



Workshop

Angular Advanced



Accessibility (a11y)

Allow everyone to use your applications

Making your websites
usable by as many people as possible

Why / What you'll learn



- Semantic HTML, which improves accessibility, also improves SEO, making your site more findable.
- Ethics and morals
- More usable by other groups
- It is also the law in some places

Disability

Overview about permanent and temporary Disabilities

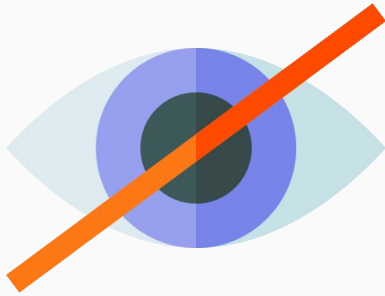
People with disabilities

- **Visual:** Blind, low-vision, color blind
- **Hearing:** Deaf, hard-of-hearing
- **Motor:** spinal cord injury, MS, Cerebral palsy, ALS
- **Cognitive:** Autism, learning, TBI, memory, attention

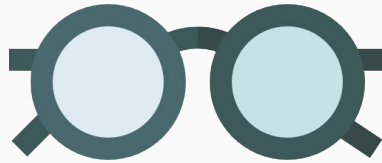
Visual Disability

Permanent

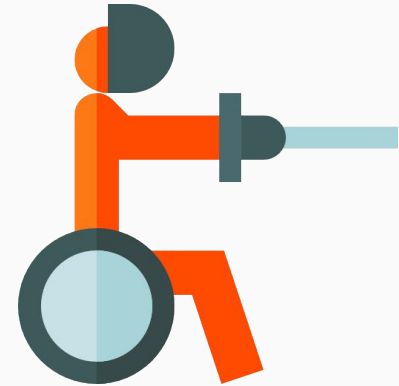
Temporary



(Color) Blind



Sunglasses

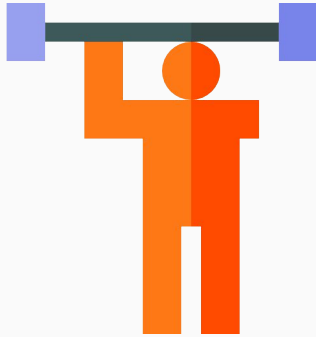


Helmet

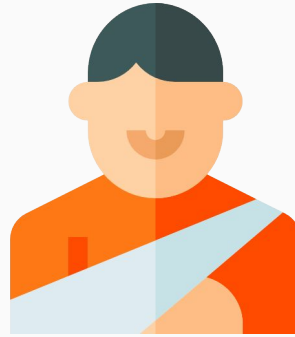
Motor disabilities

Permanent

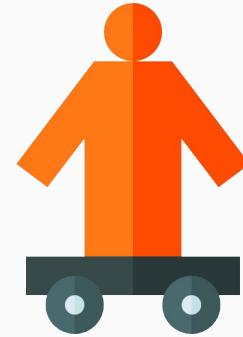
Temporary



No Hand



Broken Hand

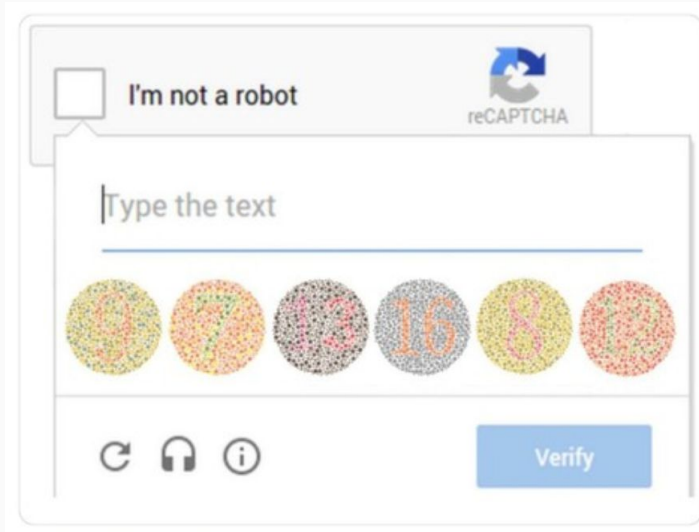


On the move

Color and Contrast

Color Blind People

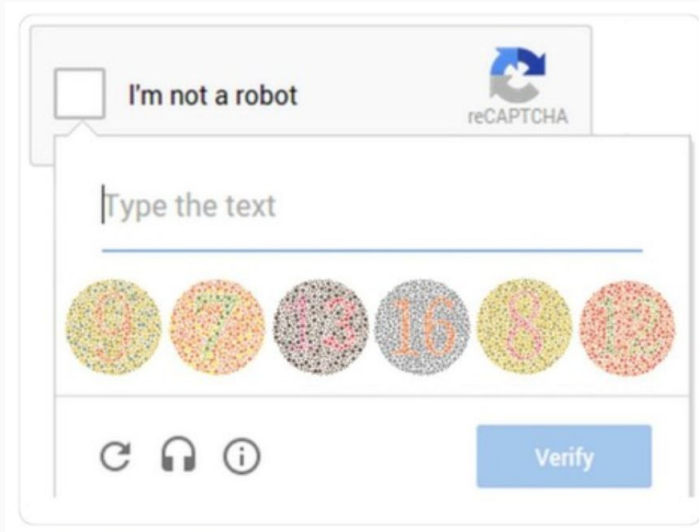
Original



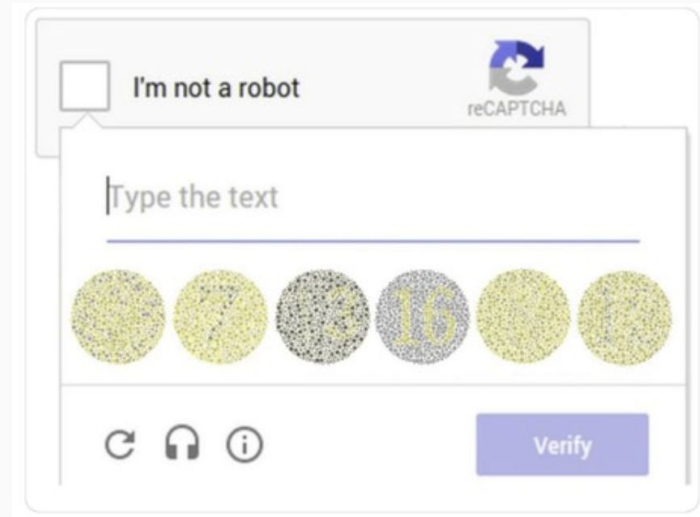
The image shows the original reCAPTCHA interface. At the top left, there is a checkbox labeled "I'm not a robot" and the reCAPTCHA logo. Below this is a text input field with the placeholder text "Type the text". Underneath the input field are six circular images containing a mix of colored dots and numbers. The numbers visible are 16, 1, and 10. At the bottom left, there are three icons: a refresh icon, a headphones icon, and an information icon. At the bottom right, there is a blue button labeled "Verify".

Color Blind People

Original



Red-Blind People



Chrome Extension: Colorblinding

[Home](#) > [Extensions](#) > Colorblinding



Colorblinding

Offered by: leocardz.com

★★★★☆ 31 | [Developer Tools](#) |  10,310 users

Add to Chrome

Chrome Extension: Colorblinding

The screenshot shows the Netflix homepage with a Chrome extension overlay on the right side. The extension menu is open, displaying a list of color vision simulation options. The 'Blue-Blind / Tritanopia' option is selected, indicated by a blue dot. Below the list is the 'Colorblinding' logo. The background shows the Netflix interface with the 'NETFLIX' logo, navigation links ('Navegar', 'KIDS', 'Descubra'), and a recommendation carousel featuring titles like 'Breaking Bad' and 'Orange Is the New Black'. A 'Recomendar' button is visible below the carousel. Below the carousel, the 'Em alta' section displays a row of popular titles including 'Gossip Girl', 'That '70s Show', 'Community', 'House', and 'Battlestar Galactica'.

www.netflix.com/MyHome

NETFLIX Navegar KIDS Descubra

Quer ideias rápidas para assistir JÁ?

- Normal Color Vision
- Red-Blind / Protanopia
- Green-Blind / Deuteranopia
- Blue-Blind / Tritanopia
- Red-Weak / Protanomaly
- Green-Weak / Deuteranomaly
- Blue-Weak / Tritanomaly
- Monochromacy / Achromatopsia
- Blue Cone Monochromacy / Achromatomaly

Colorblinding

Recomendar

Em alta

THE RETURNED
NEW EPISODE
WEEKLY

That '70s Show

COMMUNITY

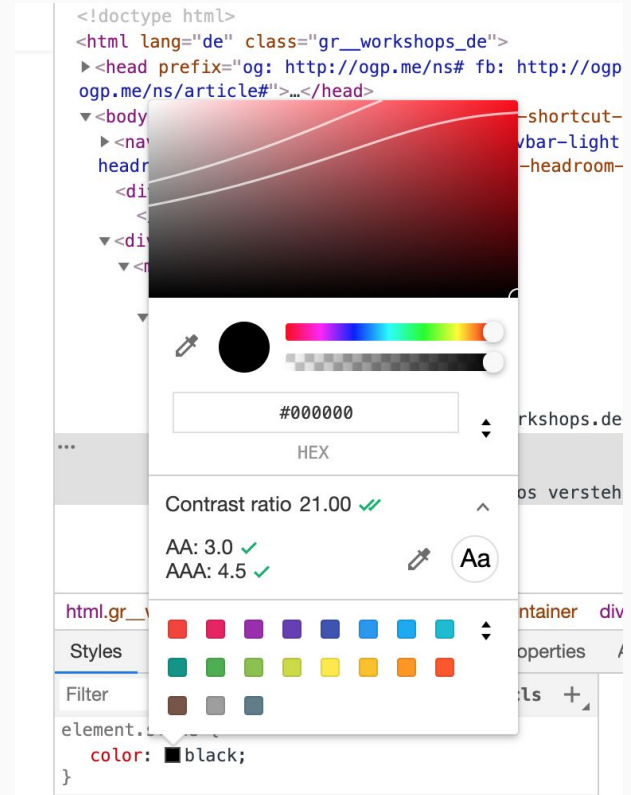
HOUSE

BATTLESTAR GALACTICA

Gossip Girl

Chrome Devtools: Contrast Ratio

- Build in chrome devtools
- Just select an element and click on the color



Task

Check your page for color-blindness



HTML

A good basis for accessibility

HTML Semantics

<code>

Best accessibility a screen reader user can have is a content structure

```
<h1>My heading</h1>
```

```
<p>This is the first section of my document.</p>
```

```
<p>I'll add another paragraph here too.</p>
```

```
<ol>
```

```
  <li>Here is</li>
```

```
  <li>a list for</li>
```

```
  <li>you to read</li>
```

```
</ol>
```

```
<h2>My subheading</h2>
```

HTML Semantics

<code>

Modern website structure for layouts (instead of table layout)

```
<header>
```

```
  <h1>Header</h1>
```

```
</header>
```

```
<nav><!-- main navigation in here --></nav>
```

```
<!-- Here is our page's main content -->
```

```
<main>
```

```
  <!-- It contains an article -->
```

```
  <article>
```

```
    <h2>Article heading</h2>
```

HTML Semantics

<code>

Beware of keyboard accessibility

```
<div (click)="clicked()">Click me!</div>
```

HTML Semantics

<code>

Beware of keyboard accessibility

```
<div (click)="clicked()">Click me!</div>
```

```
<button (click)="clicked()">Click me!</button>
```

```
<div (click)="clicked()" tabindex="0">Click me!</div>
```

HTML Semantics

<code>

Meaningful text labels

Whales are really awesome creatures. To find more out about whales, `click here`.

Whales are really awesome creatures.

`Find out more about whales`.

HTML Semantics

<code>

Define alternative text for images

```

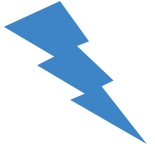
```

ARIA

Accessible Rich Internet Applications

ARIA defines a way to make Web content and
Web applications more accessible to people with
disabilities

Why / What you'll learn



- It's a official W3C specification
- More accessible to people with disabilities
- Defines roles, states and properties
- Adds more semantic to HTML elements

Role Attributes

roles defining a **type** of user interface element

Example Role

<code>

This link behaves more like a button and is marked with this role

```
<a href="#" role="button" aria-label="Delete item 1">Delete</a>
```

**Each role has
different states and
properties that can
be defined**

Example Aria-Attribute

<code>

With aria-label a screen reader knows what to read on focus

```
<a href="#" role="button" aria-label="Delete item 1">Delete</a>
```

Aria Alerts

<code>

Screen readers won't pick this up or alert users to it by default

```
<div class="errors" role="alert" aria-relevant="all">  
  <ul>  
  </ul>  
</div>
```

Aria Properties

<code>

Use labelledby if you display the alt-text anyway

```

```

```
<p id="dino-label">The Mozilla red Tyrannosaurus Rex: A two legged  
dinosaur standing upright like a human, with small arms, and a large head  
with lots of sharp teeth.</p>
```


Aria States

<code>

Tells screen-readers if they should ignore the element.

```
<p aria-hidden="true">This content is ignored by screen readers.</p>
```

```
<p>This content is not hidden.</p>
```

Abbreviations

<code>

Help people to understand your text better

```
<p>
```

Web content is marked up using

```
<abbr title="Hypertext Markup Language">HTML</abbr>.
```

```
</p>
```

Mouse-specific events

Some events can have accessibility issues e.g. with keyboard controls

- mouseover
- mouseout
- dblclick

Keyboard based Navigation

Keyboard Navigation

- Make sure there is a visible focus style for interactive elements
- Check to see that keyboard focus order matches the visual layout.
- Remove invisible focusable elements.

Keyboard Navigation

<code>

Using the tabindex attribute

```
<label>First in tab order:<input type="text"></label>
```

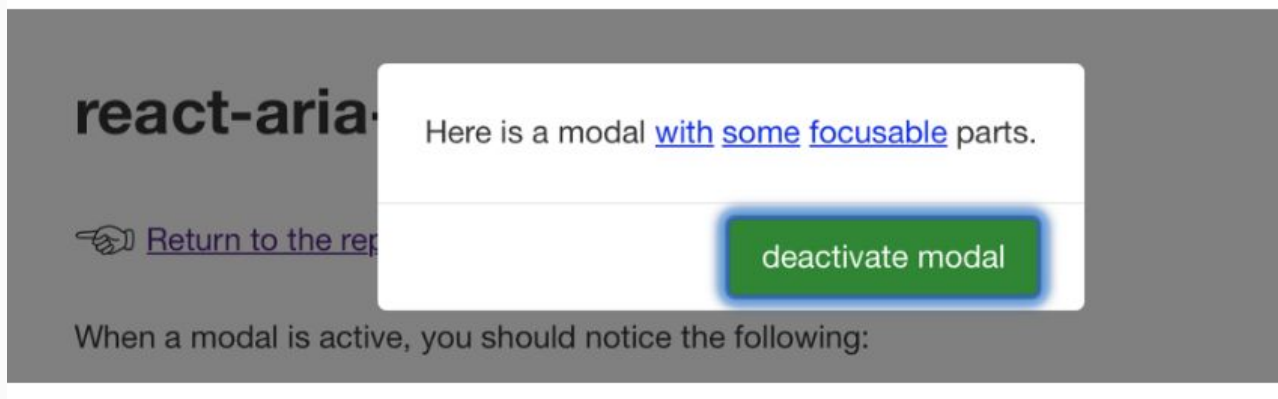
```
<div tabindex="0">Tabbable due to tabindex.</div>
```

```
<div>Not tabbable: no tabindex.</div>
```

```
<label>Third in tab order:<input type="text"></label>
```

Focus Trap

- Loops the navigation that is made using the tab key inside a element
- Help your navigation to be accessible by users with disabilities
- Modal or alert element



The A11y Project

A11y Checklist

- A community-driven effort to make web accessibility easier
- software, books, blogs, online tools
- A11y Checklist

All images

- ▼ Make sure that all `img` elements have an `alt` attribute.

1.1.1 Non-text Content.

`alt` attributes (alt text) give a description of an image for people who may not be able to view them. When an `alt` attribute isn't present on an image, a screen reader may announce the image's file name and path instead, failing to communicate the image's content.

- ▶ Make sure that decorative images have empty `alt` attribute values.

- ▶ Provide a text alternative for complex images such as charts, graphs, and maps.

- ▶ For images containing text, make sure the `alt` description includes the image's text.

Task

Fix your page for keyboard navigation



Screen Readers

Screen readers are software applications that attempt to convey what people with normal eyesight see on a display to their users via non-visual means like text-to-speech

Apple VoiceOver

CMD + F5



Microsoft Narrator

Windows logo key + Ctrl + Enter



Task

Use VoiceOver and fix image tags



Tools

Use tools to help you find

- Lacking keyboard support
- Missing labels
- Invalid ARIA attributes
- Color contrast
- ...and more!

Codelyzer

Linting rules to sure your code meets accessibility standards

Codelyzer is a tool great for teams and individuals, which helps you write consistent code, and discover potential errors.

Task

Run Codelyzer on your project




Lighthouse

Google Chrome

Lighthouse

- Google Chrome Inspector
- Also possible to run via CLI
- Only a few rules for a11y



Audits

Identify and fix common problems that affect your site's performance, accessibility, and user experience. [Learn more](#)

Device Mobile Desktop

Audits Performance Progressive Web App Best practices Accessibility SEO

Task

Run a lighthouse cli check



Axe

JavaScript library for accessibility testing

What is Axe?

- JavaScript library for accessibility testing
- Engine powering browser extensions, test integrations
- A handy unit testing tool
- Open source

axe-core

Install the axe-core package

- Install the package via npm
- Add it to your devDependencies list

```
npm install axe-core --save-dev
```

Axe a11y Check

<code>

Example check

```
// Test an element reference, selector, or include/exclude object.
var context = { exclude: ['#some-id'] };
var config = {
  rules: {
    "color-contrast": { enabled: false },
    "valid-lang": { enabled: false }
  }
};

axe.a11yCheck(context, config, function(results) {
  // do stuff with the results
});
```

Axe as Unit Test

<code>

It's easy to integrate this into your test suite

```
var axe = require('axe-core');

describe('Custom component', function() {
  it('should have no a11y violations', function(done) {
    axe.a11yCheck('.some-element-selector', {}, function (results) {
      expect(result.violations.length).toBe(0);
      done();
    });
  });
});
```

Task

Write an axe unit test



axe-webdriverjs

Use Axe with different browsers
as integration test

Axe Browser Support

- Microsoft Edge v40 and above
- Google Chrome v42 and above
- Mozilla Firefox v38 and above
- Apple Safari v7 and above
- Internet Explorer v9, 10, 11

Install the axe-core package

- Install the package via npm
- Add it to your devDependencies list

```
npm install axe-webdriverjs --save-dev
```

Integration test with axe

<code>

Using Selenium to start a firefox instance and run your a11y checks

```
var AxeBuilder = require('axe-webdriverjs'),
    WebDriver = require('selenium-webdriver');

var driver = new WebDriver.Builder().forBrowser('firefox').build();

driver
  .get('https://localhost:4000')
  .then(function (done) {
    AxeBuilder(driver)
      .analyze(function (results) {
        expect(results.violations.length).toBe(0);
        done();
      });
  });
```

Task

**Create and run an axe
integration test**



Angular Material CDK

A11y CDK Package

The Angular Material a1 1y package provides a number of tools to improve accessibility, described below.

Angular CDK a11y

- The Angular Material library aims to be fully accessible
- A11y package supports
 - LiveAnnouncer for Screen Readers
 - cdkTrapFocus for e.g. Modal Dialogs

Angular Material CDK

<code>

Example for Routing and focus management

```
router.events.pipe(filter(e => e instanceof NavigationEnd)).subscribe(
  () => {
    const mainHeader = document.querySelector('#main-content-header')
    if (mainHeader) {
      mainHeader.focus();
    }
  }
});
```