```
public static readonly Language Latvian
```

## Returns

Language

Latvian language ("lv")

# Field Lithuanian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Lithuanian language ("lt")

```
public static readonly Language Lithuanian
```

## Returns

Language

Lithuanian language ("lt")

# Field Malay

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Malay language ("ms")

```
public static readonly Language Malay
```

## Returns

Language

Malay language ("ms")

# Field Malayalam

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Malayalam language ("ml")

```
public static readonly Language Malayalam
```

## Returns

Language

Malayalam language ("ml")

# Field Marathi

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Marathi language ("mr")

```
public static readonly Language Marathi
```

## Returns

Language

Marathi language ("mr")

# Field Norwegian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Norwegian language ("nb")

```
public static readonly Language Norwegian
```

## Returns

Language

Norwegian language ("nb")

# Field Persian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Persian language ("fa")

```
public static readonly Language Persian
```

## Returns

Language

Persian language ("fa")

# Field Polish

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Polish language ("pl")

```
public static readonly Language Polish
```

## Returns

Language

Polish language ("pl")

# Field PortugueseBrazil

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Portuguese (Brazil) language ("pt-br")

```
public static readonly Language PortugueseBrazil
```

## Returns

Language

Portuguese (Brazil) language ("pt-br")

# Field PortuguesePortugal

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Portuguese (Portugal) language ("pt-pt")

```
public static readonly Language PortuguesePortugal
```

## Returns

Language

Portuguese (Portugal) language ("pt-pt")

# Field Romanian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Romanian language ("ro")

```
public static readonly Language Romanian
```

## Returns

Language

Romanian language ("ro")

# Field Russian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Russian language ("ru")

```
public static readonly Language Russian
```

## Returns

Language

Russian language ("ru")

# Field Serbian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Serbian language ("sr")

```
public static readonly Language Serbian
```

## Returns

Language

Serbian language ("sr")

# Field Slovak

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Slovak language ("sk")

```
public static readonly Language Slovak
```

## Returns

Language

Slovak language ("sk")

# Field Slovenian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Slovenian language ("sl")

```
public static readonly Language Slovenian
```

## Returns

Language

Slovenian language ("sl")

# Field SpanishLatinAmerica

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Spanish (Latin America) language ("es-419")

```
public static readonly Language SpanishLatinAmerica
```

## Returns

Language

Spanish (Latin America) language ("es-419")

# Field SpanishSpain

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Spanish language (Spain) ("es")

```
public static readonly Language SpanishSpain
```

## Returns

Language

Spanish language (Spain) ("es")

# Field Swahili

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Swahili language ("sw")

```
public static readonly Language Swahili
```

## Returns

Language

Swahili language ("sw")

# Field Swedish

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Swedish language ("sv")

```
public static readonly Language Swedish
```

## Returns

Language

Swedish language ("sv")

# Field Tamil

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Tamil language ("ta")

```
public static readonly Language Tamil
```

## Returns

Language

Tamil language ("ta")

# Field Telugu

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Telugu language ("te")

```
public static readonly Language Telugu
```

## Returns

Language

Telugu language ("te")

# Field Thai

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Thai language ("th")

```
public static readonly Language Thai
```

## Returns

Language

Thai language ("th")

# Field Turkish

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Turkish language ("tr")

```
public static readonly Language Turkish
```

## Returns

Language

Turkish language ("tr")

# Field Ukrainian

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Ukrainian language ("uk")

```
public static readonly Language Ukrainian
```

## Returns

Language

Ukrainian language ("uk")

# Field Vietnamese

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

Vietnamese language ("vi")

```
public static readonly Language Vietnamese
```

## Returns

Language

Vietnamese language ("vi")

# Method FromCulture

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## FromCulture(CultureInfo)

Initializes the Language instance from the specified culture information.

```
public static Language FromCulture(CultureInfo cultureInfo)
```

### Parameters

`cultureInfo` CultureInfo

The culture information.

### Returns

Language

The Language instance.

# Method FromString

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## FromString(string)

Returns the Language instance by its string representation.

```
public static Language FromString(string value)
```

## Parameters

`value` string

The language string representation

## Returns

Language

The Language instance that corresponds to the provided string representation.

# Operator explicit operator

Namespace: DotNetBrowser.Ui

Assembly: DotNetBrowser.dll

## explicit operator Language(CultureInfo)

Returns the Language instance by the specified culture information.

```
public static explicit operator Language(CultureInfo cultureInfo)
```

### Parameters

`cultureInfo` CultureInfo

The culture information.

### Returns

Language

The Language instance that corresponds to the specified culture information.

## explicit operator Language(string)

Returns the Language instance by its string representation.

```
public static explicit operator Language(string value)
```

### Parameters

`value` string

The language string representation

## Returns

Language

The Language instance that corresponds to the provided string representation.

# Namespace DotNetBrowser.Ui.Cursors

## Enums

CursorType

The cursor types.

# Enum CursorType

Namespace: DotNetBrowser.Ui.Cursors

Assembly: DotNetBrowser.dll

The cursor types.

```
public enum CursorType
```

## Fields

`Alias = 34`

The alias (shortcut) cursor.

`Cell = 32`

The cell cursor.

`ColumnResize = 19`

The column resizing cursor.

`ContextMenu = 33`

RThe context menu cursor.

`Copy = 37`

The copy cursor.

`Cross = 2`

The crosshair cursor.

`Custom = 44`

The custom cursor defined on the particular web page.

`DndCopy = 47`

The drag-and-drop operation "Copy" cursor.

`DndLink = 48`

The drag-and-drop operation "Link" cursor.

`DndMove = 46`

The drag-and-drop operation "Move" cursor.

`DndNone = 45`

The drag-and-drop operation "None" cursor.

`EastPanning = 22`

The east panning cursor.

`EastResize = 7`

The east sizing cursor.

`EastWestNoResize = 49`

The no-resize cursor (resize cannot be performed in east-west direction).

`EastWestResize = 16`

The two-headed east/west sizing cursor.

`Grab = 42`

The grab (hand) cursor.

`Grabbing = 43`

The grabbing hand cursor.

`Hand = 3`

The hand cursor.

`Help = 6`

The help cursor, which is a combination of an arrow and a question mark.

`IBeam = 4`

The I-beam cursor, which is used to show where the text cursor appears when the mouse is clicked.

`MiddlePanning = 21`

The middle panning cursor.

`Move = 30`

The move cursor.

`NoDrop = 36`

The cursor with which indicates that a particular region is invalid for a given operation.

`None = 38`

The special cursor that is invisible.

`NorthEastPanning = 24`

The northeast panning cursor.

`NorthEastResize = 9`

The northeast sizing cursor.

`NorthEastSouthWestNoResize = 51`

The no-resize cursor (resize cannot be performed in northeast-southwest direction).

`NorthEastSouthWestResize = 17`

The two-headed northeast/southwest sizing cursor.

`NorthPanning = 23`

The panning cursor.

`NorthResize = 8`

The northsizing cursor.

`NorthSouthNoResize = 50`

The no-resize cursor (resize cannot be performed in north-south direction).

`NorthSouthResize = 15`

The two-headed north/south sizing cursor.

`NorthWestPanning = 25`

The northwest panning cursor.

`NorthWestResize = 10`

The northwest sizing cursor.

`NorthWestSouthEastNoResize = 52`

The no-resize cursor (resize cannot be performed in northwest-southeast direction).

`NorthWestSouthEastResize = 18`

The two-headed northwest/southeast sizing cursor.

`NotAllowed = 39`

The cursor with which indicates that a particular region is not allowed for a given operation.

`Pointer = 1`

The pointer (arrow) cursor.

`Progress = 35`

The cursor that appears when an operation is in progress, for instance, when an application is starting.

`RowResize = 20`

The row resizing cursor.

`SouthEastPanning = 27`

The southeast panning cursor.

`SouthEastResize = 12`

The southeast sizing cursor.

`SouthPanning = 26`

The south panning cursor.

`SouthResize = 11`

The sizing cursor.

`SouthWestPanning = 28`

The southwest panning cursor.

`SouthWestResize = 13`

The sizing cursor.

`Unspecified = 0`

The cursor type is not specified.

`VerticalText = 31`

The text cursor (similar to I-beam) fro vertical text.

`Wait = 5`

The wait (hourglass) cursor.

`WestPanning = 29`

The west panning cursor.

`WestResize = 14`

The sizing cursor.

`ZoomIn = 40`

The zoom in cursor.

`ZoomOut = 41`

The zoom out cursor.

# Namespace DotNetBrowser.UserData

## Classes

Address

The user's address containing information about a street, city, state, etc.

UserDataProfile

The collected data entered by the user to a form and persisted to the user data store.

## Interfaces

IUserDataProfileStore

The data store for the user data.

IUserDataProfiles

A service that allows managing user data profiles.

# Class Address

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

The user's address containing information about a street, city, state, etc.

```
public sealed class Address
```

**Inheritance**

object ← Address

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

City

Gets the city from the address.

CountryCode

Gets the ISO 3166 2-letter country code.

DependentLocality

Gets the subdivision of a city, e.g. inner-city district or suburb.

State

Gets the name of the state.

Street

Gets the whole street name.

Zip

Gets the ZIP code entered by the user.

## Methods

Equals(object)

GetHashCode()

# Property City

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## City

Gets the city from the address.

```
public string City { get; }
```

## Property Value

string

The city from the address.

# Property CountryCode

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## CountryCode

Gets the ISO 3166 2-letter country code.

```
public string CountryCode { get; }
```

## Property Value

string

The 2-letter country code.

# Property DependentLocality

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## DependentLocality

Gets the subdivision of a city, e.g. inner-city district or suburb.

```
public string DependentLocality { get; }
```

## Property Value

string

The subdivision of a city.

# Property State

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## State

Gets the name of the state.

```
public string State { get; }
```

## Property Value

string

The name of the state.

# Property Street

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Street

Gets the whole street name.

```
public string Street { get; }
```

## Property Value

string

The street name.

# Property Zip

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Zip

Gets the ZIP code entered by the user.

```
public string Zip { get; }
```

## Property Value

string

The ZIP code.

# Method Equals

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Interface IUserDataProfileStore

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

The data store for the user data.

```
public interface IUserDataProfileStore
```

## Properties

All
Gets all user data records.

Profile
Gets the IProfile instance associated with this IUserDataProfileStore.

## Methods

Clear()
Clears all records in the store.

Remove(UserDataProfile)
Removes the user data from the store.

# Property All

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## All

Gets all user data records.

```
IReadOnlyList<UserDataProfile> All { get; }
```

## Property Value

IReadOnlyList<UserDataProfile>

All user data records.

## Remarks

The user data records are saved to the store via SaveUserDataProfileHandler.

## Exceptions

ObjectDisposedException

The IUserDataProfileStore has already been disposed.

# Property Profile

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this IUserDataProfileStore.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this IUserDataProfileStore.

# Method Clear

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Clear()

Clears all records in the store.

```
void Clear()
```

## Exceptions

ObjectDisposedException

The IUserDataProfileStore has already been disposed.

# Method Remove

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Remove(UserDataProfile)

Removes the user data from the store.

```
void Remove(UserDataProfile userDataProfile)
```

## Parameters

`userDataProfile` UserDataProfile

The user data profile associated with the removed user data profile records.

## Remarks

Removed user data is not displayed in the autofill suggestion pop-up.

## Exceptions

ObjectDisposedException

The IUserDataProfileStore has already been disposed.

ArgumentException

The `userDataProfile` is null.

# Interface IUserDataProfiles

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

A service that allows managing user data profiles.

```
public interface IUserDataProfiles : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

Browser
: Gets the IBrowser instance associated with this browser.

SaveUserDataProfileHandler
: Gets or sets a handler that is used when the user is prompted to save the user data profile to the user data profile store .

UpdateUserDataProfileHandler
: Gets or sets a handler that is used when the user is prompted to update the user data profile to the user data profile store. For example, when the user changes the email address to an empty string.

# Property Browser

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with this browser.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with this IUserDataProfiles instance.

# Property SaveUserDataProfileHandler

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## SaveUserDataProfileHandler

Gets or sets a handler that is used when the user is prompted to save the user data profile to the user data profile store .

```
IHandler<SaveUserDataProfileParameters, SaveUserDataProfileResponse>
SaveUserDataProfileHandler { get; set; }
```

## Property Value

IHandler<SaveUserDataProfileParameters, SaveUserDataProfileResponse>

## Remarks

This handler is equivalent to the "Save Address?" bubble in Chromium.

The handler is invoked when the user submits a web form where fields are associated with a user data profile such as city, state, street, zip code, email address, etc.

Use the Save to save this user data profile to the autofill store. All saved user data profiles are shown in the suggestion pop-up when focusing the web form control.

Use the Decline to decline to save the user data profile. If the current profile is declined then the callback will be invoked again when submitting the web form with the same user data.

The handler is not invoked if autofill is disabled.

## Exceptions

ObjectDisposedException

The IUserDataProfiles has already been disposed.

# Property UpdateUserDataProfileHandler

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## UpdateUserDataProfileHandler

Gets or sets a handler that is used when the user is prompted to update the user data profile to the user data profile store. For example, when the user changes the email address to an empty string.

```
IHandler<UpdateUserDataProfileParameters, UpdateUserDataProfileResponse>
UpdateUserDataProfileHandler { get; set; }
```

## Property Value

IHandler<UpdateUserDataProfileParameters, UpdateUserDataProfileResponse>

## Remarks

This callback is equivalent to the "Update Address?" bubble in Chromium.

The handler is invoked when the user submits a web form where fields are associated with a user data profile such as city, state, street, zip code, email address, etc.

Use the Update to update the user data profile in the autofill data store.

Use the Decline to decline to update the user data profile. If the current user data profile is declined then the callback will be invoked again when submitting the web form with the same user data.

The handler is not invoked if autofill is disabled.

## Exceptions

ObjectDisposedException

The IUserDataProfiles has already been disposed.

# Class UserDataProfile

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

The collected data entered by the user to a form and persisted to the user data store.

```
public sealed class UserDataProfile
```

**Inheritance**

object ← UserDataProfile

**Inherited Members**

object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Address

Gets the address entered by the user.

CompanyName

Gets the name of the company entered by the user.

Email

Gets the email value entered by the user.

FullName

Gets the full name value entered by the user.

PhoneNumber

Gets the phone number value entered by the user.

## Methods

Equals(object)

GetHashCode()

# Property Address

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Address

Gets the address entered by the user.

```
public Address Address { get; }
```

## Property Value

Address

The address.

# Property CompanyName

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## CompanyName

Gets the name of the company entered by the user.

```
public string CompanyName { get; }
```

## Property Value

string

The name of the company.

# Property Email

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Email

Gets the email value entered by the user.

```
public string Email { get; }
```

## Property Value

string

The email.

# Property FullName

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## FullName

Gets the full name value entered by the user.

```
public string FullName { get; }
```

## Property Value

string

The full name.

# Property PhoneNumber

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## PhoneNumber

Gets the phone number value entered by the user.

```
public string PhoneNumber { get; }
```

## Property Value

string

The phone number.

# Method Equals

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## Equals(object)

```
public override bool Equals(object obj)
```

### Parameters

`obj` object

### Returns

bool

# Method GetHashCode

Namespace: DotNetBrowser.UserData

Assembly: DotNetBrowser.dll

## GetHashCode()

```
public override int GetHashCode()
```

## Returns

int

# Namespace DotNetBrowser.UserData.Handlers

## Classes

SaveUserDataProfileParameters

The parameters of the SaveUserDataProfileHandler.

SaveUserDataProfileResponse

A response to the SaveUserDataProfileHandler.

UpdateUserDataProfileParameters

The parameters of the UpdateUserDataProfileHandler.

UpdateUserDataProfileResponse

A response to the UpdateUserDataProfileHandler.

# Class SaveUserDataProfileParameters

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

The parameters of the SaveUserDataProfileHandler.

```
public sealed class SaveUserDataProfileParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← SaveUserDataProfileParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

UserDataProfile

Gets the user data profile entered in a web form.

# Property UserDataProfile

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

## UserDataProfile

Gets the user data profile entered in a web form.

```
public UserDataProfile UserDataProfile { get; }
```

## Property Value

UserDataProfile

The user data profile.

# Class SaveUserDataProfileResponse

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

A response to the SaveUserDataProfileHandler.

```
public sealed class SaveUserDataProfileResponse
```

**Inheritance**

object ← SaveUserDataProfileResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Decline

Creates a SaveUserDataProfileResponse that notifies the browser to decline to save the user data profile.

Save

Creates a SaveUserDataProfileResponse that notifies the browser to save the user data profile to the user data store.

# Field Decline

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

Creates a SaveUserDataProfileResponse that notifies the browser to decline to save the user data profile.

```
public static SaveUserDataProfileResponse Decline
```

## Returns

SaveUserDataProfileResponse

The SaveUserDataProfileResponse instance that can be used as a return value in SaveUserDataProfileHandler implementation.

# Field Save

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

Creates a SaveUserDataProfileResponse that notifies the browser to save the user data profile to the user data store.

```
public static SaveUserDataProfileResponse Save
```

## Returns

SaveUserDataProfileResponse

The SaveUserDataProfileResponse instance that can be used as a return value in SaveUserDataProfileHandler implementation.

# Class UpdateUserDataProfileParameters

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

The parameters of the UpdateUserDataProfileHandler.

```
public sealed class UpdateUserDataProfileParameters : BrowserParameters
```

**Inheritance**

object ← BrowserParameters ← UpdateUserDataProfileParameters

**Inherited Members**

BrowserParameters.Browser , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

OriginalUserDataProfile

Gets the original user data profile that is about to be updated.

UserDataProfile

Gets the new user data profile that is about to replace the original profile.

# Property OriginalUserDataProfile

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

## OriginalUserDataProfile

Gets the original user data profile that is about to be updated.

```
public UserDataProfile OriginalUserDataProfile { get; }
```

## Property Value

UserDataProfile

The original user data profile.

# Property UserDataProfile

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

## UserDataProfile

Gets the new user data profile that is about to replace the original profile.

```
public UserDataProfile UserDataProfile { get; }
```

## Property Value

UserDataProfile

The new user data profile.

# Class UpdateUserDataProfileResponse

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

A response to the UpdateUserDataProfileHandler.

```
public sealed class UpdateUserDataProfileResponse
```

**Inheritance**

object ← UpdateUserDataProfileResponse

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

Decline

Creates a UpdateUserDataProfileResponse that notifies the browser to decline to update the user data profile.

Update

Creates a UpdateUserDataProfileResponse that notifies the browser to update the user data profile in the user data store.

# Field Decline

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

Creates a UpdateUserDataProfileResponse that notifies the browser to decline to update the user data profile.

```
public static UpdateUserDataProfileResponse Decline
```

## Returns

UpdateUserDataProfileResponse

The UpdateUserDataProfileResponse instance that can be used as a return value in UpdateUserDataProfileHandler implementation.

# Field Update

Namespace: DotNetBrowser.UserData.Handlers

Assembly: DotNetBrowser.dll

Creates a UpdateUserDataProfileResponse that notifies the browser to update the user data profile in the user data store.

```
public static UpdateUserDataProfileResponse Update
```

## Returns

UpdateUserDataProfileResponse

The UpdateUserDataProfileResponse instance that can be used as a return value in UpdateUserDataProfileHandler implementation.

# Namespace DotNetBrowser.WinForms

## Classes

BitmapExtensions

Extensions for Bitmap class.

BrowserView

The WinForms-based implementation of browser view.

# Class BitmapExtensions

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

Extensions for Bitmap class.

```
public static class BitmapExtensions
```

**Inheritance**

object ← BitmapExtensions

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Methods

ToBitmap(Bitmap)

Converts the Bitmap instance to Bitmap.

# Method ToBitmap

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## ToBitmap(Bitmap)

Converts the Bitmap instance to Bitmap.

```
public static Bitmap ToBitmap(this Bitmap bitmap)
```

### Parameters

`bitmap` Bitmap

the bitmap to convert

### Returns

Bitmap

the converted bitmap.

# Class BrowserView

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

The WinForms-based implementation of browser view.

```
public class BrowserView : Panel, IDropTarget, ISynchronizeInvoke, IWin32Window,
IBindableComponent, IComponent, IDisposable, IBrowserView
```

### Inheritance

object ← MarshalByRefObject ← Component ← Control ← ScrollableControl ← Panel ← BrowserView

### Implements

IDropTarget, ISynchronizeInvoke, IWin32Window, IBindableComponent, IComponent, IDisposable, IBrowserView

### Inherited Members

Panel.OnResize(EventArgs) , Panel.ToString() , Panel.AutoSize , Panel.AutoSizeMode , Panel.BorderStyle , Panel.CreateParams , Panel.DefaultSize , Panel.TabStop , Panel.AutoSizeChanged , ScrollableControl.ScrollStateAutoScrolling , ScrollableControl.ScrollStateHScrollVisible , ScrollableControl.ScrollStateVScrollVisible , ScrollableControl.ScrollStateUserHasScrolled , ScrollableControl.ScrollStateFullDrag , ScrollableControl.AdjustFormScrollbars(bool) , ScrollableControl.GetScrollState(int) , ScrollableControl.OnLayout(LayoutEventArgs) , ScrollableControl.OnMouseWheel(MouseEventArgs) , ScrollableControl.OnRightToLeftChanged(EventArgs) , ScrollableControl.OnPaintBackground(PaintEventArgs) , ScrollableControl.OnPaddingChanged(EventArgs) , ScrollableControl.OnVisibleChanged(EventArgs) , ScrollableControl.ScaleControl(SizeF, BoundsSpecified) , ScrollableControl.SetDisplayRectLocation(int, int) , ScrollableControl.ScrollControlIntoView(Control) , ScrollableControl.ScrollToControl(Control) , ScrollableControl.OnScroll(ScrollEventArgs) , ScrollableControl.SetAutoScrollMargin(int, int) , ScrollableControl.SetScrollState(int, bool) , ScrollableControl.WndProc(ref Message) , ScrollableControl.AutoScroll , ScrollableControl.AutoScrollMargin , ScrollableControl.AutoScrollPosition , ScrollableControl.AutoScrollMinSize , ScrollableControl.DisplayRectangle , ScrollableControl.HScroll , ScrollableControl.HorizontalScroll , ScrollableControl.VScroll , ScrollableControl.VerticalScroll , ScrollableControl.Scroll , Control.GetAccessibilityObjectById(int) , Control.SetAutoSizeMode(AutoSizeMode) , Control.GetAutoSizeMode() ,

Control.GetPreferredSize(Size) , Control.AccessibilityNotifyClients(AccessibleEvents, int) ,
Control.AccessibilityNotifyClients(AccessibleEvents, int, int) , Control.BeginInvoke(Delegate) ,
Control.BeginInvoke(Delegate, params object[]) , Control.BringToFront() ,
Control.Contains(Control) , Control.CreateAccessibilityInstance() ,
Control.CreateControlsInstance() , Control.CreateGraphics() , Control.CreateHandle() ,
Control.CreateControl() , Control.DefWndProc(ref Message) , Control.DestroyHandle() ,
Control.Dispose(bool) , Control.DoDragDrop(object, DragDropEffects) ,
Control.DrawToBitmap(Bitmap, Rectangle) , Control.EndInvoke(IAsyncResult) , Control.FindForm() ,
Control.GetTopLevel() , Control.RaiseKeyEvent(object, KeyEventArgs) ,
Control.RaiseMouseEvent(object, MouseEventArgs) , Control.Focus() ,
Control.FromChildHandle(IntPtr) , Control.FromHandle(IntPtr) ,
Control.GetChildAtPoint(Point, GetChildAtPointSkip) , Control.GetChildAtPoint(Point) ,
Control.GetContainerControl() , Control.GetScaledBounds(Rectangle, SizeF, BoundsSpecified) ,
Control.GetNextControl(Control, bool) , Control.GetStyle(ControlStyles) , Control.Hide() ,
Control.InitLayout() , Control.Invalidate(Region) , Control.Invalidate(Region, bool) ,
Control.Invalidate() , Control.Invalidate(bool) , Control.Invalidate(Rectangle) ,
Control.Invalidate(Rectangle, bool) , Control.Invoke(Delegate) ,
Control.Invoke(Delegate, params object[]) , Control.InvokePaint(Control, PaintEventArgs) ,
Control.InvokePaintBackground(Control, PaintEventArgs) , Control.IsKeyLocked(Keys) ,
Control.IsInputChar(char) , Control.IsInputKey(Keys) , Control.IsMnemonic(char, string) ,
Control.NotifyInvalidate(Rectangle) , Control.InvokeOnClick(Control, EventArgs) ,
Control.OnAutoSizeChanged(EventArgs) , Control.OnBackColorChanged(EventArgs) ,
Control.OnBackgroundImageChanged(EventArgs) ,
Control.OnBackgroundImageLayoutChanged(EventArgs) ,
Control.OnBindingContextChanged(EventArgs) , Control.OnCausesValidationChanged(EventArgs) ,
Control.OnContextMenuChanged(EventArgs) , Control.OnContextMenuStripChanged(EventArgs) ,
Control.OnCursorChanged(EventArgs) , Control.OnDockChanged(EventArgs) ,
Control.OnEnabledChanged(EventArgs) , Control.OnFontChanged(EventArgs) ,
Control.OnForeColorChanged(EventArgs) , Control.OnNotifyMessage(Message) ,
Control.OnParentBackColorChanged(EventArgs) ,
Control.OnParentBackgroundImageChanged(EventArgs) ,
Control.OnParentBindingContextChanged(EventArgs) , Control.OnParentCursorChanged(EventArgs) ,
Control.OnParentEnabledChanged(EventArgs) , Control.OnParentFontChanged(EventArgs) ,
Control.OnParentForeColorChanged(EventArgs) , Control.OnParentRightToLeftChanged(EventArgs) ,
Control.OnParentVisibleChanged(EventArgs) , Control.OnPrint(PaintEventArgs) ,
Control.OnTabIndexChanged(EventArgs) , Control.OnTabStopChanged(EventArgs) ,
Control.OnTextChanged(EventArgs) , Control.OnParentChanged(EventArgs) ,
Control.OnClick(EventArgs) , Control.OnClientSizeChanged(EventArgs) ,
Control.OnControlAdded(ControlEventArgs) , Control.OnControlRemoved(ControlEventArgs) ,
Control.OnCreateControl() , Control.OnHandleCreated(EventArgs) ,

Control.OnLocationChanged(EventArgs) , Control.OnHandleDestroyed(EventArgs) ,
Control.OnDoubleClick(EventArgs) , Control.OnDragEnter(DragEventArgs) ,
Control.OnDragOver(DragEventArgs) , Control.OnDragLeave(EventArgs) ,
Control.OnDragDrop(DragEventArgs) , Control.OnGiveFeedback(GiveFeedbackEventArgs) ,
Control.OnEnter(EventArgs) , Control.InvokeGotFocus(Control, EventArgs) ,
Control.OnGotFocus(EventArgs) , Control.OnHelpRequested(HelpEventArgs) ,
Control.OnInvalidated(InvalidateEventArgs) , Control.OnKeyDown(KeyEventArgs) ,
Control.OnKeyPress(KeyPressEventArgs) , Control.OnKeyUp(KeyEventArgs) ,
Control.OnLeave(EventArgs) , Control.InvokeLostFocus(Control, EventArgs) ,
Control.OnLostFocus(EventArgs) , Control.OnMarginChanged(EventArgs) ,
Control.OnMouseDoubleClick(MouseEventArgs) , Control.OnMouseClick(MouseEventArgs) ,
Control.OnMouseCaptureChanged(EventArgs) , Control.OnMouseDown(MouseEventArgs) ,
Control.OnMouseEnter(EventArgs) , Control.OnMouseLeave(EventArgs) ,
Control.OnMouseHover(EventArgs) , Control.OnMouseMove(MouseEventArgs) ,
Control.OnMouseUp(MouseEventArgs) , Control.OnMove(EventArgs) ,
Control.OnPaint(PaintEventArgs) , Control.OnQueryContinueDrag(QueryContinueDragEventArgs) ,
Control.OnRegionChanged(EventArgs) , Control.OnPreviewKeyDown(PreviewKeyDownEventArgs) ,
Control.OnSizeChanged(EventArgs) , Control.OnChangeUICues(UICuesEventArgs) ,
Control.OnStyleChanged(EventArgs) , Control.OnSystemColorsChanged(EventArgs) ,
Control.OnValidating(CancelEventArgs) , Control.OnValidated(EventArgs) ,
Control.PerformLayout() , Control.PerformLayout(Control, string) , Control.PointToClient(Point) ,
Control.PointToScreen(Point) , Control.PreProcessMessage(ref Message) ,
Control.PreProcessControlMessage(ref Message) , Control.ProcessCmdKey(ref Message, Keys) ,
Control.ProcessDialogChar(char) , Control.ProcessDialogKey(Keys) ,
Control.ProcessKeyEventArgs(ref Message) , Control.ProcessKeyMessage(ref Message) ,
Control.ProcessKeyPreview(ref Message) , Control.ProcessMnemonic(char) ,
Control.RaiseDragEvent(object, DragEventArgs) , Control.RaisePaintEvent(object, PaintEventArgs) ,
Control.RecreateHandle() , Control.RectangleToClient(Rectangle) ,
Control.RectangleToScreen(Rectangle) , Control.ReflectMessage(IntPtr, ref Message) ,
Control.Refresh() , Control.ResetMouseEventArgs() , Control.ResetText() ,
Control.ResumeLayout() , Control.ResumeLayout(bool) , Control.Scale(SizeF) , Control.Select() ,
Control.Select(bool, bool) , Control.SelectNextControl(Control, bool, bool, bool, bool) ,
Control.SendToBack() , Control.SetBounds(int, int, int, int) ,
Control.SetBounds(int, int, int, int, BoundsSpecified) ,
Control.SetBoundsCore(int, int, int, int, BoundsSpecified) , Control.SetClientSizeCore(int, int) ,
Control.SizeFromClientSize(Size) , Control.SetStyle(ControlStyles, bool) ,
Control.SetTopLevel(bool) , Control.SetVisibleCore(bool) ,
Control.RtlTranslateAlignment(HorizontalAlignment) ,
Control.RtlTranslateAlignment(LeftRightAlignment) ,
Control.RtlTranslateAlignment(ContentAlignment) ,

Control.RtlTranslateHorizontal(HorizontalAlignment) ,
Control.RtlTranslateLeftRight(LeftRightAlignment) , Control.RtlTranslateContent(ContentAlignment) ,
Control.Show() , Control.SuspendLayout() , Control.Update() , Control.UpdateBounds() ,
Control.UpdateBounds(int, int, int, int) , Control.UpdateBounds(int, int, int, int, int, int) ,
Control.UpdateZOrder() , Control.UpdateStyles() , Control.OnImeModeChanged(EventArgs) ,
Control.AccessibilityObject , Control.AccessibleDefaultActionDescription ,
Control.AccessibleDescription , Control.AccessibleName , Control.AccessibleRole ,
Control.AllowDrop , Control.Anchor , Control.AutoScrollOffset , Control.LayoutEngine ,
Control.BackColor , Control.BackgroundImage , Control.BackgroundImageLayout ,
Control.BindingContext , Control.Bottom , Control.Bounds , Control.CanFocus ,
Control.CanRaiseEvents , Control.CanSelect , Control.Capture , Control.CausesValidation ,
Control.CheckForIllegalCrossThreadCalls , Control.ClientRectangle , Control.ClientSize ,
Control.CompanyName , Control.ContainsFocus , Control.ContextMenu ,
Control.ContextMenuStrip , Control.Controls , Control.Created , Control.Cursor ,
Control.DataBindings , Control.DefaultBackColor , Control.DefaultCursor , Control.DefaultFont ,
Control.DefaultForeColor , Control.DefaultMargin , Control.DefaultMaximumSize ,
Control.DefaultMinimumSize , Control.DefaultPadding , Control.IsDisposed , Control.Disposing ,
Control.Dock , Control.DoubleBuffered , Control.Enabled , Control.Focused , Control.Font ,
Control.FontHeight , Control.ForeColor , Control.Handle , Control.HasChildren , Control.Height ,
Control.IsHandleCreated , Control.InvokeRequired , Control.IsAccessible , Control.IsMirrored ,
Control.Left , Control.Location , Control.Margin , Control.MaximumSize , Control.MinimumSize ,
Control.ModifierKeys , Control.MouseButtons , Control.MousePosition , Control.Name ,
Control.Parent , Control.ProductName , Control.ProductVersion , Control.RecreatingHandle ,
Control.Region , Control.RenderRightToLeft , Control.ResizeRedraw , Control.Right ,
Control.RightToLeft , Control.ScaleChildren , Control.Site , Control.Size , Control.TabIndex ,
Control.Tag , Control.Text , Control.Top , Control.TopLevelControl ,
Control.ShowKeyboardCues , Control.ShowFocusCues , Control.UseWaitCursor , Control.Visible ,
Control.Width , Control.PreferredSize , Control.Padding , Control.CanEnableIme ,
Control.DefaultImeMode , Control.ImeMode , Control.ImeModeBase ,
Control.PropagatingImeMode , Control.BackColorChanged , Control.BackgroundImageChanged ,
Control.BackgroundImageLayoutChanged , Control.BindingContextChanged ,
Control.CausesValidationChanged , Control.ClientSizeChanged , Control.ContextMenuChanged ,
Control.ContextMenuStripChanged , Control.CursorChanged , Control.DockChanged ,
Control.EnabledChanged , Control.FontChanged , Control.ForeColorChanged ,
Control.LocationChanged , Control.MarginChanged , Control.RegionChanged ,
Control.RightToLeftChanged , Control.SizeChanged , Control.TabIndexChanged ,
Control.TabStopChanged , Control.TextChanged , Control.VisibleChanged , Control.Click ,
Control.ControlAdded , Control.ControlRemoved , Control.DragDrop , Control.DragEnter ,
Control.DragOver , Control.DragLeave , Control.GiveFeedback , Control.HandleCreated ,
Control.HandleDestroyed , Control.HelpRequested , Control.Invalidated ,

Control.PaddingChanged , Control.Paint , Control.QueryContinueDrag , Control.QueryAccessibilityHelp , Control.DoubleClick , Control.Enter , Control.GotFocus , Control.KeyDown , Control.KeyPress , Control.KeyUp , Control.Layout , Control.Leave , Control.LostFocus , Control.MouseClick , Control.MouseDoubleClick , Control.MouseCaptureChanged , Control.MouseDown , Control.MouseEnter , Control.MouseLeave , Control.MouseHover , Control.MouseMove , Control.MouseUp , Control.MouseWheel , Control.Move , Control.PreviewKeyDown , Control.Resize , Control.ChangeUICues , Control.StyleChanged , Control.SystemColorsChanged , Control.Validating , Control.Validated , Control.ParentChanged , Control.ImeModeChanged , Component.Dispose() , Component.GetService(Type) , Component.Events , Component.Container , Component.DesignMode , Component.Disposed , MarshalByRefObject.MemberwiseClone(bool) , MarshalByRefObject.GetLifetimeService() , MarshalByRefObject.InitializeLifetimeService() , MarshalByRefObject.CreateObjRef(Type) , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Examples

```
public partial class Form1 : Form
{
    BrowserView webView;

    private IEngine engine;
    private IBrowser browser;

    public Form1()
    {
        webView = new BrowserView() { Dock = DockStyle.Fill };
        InitializeComponent();
        try
        {
            Task.Run(() => {
                engine = EngineFactory.Create(new EngineOptions.Builder()
                {
                    RenderingMode = RenderingMode.HardwareAccelerated
                }.Build());
                browser = engine.CreateBrowser();

            }).ContinueWith((t) =>
            {
                webView.InitializeFrom(browser);
                browser.Navigation.LoadUrl("https://google.com");
            }, TaskScheduler.FromCurrentSynchronizationContext());
```

```
        }
        catch (Exception exception)
        {
            Debug.WriteLine(exception);
        }
        FormClosing += Form1_FormClosing;
        Controls.Add(webView);
    }

    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
        browser?.Dispose();
        engine?.Dispose();
    }
}
```

## Remarks

To bind an instance of this view to a particular browser, use its InitializeFrom(IBrowserView, IBrowser) method:

```
browserView.InitializeFrom(browser);
```

This method should be called from the UI thread.

## Constructors

BrowserView()

Initializes a new instance of the BrowserView class.

## Properties

MenuItems

The collection of the current context menu items.

SaveCreditCardHandler

This handler is used when the user is prompted to save the credit cards in the credit card store, if a different handler was not configured for the browser.

SavePasswordHandler

This handler is used when the user is prompted to save the credentials in the password store, if a different handler was not configured for the browser.

SaveUserDataProfileHandler
This handler is used when the user is prompted to save the user data profiles in the user data profile store, if a different handler was not configured for the browser.

SelectCertificateHandler
This handler is used when the client SSL certificate should be selected, if a different handler was not configured for the browser.

ShowContextMenuHandler
The default context menu handler implementation for this browser view.

StartDownloadHandler
This handler is used when the browser is about to start downloading the file, if a different handler was not configured for the browser.

UpdatePasswordHandler
This handler is used when the user is prompted to update the password in the password store, if a different handler was not configured for the browser.

UpdateUserDataProfileHandler
This handler is used when the user is prompted to update the user data profile in the user data profile store, if a different handler was not configured for the browser.

## Methods

OnFocusRequested(object, EventArgs)
This method is used to handle FocusRequested event.

# Constructor BrowserView

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## BrowserView()

Initializes a new instance of the BrowserView class.

```
public BrowserView()
```

# Property MenuItems

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## MenuItems

The collection of the current context menu items.

```
public ObservableCollection<MenuItemViewModel> MenuItems { get; }
```

## Property Value

ObservableCollection<MenuItemViewModel>

# Property SaveCreditCardHandler

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## SaveCreditCardHandler

This handler is used when the user is prompted to save the credit cards in the credit card store, if a different handler was not configured for the browser.

```
public IHandler<SaveCreditCardParameters, SaveCreditCardResponse> SaveCreditCardHandler {
get; }
```

## Property Value

IHandler<SaveCreditCardParameters, SaveCreditCardResponse>

# Property SavePasswordHandler

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## SavePasswordHandler

This handler is used when the user is prompted to save the credentials in the password store, if a different handler was not configured for the browser.

```
public IHandler<SavePasswordParameters, SavePasswordResponse> SavePasswordHandler { get; }
```

## Property Value

IHandler<SavePasswordParameters, SavePasswordResponse>

# Property SaveUserDataProfileHandler

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## SaveUserDataProfileHandler

This handler is used when the user is prompted to save the user data profiles in the user data profile store, if a different handler was not configured for the browser.

```
public IHandler<SaveUserDataProfileParameters, SaveUserDataProfileResponse>
SaveUserDataProfileHandler { get; }
```

## Property Value

IHandler<SaveUserDataProfileParameters, SaveUserDataProfileResponse>

# Property SelectCertificateHandler

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## SelectCertificateHandler

This handler is used when the client SSL certificate should be selected, if a different handler was not configured for the browser.

```
public IHandler<SelectCertificateParameters, SelectCertificateResponse>
SelectCertificateHandler { get; }
```

## Property Value

IHandler<SelectCertificateParameters, SelectCertificateResponse>

# Property ShowContextMenuHandler

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## ShowContextMenuHandler

The default context menu handler implementation for this browser view.

```
public IHandler<ShowContextMenuParameters, ShowContextMenuResponse> ShowContextMenuHandler
{ get; }
```

## Property Value

IHandler<ShowContextMenuParameters, ShowContextMenuResponse>

# Property StartDownloadHandler

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## StartDownloadHandler

This handler is used when the browser is about to start downloading the file, if a different handler was not configured for the browser.

```
public IHandler<StartDownloadParameters, StartDownloadResponse> StartDownloadHandler {
get; }
```

## Property Value

IHandler<StartDownloadParameters, StartDownloadResponse>

# Property UpdatePasswordHandler

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## UpdatePasswordHandler

This handler is used when the user is prompted to update the password in the password store, if a different handler was not configured for the browser.

```
public IHandler<UpdatePasswordParameters, UpdatePasswordResponse> UpdatePasswordHandler {
get; }
```

## Property Value

IHandler<UpdatePasswordParameters, UpdatePasswordResponse>

# Property UpdateUserDataProfileHandler

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## UpdateUserDataProfileHandler

This handler is used when the user is prompted to update the user data profile in the user data profile store, if a different handler was not configured for the browser.

```
public IHandler<UpdateUserDataProfileParameters, UpdateUserDataProfileResponse>
UpdateUserDataProfileHandler { get; }
```

## Property Value

IHandler<UpdateUserDataProfileParameters, UpdateUserDataProfileResponse>

# Method OnFocusRequested

Namespace: DotNetBrowser.WinForms

Assembly: DotNetBrowser.WinForms.dll

## OnFocusRequested(object, EventArgs)

This method is used to handle FocusRequested event.

```
public void OnFocusRequested(object sender, EventArgs e)
```

## Parameters

`sender` object

the event sender.

`e` EventArgs

the event arguments.

# Namespace DotNetBrowser.WinForms.Dialogs

## Classes

ColorExtensions

Extension methods for performing color conversion.

DefaultAuthenticationHandler

Default WinForms authentication handler, which displays an authentication dialog.

DefaultStartDownloadHandler

The default WinForms implementation of StartDownloadHandler, which displays a save file dialog to specify the path to store the downloaded file.

DialogHandlerBase

The base class for all dialog handlers.

# Class ColorExtensions

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

Extension methods for performing color conversion.

```
public static class ColorExtensions
```

**Inheritance**

object ← ColorExtensions

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Methods

ToColor(Color)

Converts the Color instance to Color.

ToUIColor(Color)

Converts the Color instance to Color.

# Method ToColor

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## ToColor(Color)

Converts the Color instance to Color.

```
public static Color ToColor(this Color color)
```

## Parameters

`color` Color

the color to convert

## Returns

Color

the converted color.

# Method ToUIColor

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## ToUIColor(Color)

Converts the Color instance to Color.

```
public static Color ToUIColor(this Color color)
```

## Parameters

`color` Color

the color to convert

## Returns

Color

the converted color.

# Class DefaultAuthenticationHandler

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

Default WinForms authentication handler, which displays an authentication dialog.

```
public class DefaultAuthenticationHandler : DialogHandlerBase,
IHandler<AuthenticateParameters, AuthenticateResponse>
```

**Inheritance**

object ← DialogHandlerBase ← DefaultAuthenticationHandler

**Implements**

IHandler<AuthenticateParameters, AuthenticateResponse>

**Inherited Members**

DialogHandlerBase.Parent , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Constructors

DefaultAuthenticationHandler(Control)

Creates and initializes the dialog handler.

## Methods

Handle(AuthenticateParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

# Constructor DefaultAuthenticationHandler

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## DefaultAuthenticationHandler(Control)

Creates and initializes the dialog handler.

```
public DefaultAuthenticationHandler(Control parent)
```

## Parameters

`parent` Control

The parent object for the dialog displayed by this handler.

# Method Handle

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## Handle(AuthenticateParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

```
public AuthenticateResponse Handle(AuthenticateParameters parameters)
```

## Parameters

`parameters` AuthenticateParameters

the handler parameters.

## Returns

AuthenticateResponse

an object that represents the response that should be determined based on the provided parameters.

# Class DefaultStartDownloadHandler

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

The default WinForms implementation of StartDownloadHandler, which displays a save file dialog to specify the path to store the downloaded file.

```
public class DefaultStartDownloadHandler : DialogHandlerBase,
IHandler<StartDownloadParameters, StartDownloadResponse>
```

**Inheritance**

object ← DialogHandlerBase ← DefaultStartDownloadHandler

**Implements**

IHandler<StartDownloadParameters, StartDownloadResponse>

**Inherited Members**

DialogHandlerBase.Parent , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Constructors

DefaultStartDownloadHandler(Control)

Creates and initializes the dialog handler.

## Properties

InitialDirectory

The initial directory that is displayed by the file dialog.

## Methods

Handle(StartDownloadParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

# Constructor DefaultStartDownloadHandler

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## DefaultStartDownloadHandler(Control)

Creates and initializes the dialog handler.

```
public DefaultStartDownloadHandler(Control parent)
```

## Parameters

`parent` Control

The parent object for the dialog displayed by this handler.

# Property InitialDirectory

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## InitialDirectory

The initial directory that is displayed by the file dialog.

```
public string InitialDirectory { get; set; }
```

## Property Value

string

# Method Handle

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## Handle(StartDownloadParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

```
public StartDownloadResponse Handle(StartDownloadParameters startDownloadParameters)
```

## Parameters

`startDownloadParameters` StartDownloadParameters

the item for which the download is about to be started.

## Returns

StartDownloadResponse

an object that represents the response that should be determined based on the provided parameters.

# Class DialogHandlerBase

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

The base class for all dialog handlers.

```
public class DialogHandlerBase
```

**Inheritance**

object ← DialogHandlerBase

**Derived**

DefaultAuthenticationHandler, DefaultStartDownloadHandler

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Constructors

DialogHandlerBase(Control)

Creates and initializes the dialog handler.

## Properties

Parent

The parent object for the dialog displayed by this handler. This object is used to locate the owner window for the modal dialogs.

# Constructor DialogHandlerBase

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## DialogHandlerBase(Control)

Creates and initializes the dialog handler.

```
public DialogHandlerBase(Control parent)
```

## Parameters

`parent` Control

The parent object for the dialog displayed by this handler.

# Property Parent

Namespace: DotNetBrowser.WinForms.Dialogs

Assembly: DotNetBrowser.WinForms.dll

## Parent

The parent object for the dialog displayed by this handler. This object is used to locate the owner window for the modal dialogs.

```
public Control Parent { get; }
```

## Property Value

Control

# Namespace DotNetBrowser.WinForms.Extensions

## Classes

DefaultOpenExtensionPopupHandler

Default extension popup handler, which displays an extension popup as a separate window.

# Class DefaultOpenExtensionPopupHandler

Namespace: DotNetBrowser.WinForms.Extensions

Assembly: DotNetBrowser.WinForms.dll

Default extension popup handler, which displays an extension popup as a separate window.

```
public sealed class DefaultOpenExtensionPopupHandler :
IHandler<OpenExtensionPopupParameters>
```

**Inheritance**

object ← DefaultOpenExtensionPopupHandler

**Implements**

IHandler<OpenExtensionPopupParameters>

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType()

## Constructors

DefaultOpenExtensionPopupHandler(Control)

Creates and initializes a new instance of DefaultOpenExtensionPopupHandler.

## Methods

Handle(OpenExtensionPopupParameters)

This method is called when Chromium is about to display the extension popup window.

# Constructor DefaultOpenExtensionPopupHandler

Namespace: DotNetBrowser.WinForms.Extensions

Assembly: DotNetBrowser.WinForms.dll

## DefaultOpenExtensionPopupHandler(Control)

Creates and initializes a new instance of DefaultOpenExtensionPopupHandler.

```
public DefaultOpenExtensionPopupHandler(Control parent)
```

## Parameters

`parent` Control

The parent object for the window displayed by this handler.

# Method Handle

Namespace: DotNetBrowser.WinForms.Extensions

Assembly: DotNetBrowser.WinForms.dll

## Handle(OpenExtensionPopupParameters)

This method is called when Chromium is about to display the extension popup window.

```
public void Handle(OpenExtensionPopupParameters parameters)
```

## Parameters

`parameters` OpenExtensionPopupParameters

The extension popup parameters.

# Namespace DotNetBrowser.Wpf

## Classes

BitmapExtensions

Extensions for Bitmap class.

BrowserView

The Wpf-based implementation of browser view.

# Class BitmapExtensions

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

Extensions for Bitmap class.

```
public static class BitmapExtensions
```

**Inheritance**

object ← BitmapExtensions

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Methods

ToBitmap(Bitmap)

Converts the Bitmap instance to Bitmap.

ToBitmapSource(Bitmap)

Converts the Bitmap instance to BitmapSource that can be then used in building the UI, e.g. displaying icons.

# Method ToBitmap

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## ToBitmap(Bitmap)

Converts the Bitmap instance to Bitmap.

```
public static Bitmap ToBitmap(this Bitmap bitmap)
```

### Parameters

`bitmap` Bitmap

the bitmap to convert

### Returns

Bitmap

the converted bitmap.

# Method ToBitmapSource

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## ToBitmapSource(Bitmap)

Converts the Bitmap instance to BitmapSource that can be then used in building the UI, e.g. displaying icons.

```
public static BitmapSource ToBitmapSource(this Bitmap bitmap)
```

### Parameters

`bitmap` Bitmap

the bitmap to convert

### Returns

BitmapSource

the converted bitmap.

# Class BrowserView

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

The Wpf-based implementation of browser view.

```
public class BrowserView : IBrowserView
```

### Inheritance

object ← BrowserView

### Implements

IBrowserView

### Inherited Members

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Examples

```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApp1"
        xmlns:wpf="clr-namespace:DotNetBrowser.Wpf;assembly=DotNetBrowser.Wpf"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800" Closing="Window_Closing">
    <Grid>
        <wpf:BrowserView Name ="BrowserView" HorizontalAlignment="Stretch"
VerticalAlignment="Stretch"/>
    </Grid>
</Window>
```

```
public partial class MainWindow : Window
{
```

```
    private IEngine engine;
    private IBrowser browser;

    public MainWindow()
    {
        try
        {
            Task.Run(() => {
                engine = EngineFactory.Create(new EngineOptions.Builder()
                {
                    RenderingMode = RenderingMode.HardwareAccelerated
                }.Build());
                browser = engine.CreateBrowser();

            }).ContinueWith((t) =>
            {
                BrowserView.InitializeFrom(browser);
                browser.Navigation.LoadUrl("https://www.teamdev.com/");
            }, TaskScheduler.FromCurrentSynchronizationContext());

            InitializeComponent();
        }
        catch (Exception exception)
        {
            Debug.WriteLine(exception);
        }
    }

    private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
    {
        engine.Dispose();
    }
}
```

## Remarks

To bind an instance of this view to a particular browser, use its InitializeFrom(IBrowserView, IBrowser) method:

`browserView.InitializeFrom(browser);`

This method should be called from the UI thread.

## Constructors

BrowserView()
Initializes a new instance of the BrowserView class.

## Properties

MenuItems
The collection of the current context menu items.

SaveCreditCardHandler
This handler is used when the user is prompted to save the credit cards in the credit card store, if a different handler was not configured for the browser.

SavePasswordHandler
This handler is used when the user is prompted to save the credentials in the password store, if a different handler was not configured for the browser.

SaveUserDataProfileHandler
This handler is used when the user is prompted to save the user data profiles in the user data profile store, if a different handler was not configured for the browser.

SelectCertificateHandler
This handler is used when the client SSL certificate should be selected, if a different handler was not configured for the browser.

ShowContextMenuHandler
The default context menu handler implementation for this browser view.

StartDownloadHandler
This handler is used when the browser is about to start downloading the file, if a different handler was not configured for the browser.

UpdatePasswordHandler
This handler is used when the user is prompted to update the password in the password store, if a different handler was not configured for the browser.

UpdateUserDataProfileHandler
This handler is used when the user is prompted to update the user data profile in the user data profile store, if a different handler was not configured for the browser.

# Methods

OnFocusRequested(object, EventArgs)

This method is used to handle FocusRequested event.

# Constructor BrowserView

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## BrowserView()

Initializes a new instance of the BrowserView class.

```
public BrowserView()
```

# Property MenuItems

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## MenuItems

The collection of the current context menu items.

```
public ObservableCollection<MenuItemViewModel> MenuItems { get; }
```

## Property Value

ObservableCollection<MenuItemViewModel>

# Property SaveCreditCardHandler

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## SaveCreditCardHandler

This handler is used when the user is prompted to save the credit cards in the credit card store, if a different handler was not configured for the browser.

```
public IHandler<SaveCreditCardParameters, SaveCreditCardResponse> SaveCreditCardHandler {
get; }
```

## Property Value

IHandler<SaveCreditCardParameters, SaveCreditCardResponse>

# Property SavePasswordHandler

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## SavePasswordHandler

This handler is used when the user is prompted to save the credentials in the password store, if a different handler was not configured for the browser.

```
public IHandler<SavePasswordParameters, SavePasswordResponse> SavePasswordHandler { get; }
```

## Property Value

IHandler<SavePasswordParameters, SavePasswordResponse>

# Property SaveUserDataProfileHandler

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## SaveUserDataProfileHandler

This handler is used when the user is prompted to save the user data profiles in the user data profile store, if a different handler was not configured for the browser.

```
public IHandler<SaveUserDataProfileParameters, SaveUserDataProfileResponse>
SaveUserDataProfileHandler { get; }
```

## Property Value

IHandler<SaveUserDataProfileParameters, SaveUserDataProfileResponse>

# Property SelectCertificateHandler

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## SelectCertificateHandler

This handler is used when the client SSL certificate should be selected, if a different handler was not configured for the browser.

```
public IHandler<SelectCertificateParameters, SelectCertificateResponse>
SelectCertificateHandler { get; }
```

## Property Value

IHandler<SelectCertificateParameters, SelectCertificateResponse>

# Property ShowContextMenuHandler

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## ShowContextMenuHandler

The default context menu handler implementation for this browser view.

```
public IHandler<ShowContextMenuParameters, ShowContextMenuResponse> ShowContextMenuHandler
{ get; }
```

## Property Value

IHandler<ShowContextMenuParameters, ShowContextMenuResponse>

# Property StartDownloadHandler

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## StartDownloadHandler

This handler is used when the browser is about to start downloading the file, if a different handler was not configured for the browser.

```
public IHandler<StartDownloadParameters, StartDownloadResponse> StartDownloadHandler {
get; }
```

## Property Value

IHandler<StartDownloadParameters, StartDownloadResponse>

# Property UpdatePasswordHandler

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## UpdatePasswordHandler

This handler is used when the user is prompted to update the password in the password store, if a different handler was not configured for the browser.

```
public IHandler<UpdatePasswordParameters, UpdatePasswordResponse> UpdatePasswordHandler {
get; }
```

## Property Value

IHandler<UpdatePasswordParameters, UpdatePasswordResponse>

# Property UpdateUserDataProfileHandler

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## UpdateUserDataProfileHandler

This handler is used when the user is prompted to update the user data profile in the user data profile store, if a different handler was not configured for the browser.

```
public IHandler<UpdateUserDataProfileParameters, UpdateUserDataProfileResponse>
UpdateUserDataProfileHandler { get; }
```

## Property Value

IHandler<UpdateUserDataProfileParameters, UpdateUserDataProfileResponse>

# Method OnFocusRequested

Namespace: DotNetBrowser.Wpf

Assembly: DotNetBrowser.Wpf.dll

## OnFocusRequested(object, EventArgs)

This method is used to handle FocusRequested event.

```
public void OnFocusRequested(object sender, EventArgs e)
```

## Parameters

`sender` object

the event sender.

`e` EventArgs

the event arguments.

# Namespace DotNetBrowser.Wpf.ContextMenu

## Classes

MenuItemViewModel

# Class MenuItemViewModel

Namespace: DotNetBrowser.Wpf.ContextMenu

Assembly: DotNetBrowser.Wpf.dll

```
public abstract class MenuItemViewModel
```

**Inheritance**

object ← MenuItemViewModel

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Properties

Command

Header

IsCheckable

IsChecked

IsEnabled

MenuItems

# Property Command

Namespace: DotNetBrowser.Wpf.ContextMenu

Assembly: DotNetBrowser.Wpf.dll

## Command

```
public ICommand Command { get; }
```

## Property Value

ICommand

# Property Header

Namespace: DotNetBrowser.Wpf.ContextMenu

Assembly: DotNetBrowser.Wpf.dll

## Header

```
public string Header { get; }
```

## Property Value

string

# Property IsCheckable

Namespace: DotNetBrowser.Wpf.ContextMenu

Assembly: DotNetBrowser.Wpf.dll

## IsCheckable

```
public bool IsCheckable { get; }
```

### Property Value

bool

# Property IsChecked

Namespace: DotNetBrowser.Wpf.ContextMenu

Assembly: DotNetBrowser.Wpf.dll

## IsChecked

```
public bool IsChecked { get; }
```

### Property Value

bool

# Property IsEnabled

Namespace: DotNetBrowser.Wpf.ContextMenu

Assembly: DotNetBrowser.Wpf.dll

## IsEnabled

```
public bool IsEnabled { get; }
```

### Property Value

bool

# Property MenuItems

Namespace: DotNetBrowser.Wpf.ContextMenu

Assembly: DotNetBrowser.Wpf.dll

## MenuItems

```
public ObservableCollection<MenuItemViewModel> MenuItems { get; }
```

## Property Value

ObservableCollection<MenuItemViewModel>

# Namespace DotNetBrowser.Wpf.Dialogs

## Classes

ColorExtensions

Extension methods for performing color conversion.

DefaultAuthenticationHandler

Default Wpf authentication handler, which displays an authentication dialog.

DefaultStartDownloadHandler

The default Wpf implementation of StartDownloadHandler, which displays a save file dialog to specify the path to store the downloaded file.

DialogHandlerBase

The base class for all dialog handlers.

# Class ColorExtensions

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

Extension methods for performing color conversion.

```
public static class ColorExtensions
```

**Inheritance**

object ← ColorExtensions

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Methods

ToColor(Color)

Converts the Color instance to Color.

ToUIColor(Color)

Converts the Color instance to Color.

# Method ToColor

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## ToColor(Color)

Converts the Color instance to Color.

```
public static Color ToColor(this Color color)
```

### Parameters

`color` Color

the color to convert

### Returns

Color

the converted color.

# Method ToUIColor

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## ToUIColor(Color)

Converts the Color instance to Color.

```
public static Color ToUIColor(this Color color)
```

## Parameters

`color` Color

the color to convert

## Returns

Color

the converted color.

# Class DefaultAuthenticationHandler

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

Default Wpf authentication handler, which displays an authentication dialog.

```
public class DefaultAuthenticationHandler : DialogHandlerBase,
IHandler<AuthenticateParameters, AuthenticateResponse>
```

**Inheritance**

object ← DialogHandlerBase ← DefaultAuthenticationHandler

**Implements**

IHandler<AuthenticateParameters, AuthenticateResponse>

**Inherited Members**

DialogHandlerBase.Parent , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Constructors

DefaultAuthenticationHandler(DependencyObject)

Creates and initializes the dialog handler.

## Methods

Handle(AuthenticateParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

# Constructor DefaultAuthenticationHandler

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## DefaultAuthenticationHandler(DependencyObject)

Creates and initializes the dialog handler.

```
public DefaultAuthenticationHandler(DependencyObject parent)
```

## Parameters

`parent` DependencyObject

The parent object for the dialog displayed by this handler.

# Method Handle

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## Handle(AuthenticateParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

```
public AuthenticateResponse Handle(AuthenticateParameters parameters)
```

### Parameters

`parameters` AuthenticateParameters

the handler parameters.

### Returns

AuthenticateResponse

an object that represents the response that should be determined based on the provided parameters.

# Class DefaultStartDownloadHandler

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

The default Wpf implementation of StartDownloadHandler, which displays a save file dialog to specify the path to store the downloaded file.

```
public class DefaultStartDownloadHandler : DialogHandlerBase,
IHandler<StartDownloadParameters, StartDownloadResponse>
```

**Inheritance**

object ← DialogHandlerBase ← DefaultStartDownloadHandler

**Implements**

IHandler<StartDownloadParameters, StartDownloadResponse>

**Inherited Members**

DialogHandlerBase.Parent , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Constructors

DefaultStartDownloadHandler(DependencyObject)

Creates and initializes the dialog handler.

## Properties

InitialDirectory

The initial directory that is displayed by the file dialog.

## Methods

Handle(StartDownloadParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

# Constructor DefaultStartDownloadHandler

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## DefaultStartDownloadHandler(DependencyObject)

Creates and initializes the dialog handler.

```
public DefaultStartDownloadHandler(DependencyObject parent)
```

## Parameters

`parent` DependencyObject

The parent object for the dialog displayed by this handler.

# Property InitialDirectory

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## InitialDirectory

The initial directory that is displayed by the file dialog.

```
public string InitialDirectory { get; set; }
```

## Property Value

string

# Method Handle

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## Handle(StartDownloadParameters)

This method is called when the Chromium callback needs a response that may be determined based on the provided parameters.

```
public StartDownloadResponse Handle(StartDownloadParameters startDownloadParameters)
```

### Parameters

`startDownloadParameters` StartDownloadParameters

the item for which the download is about to be started.

### Returns

StartDownloadResponse

an object that represents the response that should be determined based on the provided parameters.

# Class DialogHandlerBase

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

The base class for all dialog handlers.

```
public class DialogHandlerBase
```

**Inheritance**

object ← DialogHandlerBase

**Derived**

DefaultAuthenticationHandler, DefaultStartDownloadHandler

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone()

## Constructors

DialogHandlerBase(DependencyObject)

Creates and initializes the dialog handler.

## Properties

Parent

The parent object for the dialog displayed by this handler. This object is used to locate the owner window for the modal dialogs.

# Constructor DialogHandlerBase

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## DialogHandlerBase(DependencyObject)

Creates and initializes the dialog handler.

```
public DialogHandlerBase(DependencyObject parent)
```

## Parameters

`parent` DependencyObject

The parent object for the dialog displayed by this handler.

# Property Parent

Namespace: DotNetBrowser.Wpf.Dialogs

Assembly: DotNetBrowser.Wpf.dll

## Parent

The parent object for the dialog displayed by this handler. This object is used to locate the owner window for the modal dialogs.

```
public DependencyObject Parent { get; }
```

## Property Value

DependencyObject

# Namespace DotNetBrowser.Wpf.Extensions

## Classes

DefaultOpenExtensionPopupHandler

Default extension popup handler, which displays an extension popup as a separate window.

# Class DefaultOpenExtensionPopupHandler

Namespace: DotNetBrowser.Wpf.Extensions

Assembly: DotNetBrowser.Wpf.dll

Default extension popup handler, which displays an extension popup as a separate window.

```
public sealed class DefaultOpenExtensionPopupHandler :
IHandler<OpenExtensionPopupParameters>
```

**Inheritance**

object ← DefaultOpenExtensionPopupHandler

**Implements**

IHandler<OpenExtensionPopupParameters>

**Inherited Members**

object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode() , object.GetType()

## Constructors

DefaultOpenExtensionPopupHandler(FrameworkElement)

Creates and initializes a new instance of DefaultOpenExtensionPopupHandler.

## Methods

Handle(OpenExtensionPopupParameters)

This method is called when Chromium is about to display the extension popup window.

# Constructor DefaultOpenExtensionPopupHandler

Namespace: DotNetBrowser.Wpf.Extensions

Assembly: DotNetBrowser.Wpf.dll

## DefaultOpenExtensionPopupHandler(FrameworkElement)

Creates and initializes a new instance of DefaultOpenExtensionPopupHandler.

```
public DefaultOpenExtensionPopupHandler(FrameworkElement parent)
```

## Parameters

`parent` FrameworkElement

The parent object for the window displayed by this handler.

# Method Handle

Namespace: DotNetBrowser.Wpf.Extensions

Assembly: DotNetBrowser.Wpf.dll

## Handle(OpenExtensionPopupParameters)

This method is called when Chromium is about to display the extension popup window.

```
public void Handle(OpenExtensionPopupParameters parameters)
```

## Parameters

`parameters` OpenExtensionPopupParameters

The extension popup parameters.

# Namespace DotNetBrowser.Zoom

## Classes

Level

A zoom level of a web page. Provides a set of the pre-defined constants. Each constant name consists of the `P` prefix followed by a number. The number represents a zoom level in percents. For example, the P25 and P110 values represent the 25% and 110% zoom levels accordingly.

## Interfaces

IZoom

Allows zooming content of a web page.

IZoomLevels

Allows configuring the default zoom levels and receive notifications about zoom changes.

# Interface IZoom

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

Allows zooming content of a web page.

```
public interface IZoom : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Remarks

A IZoom instance belongs to a IBrowser instance and can be used only if the IBrowser instance is alive. When the IBrowser instance is closed, IZoom instance automatically updates its internal state and doesn't allow modifying zoom anymore. The ObjectDisposedException will be thrown in this case.

The zoom level is configured for each domain separately, so if you set zoom level for the a.com web page, it will not be applied for the b.com web page. If you change zoom level for one domain and then load another one, then the zoom level for another domain will be default.

## Properties

Browser
: Gets the IBrowser instance associated with this object.

Enabled
: Enables or disables zoom changes.

Level
: Gets or sets the zoom level for the currently loaded web page.

## Methods

In()
: Updates the zoom level for the currently loaded web page on one step up.

Out()

Updates the zoom level for the currently loaded web page on one step down.

Reset()

Resets the zoom level for the currently loaded web page to default value.

# Property Browser

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## Browser

Gets the IBrowser instance associated with this object.

```
IBrowser Browser { get; }
```

## Property Value

IBrowser

The IBrowser instance associated with this object.

# Property Enabled

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## Enabled

Enables or disables zoom changes.

```
bool Enabled { get; set; }
```

## Property Value

bool

`true` if zoom functionality is enabled, `false` otherwise.

## Remarks

By default, zoom functionality is enabled. You can change zoom level for the currently loaded web page.

To disable zoom functionality, set this property to `false`. In this case, the browser instance will revert zoom to the default level, and all attempts to change zoom will be ignored.

## Exceptions

ObjectDisposedException

The IZoom has already been disposed.

# Property Level

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## Level

Gets or sets the zoom level for the currently loaded web page.

```
Level Level { get; set; }
```

## Property Value

Level

The zoom level.

## Remarks

Zoom level is configured for each domain separately. For example, if you load the www.a.com web page and set zoom level to P250, then load the www.b.org web page, the zoom level for www.b.org web page will be reset to default value. When you load the www.a.com web page again, its zoom level will be restored to P250 automatically.

## Exceptions

ObjectDisposedException

The IZoom has already been disposed.

# Method In

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## In()

Updates the zoom level for the currently loaded web page on one step up.

```
void In()
```

## Exceptions

ObjectDisposedException

The IZoom has already been disposed.

# Method Out

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## Out()

Updates the zoom level for the currently loaded web page on one step down.

```
void Out()
```

## Exceptions

ObjectDisposedException

The IZoom has already been disposed.

# Method Reset

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## Reset()

Resets the zoom level for the currently loaded web page to default value.

```
void Reset()
```

## Exceptions

ObjectDisposedException

The IZoom has already been disposed.

# Interface IZoomLevels

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

Allows configuring the default zoom levels and receive notifications about zoom changes.

```
public interface IZoomLevels : IAutoDisposable
```

**Inherited Members**
IAutoDisposable.IsDisposed , IAutoDisposable.Disposed

## Properties

DefaultLevel
Gets or sets the default zoom level for pages that don't override it.

Engine
Gets the IEngine instance associated with this object.

Profile
Gets the IProfile instance associated with this object.

## Events

LevelChanged
Occurs when the zoom level for a specific URL has been changed.

# Property DefaultLevel

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## DefaultLevel

Gets or sets the default zoom level for pages that don't override it.

```
Level DefaultLevel { get; set; }
```

## Property Value

Level

The default zoom level for the web pages. By default, zoom level is P100

## Exceptions

ObjectDisposedException

The IZoomLevels has already been disposed.

# Property Engine

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## Engine

Gets the IEngine instance associated with this object.

```
IEngine Engine { get; }
```

## Property Value

IEngine

The IEngine instance associated with this object.

# Property Profile

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## Profile

Gets the IProfile instance associated with this object.

```
IProfile Profile { get; }
```

## Property Value

IProfile

The IProfile instance associated with this object.

# Event LevelChanged

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

Occurs when the zoom level for a specific URL has been changed.

```
event EventHandler<LevelChangedEventArgs> LevelChanged
```

## Returns

EventHandler<LevelChangedEventArgs>

Occurs when the zoom level for a specific URL has been changed.

# Class Level

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

A zoom level of a web page. Provides a set of the pre-defined constants. Each constant name consists of the `P` prefix followed by a number. The number represents a zoom level in percents. For example, the P25 and P110 values represent the 25% and 110% zoom levels accordingly.

```
public sealed class Level : TypedEnum<double>
```

**Inheritance**

object ← TypedEnum<double> ← Level

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

## Fields

P100

100% (default zoom level)

P110

110%

P125

125%

P150

150%

P175

175%

P200

200%

P25

25%

P250
250%

P300
300%

P33
33%

P400
400%

P50
50%

P500
500%

P67
67%

P75
75%

P80
80%

P90
90%

## Methods

Create(double)
Initializes a new Level instance from its double representation.

## Operators

explicit operator Level(double)
Initializes a new Level instance from its double representation.

# Field P100

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

100% (default zoom level)

```
public static readonly Level P100
```

## Returns

Level

100% (default zoom level)

# Field P110

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

110%

```
public static readonly Level P110
```

## Returns

Level

110%

# Field P125

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

125%

```
public static readonly Level P125
```

## Returns

Level

125%

# Field P150

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

150%

```
public static readonly Level P150
```

## Returns

Level

150%

# Field P175

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

175%

```
public static readonly Level P175
```

## Returns

Level

175%

# Field P200

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

200%

```
public static readonly Level P200
```

## Returns

Level

200%

# Field P25

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

25%

```
public static readonly Level P25
```

## Returns

Level

25%

# Field P250

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

250%

```
public static readonly Level P250
```

## Returns

Level

250%

# Field P300

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

300%

```
public static readonly Level P300
```

## Returns

Level

300%

# Field P33

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

33%

```
public static readonly Level P33
```

## Returns

Level

33%

# Field P400

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

400%

```
public static readonly Level P400
```

## Returns

Level

400%

# Field P50

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

50%

```
public static readonly Level P50
```

## Returns

Level

50%

# Field P500

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

500%

```
public static readonly Level P500
```

## Returns

Level

500%

# Field P67

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

67%

```
public static readonly Level P67
```

## Returns

Level

67%

# Field P75

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

75%

```
public static readonly Level P75
```

## Returns

Level

75%

# Field P80

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

80%

```
public static readonly Level P80
```

## Returns

Level

80%

# Field P90

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

90%

```
public static readonly Level P90
```

## Returns

Level

90%

# Method Create

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## Create(double)

Initializes a new Level instance from its double representation.

```
public static Level Create(double value)
```

## Parameters

`value` double

The level representation.

## Returns

Level

The initialized Level instance.

# Operator explicit operator

Namespace: DotNetBrowser.Zoom

Assembly: DotNetBrowser.dll

## explicit operator Level(double)

Initializes a new Level instance from its double representation.

```
public static explicit operator Level(double value)
```

### Parameters

`value` double

The level representation.

### Returns

Level

The initialized Level instance.

# Namespace DotNetBrowser.Zoom.Events

## Classes

LevelChangedEventArgs

The event arguments for the LevelChanged event.

# Class LevelChangedEventArgs

Namespace: DotNetBrowser.Zoom.Events

Assembly: DotNetBrowser.dll

The event arguments for the LevelChanged event.

```
public class LevelChangedEventArgs : EventArgs
```

**Inheritance**

object ← EventArgs ← LevelChangedEventArgs

**Inherited Members**

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

## Properties

Host

Gets the host of the URL of the web page which zoom level has been changed.

Level

Gets the new zoom level of a web page.

ZoomLevels

Gets the IZoomLevels instance associated with this object.

# Property Host

Namespace: DotNetBrowser.Zoom.Events

Assembly: DotNetBrowser.dll

## Host

Gets the host of the URL of the web page which zoom level has been changed.

```
public string Host { get; }
```

## Property Value

string

The host of the URL of the web page which zoom level has been changed.

# Property Level

Namespace: DotNetBrowser.Zoom.Events

Assembly: DotNetBrowser.dll

## Level

Gets the new zoom level of a web page.

```
public Level Level { get; }
```

## Property Value

Level

The new zoom level of a web page.

# Property ZoomLevels

Namespace: DotNetBrowser.Zoom.Events

Assembly: DotNetBrowser.dll

## ZoomLevels

Gets the IZoomLevels instance associated with this object.

```
public IZoomLevels ZoomLevels { get; }
```

## Property Value

IZoomLevels

The IZoomLevels instance associated with this object.