AMBIENT-IT

+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated 06/06/2025

Sign up

JavaScript training : Advanced programming

3 days (21 hours)

Presentation

Discover our "JavaScript, advanced programming" course, specifically designed to master the latest developments in the language. Perfectly suited to experienced developers, it gives you in-depth mastery of the language for rich, robust and secure web applications from ES6 to ES2025.

Learn how to write clear, high-performance code using the language's advanced functions We've got you covered: closures, generators, modern classes, generators, modules and fine-tuned management of execution contexts. Get hands-on experience with realistic, practical workshops throughout your training. In particular, you'll delve into asynchronous code management, mastering promises, async/await, Fetch API, and advanced event management techniques. You'll explore native Web Components, modular architecture and lightweight JavaScript micro-frameworks such as Alpine.js or Lit, to select the most appropriate technology for each context. Enhance the quality of your code with the most effective tools on the market (ESLint, Jest, Prettier), while ensuring optimum security against common vulnerabilities (XSS, CSP). You'll also master the best security practices for effectively protecting your site. Finally, get ready now for future language evolutions, such as Pattern Matching and native immutability, to stay at the cutting edge of JavaScript.

This advanced JavaScript training course will introduce you to the latest version of the language. JavaScript 15 - ES2024.

Objectives

- Master the advanced features of JavaScript ES6
- Build an application using JavaScript's advanced features
- Browse and modify the DOM with the JQuery library
- Managing HTML5 JavaScript APIs
- Using JavaScript for different use cases: web applications, browsers, software, servers, etc.
- Understanding server-side development with Nodejs
- Mastering debugging
- An in-depth understanding of Object-Oriented Programming

Target audience

- Front-end developers
- Technical architects
- UI designer
- Project managers
- Design engineer

Prerequisites

- Have completed our initial JavaScript training
- Knowledge of HTML and CSS
- Basic knowledge of XML

Program of our JavaScript : Advanced programming

Introduction to modern JavaScript with ES6+

- Browser and runtime compatibility
- How do you choose the right version at the right time?
- Tools to get you started (VS Code, DevTools, npm, yarn, Webpack, etc.)
- Modern variables (let, const) - block scope
- Arrow functions
- Destructuring (objects/tables)
- Rest/spread (...) operators on arrays and objects
- New safety operators (?., ??)
- Template literals (interpolated strings)
- Composition with Mixins (reusability, modularity)
- Immutability (principles and best practices)
- Hands-on workshop: Set up your modern JavaScript environment (VS Code, npm, DevTools), then create a simple script incorporating the new syntaxes (destructuring, advanced operators, arrow functions, etc.) to efficiently manipulate structured data.

ECMAScript developments (ES6 ? ES2024)

- ES6 (2015): Classes, Modules, let/const, Arrow functions
- ES7 (2016): Exponentiation, Array.includes()
- ES8 (2017): Async/Await, Object.values()
- ES9 (2018): Rest/Spread objects, Regex improved
- ES10 (2019): Array.flat(), Object.fromEntries()
- ES11 (2020) : Optional chaining (?.), Nullish coalescing (??)
- ES12 (2021): Promise.any(), Numeric separators (_)
- ES13 (2022): Private fields (#), Private methods
- ES14 (2023): findLast(), Regex improvements
- ES15 (2024): Ergonomics, type accuracy, performance optimization

- The future ES2025: pattern matching, immutable structures (Records/Tuples)
- Practical workshop: Implement your choice of 3 new ES6+ features.

Advanced JavaScript functions

- Closures (closed functions on a lexical context)
- Execution context (this) and call/apply/bind methods
- Arrow functions vs. classic functions (this link)
- Immediate anonymous functions (IIFE)
- Generators (function* with yield) and custom iterators
- Practical workshop: Implement a closure function to create a counter with a private value (internal status memory).

Object-oriented programming in JS

- Prototypes and prototypal inheritance
- Constructor functions and the new keyword
- ES6 classes (class syntax, inheritance with extends, private fields #)
- Instance vs. static properties and methods
- Practical workshop: Design a small hierarchy of objects (class and subclass) to model a concrete case involving inheritance.

JS architecture & patterns

- Modularity and SOLID in JS
- Patterns Observer, Factory, Singleton
- Dependency injection
- Maintainability and scalability
- Practical workshop: Implementing the Observer pattern to notify components.

Modules and code organization

- ES6 modules (import/export)
- Bundler and transpilation (Webpack, Babel)
- Dependency management with npm (package.json file, scripts)
- Practical workshop: Refactor an existing application into ES6 modules and build it using a bundler (e.g. Webpack) to optimize loading.

Advanced asynchronous JavaScript

- Asynchronous management and event loops
- Execution model: event loop, stack, task & micro-task queues
- Function callbacks and control inversion
- JavaScript promises (Promise objects, then/catch strings)
- Modern async/await syntax for promises
- Promises: creation, chaining, error handling, Promise.allSettled

- async / await: parallelism, cancellation with AbortController
- Asynchronous generators (for-await-of)
- Asynchronous Web APIs: fetch, Streams, Web Workers
- HTTP calls with the Fetch API (modern AJAX requests)
- Practical workshop: Consuming data from an external API using fetch and async/await to display dynamic results on the page.

Native Web Components

- Custom HTML elements (API Custom Elements)
- Shadow DOM (encapsulation of DOM and styles)
- HTML templates and slots for internal component content
- Web Components lifecycle (callback methods such as connectedCallback, etc.)
- Hands-on workshop: Create a Custom Element with Shadow DOM to encapsulate a reusable, styled widget in isolation.

Code quality and advanced testing

- Static analysis with ESLint (linting and coding rules)
- Automatic code formatting with Prettier
- Automated unit testing with Jest
- Practical workshop: Set up ESLint and Prettier on an existing project, then write and run a simple unit test with Jest to validate a function.

An overview of lightweight JavaScript micro-frameworks

- Alpine.js (minimalist framework that integrates via simple HTML attributes)
- Lit (lightweight library for building responsive Web Components)
- Svelte (compiled framework, no heavy runtime on the browser side)
- Practical workshop: Experiment with one of these micro-frameworks by implementing a simple feature (e.g. an interactive counter) to compare their approach with that of native JS (vanilla).

Safety and best practices

- XSS vulnerabilities and client-side injections (browser code security)
- Content Security Policy
- ECMAScript version compatibility (browser targets and Babel transpilation)
- Appropriate technological choices (native JavaScript vs. frameworks, depending on project needs)
- Practical workshop: Analyze an existing code base to correct potential XSS vulnerabilities and apply good security practices while adapting the code to compatibility constraints.

Companies concerned

This course is aimed at both individuals and companies, large or small,

wishing to train its teams in a new advanced IT technology, or to acquire specific business knowledge or modern methods.

Positioning on entry to training

Positioning at the start of training complies with Qualiopi quality criteria. As soon as registration is finalized, the learner receives a self-assessment questionnaire which enables us to assess his or her estimated level of proficiency in different types of technology, as well as his or her expectations and personal objectives for the training to come, within the limits imposed by the selected format. This questionnaire also enables us to anticipate any connection or security difficulties within the company (intra-company or virtual classroom) which could be problematic for the follow-up and smooth running of the training session.

Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

Organization

The course alternates theoretical input from the trainer, supported by examples, with brainstorming sessions and group work.

Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

Sanction

A certificate will be issued to each trainee who completes the course.