

JavaScript and Node.js Application Development

Language foundations, server-side tools, APIs and a containerised example

FA-0734 · 6 days; optional 1-day Angular introduction

Software Development · DevOps, Cloud & Infrastructure

Introduction

Build JavaScript foundations and apply them to a small Node.js command-line and web/API project. Explore asynchronous behaviour, event-driven interfaces, CouchDB persistence and a controlled container/deployment example.

The main route is six days, followed by an optional one-day Angular starter exercise. Learners with established JavaScript skills may use the four-day Node core; neither route promises a complete production microservice platform or full Angular expertise.

Learning outcomes

- Explain functions, objects, classes, prototypes, scope and asynchronous JavaScript.
- Build a small Node.js CLI or API and apply selected design patterns.
- Use events, streams, child processes or sockets in bounded examples.
- Connect to CouchDB and handle document update conflicts.
- Apply a suitable authentication/permission approach and inspect error cases.
- Package a supplied application with Docker and review deployment requirements.
- If attending the optional day, connect a small Angular starter interface to the sample API.

Prerequisites

- Basic programming familiarity and command-line/file-management skills; no prior Node.js expertise required.
- JavaScript experience for learners skipping the first two days.
- A supported Node.js release, suitable editor, compatible CouchDB/Docker lab and approved installation/network access.
- Basic HTML/CSS is helpful for the optional supplied Angular project.

Training outline

1. Day 1 — JavaScript functions and object design

- Functions as objects, object construction and prototype-based inheritance.
- Current class syntax and its relationship to prototypes; JavaScript does support classes.
- Compare var, let and const scope and distinguish module scope from the global object.
- Use small examples to inspect object responsibilities and suitable factory patterns.

2. Day 2 — Context and asynchronous behaviour

- The "this" keyword in Java Script
- Running in strict mode
- Callbacks
- Error handling with callbacks
- Promises
- Chaining promises
- async and await

Inspect this binding, strict-mode effects and rejection/error propagation; compare callbacks, promises and async/await without assuming asynchronous syntax makes CPU work non-blocking.

3. Day 3 — Node.js and command-line structure

- Node runtime, modules, packages and the selected project structure.
- Build a small CLI with clear input/output and error behaviour.
- Use a model/factory example to organise responsibilities; assess complexity rather than apply patterns mechanically.

4. Day 4 — Events, processes, streams and sockets

- Use a selected EventEmitter example and handle errors.
- Inspect child-process lifecycle and platform constraints; distinguish separate processes from worker-pool activity.
- Trace a small stream with backpressure and a socket example where relevant.
- Identify work that can block the event loop and review resource cleanup and failure behaviour.

5. Day 5 — CouchDB and controlled packaging

- Read/write sample JSON documents through CouchDB's HTTP interface and handle revision/update conflicts.
- Prepare a small Docker image with compatible dependencies and configuration kept separate from code.
- Run the sample container in the lab and review logs, networking, persistence and secret handling.
- Compare this exercise with actual operational release, monitoring and recovery requirements; publishing a container is not proof of production readiness.

6. Day 6 — REST API and access controls

- Implement selected REST endpoints, validate input and return appropriate responses.
- Connect the API to the sample document store and inspect success and failure paths.
- Use a suitable authentication and permission approach; protect credentials and review the required transport/session/token assumptions.
- Review application boundaries and microservice trade-offs using the modest project rather than a complete distributed architecture.

7. Optional day 7 — Angular starter interface

- Orient to a supplied compatible Angular project and its component/data flow.
- Call the sample API, display results and handle a simple loading or error state.
- Review the interface/backend contract and identify next steps for deeper frontend development.