BUILD · TAUGHT WITH NODE.JS

Backend Web

APIs, databases, authentication and the servers that answer them. You build services that hold real data, survive real traffic, and fail in ways you can explain.

Foundational | 4 phases | 20 sessions | 30 contact hours

Syllabus

4 phases, 20 sessions. Each phase ends in something you have built.

1 The Runtime Before the Framework

Write, run, and debug JavaScript on your own machine instead of in a browser.

5 sessions
$72 alone

1.1	**JavaScript, Server Side**	Values, functions, arrays, objects, and running a file with node. FREE PREVIEW
1.2	**Modules, npm, and node_modules**	ESM imports, package.json, semver ranges, and what npm install actually put on your disk.
1.3	**Async Without Fear**	Promises, await, and the event loop - traced by hand, not hand-waved.
1.4	**When It Breaks**	Stack traces, try/catch around await, unhandled rejections, and stepping through code in a debugger.
1.5	**The Toolbox Is Already Installed**	node --watch, node --run, --env-file, and node:test, with no nodemon or dotenv in sight.

By the end: **A command-line tool that reads files, calls a public API, and writes a report.**

2 HTTP and Your First API

Turn incoming requests into a JSON API a stranger could use without asking you.

5 sessions
$72 alone

2.1	**HTTP, Unabstracted**	Methods, status codes, headers, and bodies, served from node:http and poked with curl.
2.2	**Express 5**	Routes, params, middleware, JSON bodies, and an error handler that finally catches async throws.
2.3	**Designing Endpoints**	Resources, verbs, pagination, and choosing 201, 404, and 422 on purpose.
2.4	**Never Trust the Body**	Zod 4 schemas at the edge, 400s that explain themselves, and types inferred from the schema.
2.5	**Types Without a Build Step**	Annotating handlers in .ts files Node runs directly, and the enums and decorators it refuses.

By the end: **An in-memory CRUD API with validated input, honest status codes, and one error handler.**

3 Data That Survives a Restart

Put real, related data behind the API and keep it correct while the schema changes.

5 sessions
$72 alone

3.1	**SQL Before Any ORM**	SELECT, INSERT, UPDATE, and DELETE written by hand against node:sqlite, with nothing to install.
3.2	**Relationships and Constraints**	Keys, joins, NOT NULL, and letting the database refuse bad rows for you.
3.3	**Postgres From Node**	The pg driver, a connection pool, parameterized queries, and a data layer outside your routes.
3.4	**Migrations and Transactions**	Drizzle schema files, generated migrations, and changing a table without losing rows.
3.5	**Tests You Can Trust**	node:test, a fresh database per run, and fixtures that do not lie to you.

By the end: **A Postgres-backed API with migrations, seed data, and tests that run on a throwaway database.**

4 Users, Trust, and Production

Let real people log in without handing them each other's data, then run it somewhere real.

5 sessions
$72 alone

4.1	**Authentication**	Argon2id at the parameters OWASP names, server-side sessions, and HttpOnly cookies instead of a hand-rolled JWT.
4.2	**Authorization**	Ownership checks inside the query, and the missing WHERE clause that leaks every other user's rows.
4.3	**Attacks You'll Actually Meet**	SQL injection, XSS, CSRF, secrets committed to Git, and what CORS is not.
4.4	**Your Dependencies Run Your Code**	Lockfiles, npm audit, provenance, and why a package you never chose still executes on your laptop.
4.5	**Ship It and Watch It**	Docker, env config, migrations on deploy, structured logs, /health, and finding the slow request.

By the end: **A deployed, containerized API with sessions, ownership checks, structured logs, and a health check.**

Tools you will use

Node.js 26 | TypeScript | Express 5 | PostgreSQL 18 | Drizzle ORM | Zod 4 | node:test | Docker

What you will build

1. **The Bare-Metal Server**
   HTTP by hand, before Express does it for you
2. **Library Catalog API**
   Two related tables, migrations, and tests on a throwaway database
3. **Login You Can Defend**
   Argon2id, server-side sessions, and ownership checks that hold

Registration

Phase 1: The Runtime Before the Framework	$72
Phase 2: HTTP and Your First API	$72
Phase 3: Data That Survives a Restart	$72
Phase 4: Users, Trust, and Production	$72
Whole track, all 4 phases	**$240**

Buying every phase separately costs $288, so the whole track saves $48.

No payment is taken online. Registering creates an order and payment instructions are sent with it. Access opens once payment is confirmed.