Course Outline | Contemporary JavaScript

1 day(s)

Overview

JavaScript has grown to become a major theme in modern software-development that is driving the evolution of the Web from a static-document delivery system to a dynamic application-deployment infrastructure. Moreover, ever-increasing client-side complexity, and a choice of client-server communication techniques raise critical design-considerations that modern JavaScript-developers must address. Yet few JavaScript and AJAX courses go beyond an essentially syntax-oriented syllabus, thus failing to provide the grounding in design- and implementation-theory that non-trivial JavaScript programming requires.

This three-day course addresses such deficiencies by taking contemporary JavaScript-training to the next level. Covering the language in full, it includes the extensions that ECMAScript-5/HTML5 introduced, and augments this with coverage of page-structure and -style manipulation, and exploration of the range of client-server communication techniques that are now available. Crucially, it provides delegates with a firm grounding in application-design principles that go far beyond low-level syntax and semantics, and which equip delegates with the techniques and approaches they need in order to implement robust, secure and extensible applications that embody a high degree of code-reuse.

Developed and delivered by an accomplished practitioner and trainer with over thirty years experience in programming, this course includes a rich and varied set of hands-on exercises that allow delegates to explore at first-hand the concepts and issues covered.

Target Student

This course is suitable for developers encountering JavaScript for the first time, as well as those with experience of the language who wish to improve their understanding of the language, and of the design principles that complex client-side development requires. It assumes no previous understanding and experience of programming in JavaScript specifically, but some understanding and experience of programming in general is essential. Some understanding and experience in HTML and CSS is also assumed.

Learning Outcomes

By attending this course you will acquire an in-depth understanding of JavaScript syntax and semantics, along with a broad range of techniques and approaches that you can use to develop well-designed, robust and scalable applications that are composed of re-usable components.

Course Outline

Language fundamentals

- Reserved words and operators
- Scalar objects (simple numbers and strings)
- Flow control and conditional logic, loops
- Language caveats, quirks and defects
- Best practice

The JavaScript object-model

- Non-scalar types (arrays and class-like objects)
- Membership operators
- Object and array property-iteration
- Object extension
- Use and misuse of prototypical inheritance
- Freezing, sealing and preventing object extension
- Configuring object-properties
- Get and set methods
- Object-literal notation and JSON

Functions

- Functions definitions, arguments and returns
- Functions as methods
- Function properties
- Functions as constructors
- Inner functions
- The function-call mechanism
- References to functions
- Closures
- Function-references and the Strategy pattern (callbacks)
- Recursion
- Overriding, augmenting and underpinning object methods
- Aspect Orientation

Exception Handling

- Proprietary approaches and their drawbacks
- Try/throw/catch/finally
- Exception types
- Generating stack traces

Pre-Defined Types and the Run-Time Environment

- The Global and Math objects, the Date and Regular Expression types
- The Document Object Model
- DOM node-types
- Page modification through DOM and Style-property manipulation
- Event handling
- Cookies

Client-Server Communication

- XMLHTTPRequest (XHR)
- WebSockets and Server-Sent Events
- Caching, latency, concurrency and race conditions
- Security
- Dynamic and on-demand script-loading

Design theory

- Symmetry and its exploitation
- Subsumption (Has A) and inheritance (Is A) abuse and misuse
- The Module, Null Object, Decorator and Chain of Responsibility patterns
- Type invariance and its role in object composition and decoration
- Resolving scaling-challenges using transaction-based designs