Effective JavaScript (FT212)

24 Hours

Outline

JavaScript has become essential for all developers wishing to deliver a rich user experience. Optimizing code is vital for a seamless user experience and can dramatically decrease the time it takes to execute your JavaScript code. Building reusable, scalable and performance oriented JavaScript code requires a deep understanding of the language.
In this course we will present the tools and patterns to build scalable JavaScript web sites. Each module of this course covers a different thematic area of effective JavaScript.

Target audience

This course is intended for JavaScript programmers who want to deepen their understanding of the language, and get familiar with the best practices and tools to build scalable and reliable JS applications.

Prerequisites

- At least 1 year of experience in JavaScript programming
- At least 3 years of experience in writing OOP code in a high-level programming language (e.g., Java, C++, C#)

Objectives

On completing this course, delegates will be able to:

- Build more predictable, reliable, and maintainable JavaScript applications and libraries
- Identify common pitfalls and bottlenecks in JS code
- Use design patterns in JavaScript
- Know how to run a JS profiler
- Employ the new features of ECMAScript 6

Contents

Day 1

Module 1: JavaScript Types - 1 Hour

- Understand JS floating-point numbers
- Beware of implicit coercions
- Prefer primitives to object wrappers
- Avoid using == with mixed types
- Thinks of strings as sequences of 16-bit code units

Module 2: Variable Scope - 1 Hour

- Minimize use of the global object
- Always declare local variables
- Avoid with
- Get comfortable with closures
- Understand variable hoisting
- Use self-invoking functions to create local scopes

Module 3: Working with Functions - 2 Hours

- Understand the difference between function, method and constructor calls
- Get comfortable using higher-order functions
- Use call() to call methods with a custom receiver
- Use arguments to create variadic functions
- Never modify the arguments object
- Use bind() to curry functions
- Prefer closures to strings for encapsulating code
- Avoid relying on the toString() method of functions

Module 4: Objects and prototypes Part I - 4 Hours

- Understand the difference between prototype, getPrototypeOf, and__proto__
- Never modify __proto
- Make your constructors new-agnostic

- Store methods on prototypes
- Use closures to store private data
- Store instance state only on instance objects
- Recognize the implicit binding of this

Day 2

Module 5: Objects and prototypes Part II - 2 Hours

- Call superclass constructors from subclass constructors
- Never reuse superclass property names
- Avoid Inheriting from standard classes
- Treat prototypes as an implementation detail
- Avoid reckless monkey-patching

Module 6: Arrays and Dictionaries - 2 Hours

- Build lightweight dictionaries from instances of object
- Prefer arrays to dictionaries for ordered collections
- Avoid modifying an object during enumeration
- Prefer for loops to for...in loops for array iteration
- Prefer iteration methods to loops
- Reuse generic array methods on array-like objects
- Prefer array literals to the array constructor

Module 7: Libraries and API Design - 2 Hours

- Maintain consistent conventions
- Treat undefined as “no value”
- Avoid unnecessary state
- Use structural typing for flexible interfaces
- Distinguish between Array and Array-like
- Avoid Excessive Coercion
- Support method chaining

Module 8: Working with the DOM - 2 Hours

- Minimize DOM access
- Minimize the number of reflows and repaints
- Change CSS classes, not styles
- Caching your objects

Day 3

Module 9: Design Patterns in Javascript – 2 Hours

- Singleton
- Factory
- Decorator
- Proxy
- Observer

Module 10: Concurrency - 2 Hours

- Don’t block the event queue on I/O
- Don’t block the event queue on computation
- Be aware of dropped errors
- Use recursion for asynchronous loops
- Never call asynchronous callbacks synchronously
- Use promises for cleaner asynchronous logic

Module 11: General Performance Tips - 1 Hour

- JavaScript profiling
- Shrink your JS files
- Access JavaScript files through CDNs

Module 12: ECMAScript 6 - 3 Hours

- Using let variables
- Using arrow functions
- Destructuring objects and arrays for easier data access
- Introducing JavaScript classes
- Sets and maps
- Iterators and generators
- Encapsulating code with modules
- Promises and asynchronous programming