Intermediate Development with Node.js

Description

This course builds on the fundamentals of Node.js to help developers deepen their skills in server-side JavaScript. It covers advanced asynchronous programming techniques, performance optimization, security, testing, and scalable application design. Participants will explore real-world use cases such as building RESTful APIs, managing stateful applications, integrating databases, and deploying applications. Through hands-on labs, students will gain the experience necessary to create production-ready Node.js applications.

Objectives

By the end of this course, students will be able to:

- Build advanced RESTful APIs and microservices with Node.js and Express.
- Use advanced asynchronous patterns (async iterators, workers, queues, streams).
- Apply security best practices (authentication, authorization, environment management).
- Optimize performance and manage scalability in Node.js applications.
- Use clustering, worker threads, and load balancing for concurrency.
- Write and maintain automated tests for Node.js codebases.
- Integrate with SQL and NoSQL databases, including advanced queries and transactions.
- Use WebSockets for real-time communication.
- Deploy Node.js applications to cloud platforms (AWS, Azure, or similar).
- Apply logging, monitoring, and debugging strategies for production environments.

Topics

- Advanced Asynchronous Programming
- Advanced Express and REST API Design
- Authentication and Security
- Working with Databases (SQL and NoSQL)
- State Management and Caching
- Streams and Real-Time Communication (WebSockets)
- Performance and Scalability
- Testing and Debugging Node.js Applications
- Deployment and DevOps with Node.js
- Monitoring and Logging in Production

Audience

This course is designed for developers with prior experience in Node.js who want to build more complex, scalable, and secure server-side applications, and who need to prepare their applications for production use.

Prerequisites

Students should have completed an introductory Node.js course or have equivalent experience, including building basic servers, working with the file system, and using Express for routing.

Duration

Five days

Pricing

This course is available for lease on a per seat / per day rate. The courseware includes slides, notes, demos, assignments, and complete source code for all assignments.

Course Outline

I. Advanced Asynchronous Programming

A. Event loop review and async performance considerations

B. Async iterators and generators

C. Worker threads and background tasks

D. Message queues and job scheduling

II. Advanced Express and REST API Design

A. Designing scalable RESTful APIs

B. Express middleware patterns and composition

C. Error handling and request validation

D. API versioning and rate limiting

III. Authentication and Security

A. User authentication (sessions, JWT, OAuth2)

B. Role-based access control and permissions

C. Securing environment variables and secrets

D. Protecting against common attacks (XSS, CSRF, SQL injection)

IV. Working with Databases

A. Advanced queries with MongoDB and Mongoose

B. SQL databases with Node.js (PostgreSQL/MySQL)

C. Transactions and data integrity

D. Connection pooling and performance tuning

V. State Management and Caching

A. Using Redis or Memcached for caching

B. Session storage in distributed environments

C. Strategies for scaling stateful apps

VI. Streams and Real-Time Communication

A. Advanced stream handling in Node.js

B. Implementing WebSockets with Socket.io

C. Building real-time apps (chat, notifications, live dashboards)

VII. Performance and Scalability

A. Profiling Node.js applications

B. Load balancing with clustering and PM2

C. Worker threads vs child processes

D. Horizontal scaling and microservices patterns

VIII. Testing and Debugging Node.js Applications

A. Unit testing with Jest or Mocha

B. Integration and end-to-end testing

C. Test-driven development (TDD) in Node.js

D. Debugging tools and techniques

IX. Deployment and DevOps with Node.js

A. Preparing apps for production

B. Deploying Node.js to AWS, Azure, or GCP

C. Using containers (Docker) for Node.js apps

D. Continuous Integration/Continuous Deployment (CI/CD) pipelines

X. Monitoring and Logging in Production

A. Logging strategies (Winston, pino)

B. Error tracking (Sentry, similar tools)

C. Health checks and metrics

D. Observability in distributed systems