Intermediate JavaScript

Prepared for Salesforce

Last Updated: May 17, 2023

Course Overview:

This workshop teaches the participants how to take advantage of some of the advanced features offered by JavaScript for building better maintainable large scale JS applications. This class dives deep into the design decisions made in the language and how it is different from other general purpose languages. At the end of this course, participants will have a deeper understanding of the language, confident enough to solve complex problems and have the foundation needed to build full stack applications in JS.

Course Duration: This course will be delivered in 3 days

Course Prerequisites:

- Basic git familiarity (clone, pull, etc.)
- Working with JS for 6+ months
- Comfortable with basic HTML and CSS (including form elements)
- Comfortable with debugging and writing code

Course Objectives

- At the end of this course, you will be able to
  - Understand and exploit the weakly typed & dynamic nature of JavaScript
  - Gain deeper understanding of the invocation context (“this”)
  - Apply functional programming techniques to write maintainable JS
  - Organize logic using OO programming paradigms
  - Solve asynchronous problems using various techniques such as callbacks, event-emitters, promises & async-await
  - Use some of the HTML5 APIs
  - Use the tools needed for building complex JS applications
  - Secure JS applications from the common threats

Technical Setup

- Chrome Browser
- Any editor (Visual Studio Code / WebStorm / Atom etc)
- Node.js (https://nodejs.org)

Course Outline

- JavaScript Language
  - Modern ES syntax
  - Execution context (this)
  - Scope and closures
- Functional Programming Part 1
  - Defining functions and parameters
  - Functions are objects
  - Higher order functions
  - Array data processing
  - Closures
  - Setting execution context
- The Document Object Model and Browser APIs
  - Referencing elements
  - Manipulating the DOM
  - Node attributes and content
  - Events
- Browser APIs
  - setTimeout and setInterval
  - LocalStorage
  - Web Workers
  - Websockets
  - Location, URLSearchParams, History
- Forms
  - Form elements
  - FormData & Forms vs. AJAX
  - Validations
  - File input
- Asynchronous Programming
  - JavaScript runtime
  - Promises
  - Async and await keywords
  - AJAX with axios
- Object Oriented Programming (Classical)
  - Methods
  - Static fields
  - Private fields
  - Getters and Setters
  - Instance fields
  - Inheritance
- Object Oriented Programming (Prototypal)

    - Prototype chain
    - Constructor functions
    - Behavior sharing patterns
- Functional Programming Part 2
    - Immutability
    - Currying and partial application
    - Point-free programming
    - Ramda / Lodash
    - Function composition
- Managing UI State
    - Dangers of UI as a function of time
    - Principles of declarative UI
    - Immutable CRUD operations in app state
- JavaScript Tooling
    - Node, npm, and yarn
    - Babel
    - Webpack
    - Linting
    - TypeScript
- Testing with Jest
    - Basics and matchers
    - Setup and environment
    - Stubs and spies
    - Timers
    - Async
    - Module mocks
- Debugging
    - Console.log Driven Development
    - DevTools
    - Debugger in IDE
- Intro to Web Security
    - XSS
    - Content Security Policy
    - Escaping user input