

Node.js Development

Course Duration ~75 Hours · 10 Weeks · Live Online · Build-Along · Deployed Services

Curriculum reviewed: July 2026 — the Node runtime moves fast; this syllabus tracks the current baseline

[Runtime Internals](#)
[Native TypeScript](#)
[Streams](#)
[APIs & Data](#)
[Queues & Real-Time](#)
[Production Ops](#)

In short: Build backends that stay up. Most Node courses still have you install four tools that ship with the runtime now — CommonJS, ts-node, Jest, and Express as the only option are the old stack. This course teaches the modern baseline: native TypeScript execution (with the crucial caveat that Node strips types but does not check them — tsc stays in CI — and the erasable-syntax constraints that push real design choices), the built-in test runner as the default, ESM with the CommonJS interop that still bites, and the permission model tied to supply-chain security. It is standalone backend engineering — services and APIs many clients consume, queue workers, streaming and file processing, real-time servers, microservices — where our Next.js course covers backend logic living inside the framework that serves a UI. Across 10 modules you diagnose event-loop blockages with flame graphs, handle backpressure correctly, ship typed APIs with a defended framework choice (Express vs Fastify vs Hono), build queues with retries and dead letters, harden auth against the OWASP API risks, and deploy an observable capstone with OpenTelemetry. Bun and Deno are treated honestly, and the release-line question is answered directly. The AI question too: AI writes endpoints quickly and gets error handling, backpressure, connection lifecycles and concurrency failure modes wrong — the value is architecture, review, debugging and operational judgment.

Duration	10 weeks · ~75 hours · live + hands-on	Format	Live online · build-along · deployed services
Prerequisite	Solid JavaScript. TypeScript taught from first principles. No backend experience required.	Curriculum	10 modules, from the event loop to production operations and capstone
Portfolio	7 deployed, production-style services plus a defended capstone	Ideal for	Fresh IT graduates, frontend developers moving backend, and Node developers modernising off CommonJS/ts-node/Jest

Why Learn Node When AI Writes Endpoints?

AI writes Node endpoints quickly and gets the things that matter wrong — error handling, backpressure, connection lifecycles, security boundaries, failure modes under concurrency. Those only surface in production. The value is architecture, review, debugging and operational judgment. You'll use AI tooling as an accelerator you review, and learn the failure patterns generated backend code repeats — so you ship its speed without shipping its mistakes.

Course Highlights

- Every module leads with what you build and ship, not what you "learn about"
- The runtime, properly understood: event loop, streams, backpressure, worker threads
- The modern baseline: native TypeScript execution, built-in test runner, ESM, permission model
- Types stripped is not types checked — tsc in CI, and the erasable-syntax design choices
- Framework choice taught, not defaulted: Express, Fastify and Hono compared honestly
- Distributed-work reality: queues, retries, idempotency, dead letters, graceful shutdown
- Supply-chain security and the permission model as first-class material
- Observability from day one: structured logs, metrics, tracing with OpenTelemetry

Hands-on Projects & Portfolio

- **Typed REST API** — validation, layered architecture, migrations and full test coverage
- **Streaming File Processor** — handles inputs larger than available memory without falling over
- **Background Job System** — queue, retries, idempotency and dead letter handling

- **Real-Time Service** — WebSockets, presence and reconnection handling
- **Authentication Service** — OAuth, refresh tokens, roles and rate limiting
- **Performance Investigation** — find and fix an event-loop blockage and a memory leak, with profiling evidence
- **Observable Deployment** — structured logs, metrics, tracing and load-test results
- **Capstone** — production backend, containerised, monitored and defended on architecture

Tools & Stack

Node.js · TypeScript · Express · Fastify · Hono · PostgreSQL · Prisma or Drizzle · Redis · BullMQ · Zod · node:test · Vitest · Supertest · Testcontainers · Docker · Kubernetes · OpenTelemetry · Pino · nginx · AWS · GitHub Actions · Git

Career Outcomes

Node remains one of the most widely deployed backend runtimes, and its biggest practical advantage is that a JavaScript or TypeScript developer can own both sides of a product without switching languages. The differentiator at hire time is not writing endpoints — it is understanding the event loop, streams, failure modes and observability well enough to keep a service alive under load. This course targets:

- Backend Developer · Node.js Developer · Full-Stack Developer · API Engineer
- Platform Engineer · Microservices Developer · DevOps-adjacent backend engineer · Technical founder shipping independent services

Who This Course Is For

- Final-year students and fresh IT graduates targeting backend or full-stack roles
- Frontend and React developers moving into backend work
- **Node developers trained on CommonJS, ts-node and Jest** who want the modern baseline
- Python, PHP, Java and .NET developers adding Node to their stack
- Full-stack developers who can build features but not diagnose production incidents
- Freelancers and founders shipping services alone

Module 1: The Runtime, Properly Understood

9 Hours

YOU SHIP

- A diagnosed event-loop blockage: reproduce it, read the flame graph, fix it with worker threads — the module that separates people who can debug Node from people who can only write it

TOPICS

- The event loop, phases and the microtask queue; what blocks it and how to tell; non-blocking I/O
- Single-threaded concurrency versus parallelism; worker threads and child processes; clustering
- Memory and garbage collection basics; reading a flame graph

Assessment: Quiz + Lab

Module 2: Modern Project Setup

7 Hours

YOU SHIP

- A modern project skeleton: ESM, native TypeScript execution for scripts, tsc as the CI type-check gate, a workspace layout, and a deliberate decision on where a build step still earns its place

TOPICS

- ESM and module resolution; package configuration; native TypeScript execution and its erasable-syntax constraints
- tsc as a CI type-check gate; when a build step still earns its place
- Dependency management; lockfiles; workspaces and monorepos; supply-chain hygiene

Assessment: Quiz + Lab

Module 3: Core APIs & Streams

8 Hours

YOU SHIP

- A streaming file-processing service that transforms, compresses and hashes inputs larger than available memory — with backpressure handled correctly, not by accident

TOPICS

- The standard library that removes dependencies: filesystem and paths, buffers, built-in fetch and HTTP clients, events and the emitter pattern
- Streams and backpressure; pipelines; compression; crypto
- Processing files larger than memory without falling over

Assessment: Quiz + Lab**Module 4: Building HTTP APIs**

8 Hours

YOU SHIP

- A typed REST API with validation, first-class error handling, pagination, versioning, uploads and OpenAPI docs — plus a written framework choice: Express vs Fastify vs Hono for this workload

TOPICS

- Routing, middleware and request lifecycle; REST design that survives contact with real clients
- Validation and typed contracts; error handling as a first-class concern; pagination; versioning; file uploads; OpenAPI documentation
- Express as the incumbent alongside Fastify and Hono — how to choose rather than default

Assessment: Quiz + Lab**Module 5: Data Layer**

7 Hours

YOU SHIP

- A production data layer: typed ORM schema and migrations, transactions, tuned connection pooling, an N+1 hunt with fixes, and Redis caching — behind a repository pattern that stays testable

TOPICS

- Relational databases with a typed ORM or query builder; schema design; migrations; transactions
- Connection pooling and lifecycle; N+1 problems; Redis for caching and sessions; document stores where they fit
- Repository patterns that keep data access testable

Assessment: Quiz + Lab**Module 6: Authentication, Authorisation & Security**

7 Hours

YOU SHIP

- An authentication service with OAuth, refresh tokens, role and permission models and rate limiting — hardened against the OWASP API risks and run under the runtime permission model

TOPICS

- Sessions versus tokens; JWT handling and its pitfalls; OAuth and third-party identity; password storage
- Role and permission models; rate limiting; input sanitisation; the OWASP API risks; secrets management
- The runtime permission model; dependency auditing and supply-chain security

Assessment: Quiz + Lab**Module 7: Asynchronous & Distributed Work**

8 Hours

YOU SHIP

- A background job system with a queue, retries, idempotency and dead letter handling — plus a real-time WebSocket service with presence, reconnection and graceful shutdown

TOPICS

- Job queues and background processing; retries, idempotency and dead letter handling; scheduled work; long-running tasks
- Webhooks in both directions; real-time with WebSockets and server-sent events
- Event-driven service communication; graceful shutdown

Assessment: Quiz + Lab

Module 8: Testing & Quality

7 Hours

YOU SHIP

- A fully tested service — unit and integration tests on the built-in runner, database tests in containers, contract and end-to-end API tests — with coverage and lint gates in CI

TOPICS

- The built-in test runner as the default — ecosystem alternatives covered honestly; unit versus integration testing
- Test doubles and dependency injection for testability; database testing with containers; contract testing; end-to-end API tests
- Coverage that means something; linting and CI gates

Assessment: Quiz + Lab

Module 9: Architecture at Scale

7 Hours

YOU SHIP

- A refactor of a codebase that outgrew its structure — layered, injected, configurable — plus a written monolith-vs-microservices decision with service boundaries drawn and defended

TOPICS

- Layering and separation of concerns; dependency injection; configuration and environment management
- The monolith-versus-microservices decision made honestly; service boundaries; inter-service communication; shared libraries
- Refactoring a codebase that outgrew its structure

Assessment: Quiz + Lab

Module 10: Production & Capstone

7 Hours

YOU SHIP

- Your capstone: one production-grade backend — containerised, health-checked, observable with logs, metrics and traces, load-tested — deployed and defended on architecture

TOPICS

- Containerising Node properly; process management; health checks and graceful restarts
- Structured logging; metrics and distributed tracing with OpenTelemetry; error tracking; performance profiling and memory-leak hunting; load testing
- Deployment targets including serverless and cold-start trade-offs; CI/CD; on-call reality; capstone design, build, deploy and defense

Assessment: Capstone Project + Review