SOFTWARE DEVELOPMENT AND ENGINEERING PRACTICES

Server-Side Development with Node.js and Express

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

Node.js lets JavaScript developers build the server side with the language they already know, and Express remains the framework most Node servers are built on or descended from. The catch is that Node is unforgiving of fuzzy understanding: it is single-threaded and asynchronous to its core, so a developer who does not really understand the event loop writes servers that block, leak, or fall over under load, and Express's minimalism means every structural decision is yours to get right.

This is a hands-on, practitioner course. It starts with how Node actually works, the event loop and asynchronous model, because every Express feature and every production problem traces back to it. From there it layers up: routing and request handling, middleware as the framework's central idea, persistence, security, and the error handling and configuration that make a service production-worthy. Rather than tour the entire npm ecosystem, it goes deep on one well-built API, developed across the whole course. Every module ends with a lab and builds on the one before.

Who Should Attend

- JavaScript developers moving from the front end to the server
- Back-end developers from other stacks (C#, Java, Python) adding Node.js
- Developers maintaining an existing Node or Express service they did not build

Developers whose JavaScript predates ES6 should take *Modern JavaScript (ES6+)* first.

Prerequisites

- Solid modern JavaScript, including promises and async/await
- Basic understanding of HTTP: requests, responses, and status codes
- Comfortable with a terminal and npm

What You Will Learn

- Explain Node's event loop and asynchronous model and what they mean for server code
- Build an Express application with clean routing and well-organized request handlers
- Design and use middleware for logging, validation, authentication, and error handling
- Connect a Node service to a database and structure the data access layer
- Secure an Express API: authentication with JWT, input validation, and the standard hardening steps
- Prepare a service for production: configuration, error handling, logging, and deployment

Course Outline

Day one: how Node works, and building on Express

- Node.js Under the Hood
  - The event loop: one thread, no waiting, and why that scales
  - Asynchronous patterns in server code: promises and async/await in practice

  - Node project mechanics: npm, package.json, and project layout
  - Lab: build a tiny HTTP server with no framework, then block its event loop and watch what happens
- Express Fundamentals
  - What Express adds, and the minimalism that is both gift and trap
  - Routing: methods, paths, parameters, and routers for structure
  - Handling requests and responses: JSON, status codes, and consistent shapes
  - Lab: scaffold the course API in Express with clean, router-based structure
- Middleware
  - The middleware pipeline: how every request actually flows
  - Built-in and third-party middleware: body parsing, CORS, logging
  - Writing your own middleware, and ordering it correctly
  - Lab: add logging, validation, and a custom middleware to your API and trace a request through the stack

Day two: data, security, and production readiness

- Working with Data
  - Choosing a database and driver: the lay of the land from PostgreSQL to MongoDB
  - Structuring data access behind a service layer instead of inside route handlers
  - Async data flows done right: no unhandled rejections, no forgotten awaits
  - Lab: connect the API to a database and implement full CRUD through a service layer
- Security and Authentication
  - Authentication with JWT: issuing, verifying, and protecting routes
  - Input validation and the injection mistakes that top every vulnerability list
  - Standard hardening: helmet, rate limiting, and sane CORS
  - Lab: secure the API with JWT authentication and verify both access and rejection paths
- Errors, Configuration, and Shipping
  - Centralized error handling that never leaks internals to clients
  - Configuration and secrets: environment variables done properly
  - Logging for production, and deployment options from PaaS to containers
  - Lab: add centralized error handling and configuration, then deploy the finished API

Extended Version

The three-day version keeps the same gradient and adds depth for real services:

- TypeScript in Node projects, and why most teams now start there (see *TypeScript for JavaScript Developers*)
- Testing Node services: unit tests and API tests with Supertest
- Performance and resilience: clustering, caching, and graceful shutdown
- A capstone: extend the course API with a complete new secured resource, tested and deployed