

Next.js Development

Course Duration ~60 Hours · 8 Weeks · Live Online · Build-Along · Deployed Production Projects

Curriculum reviewed: July 2026 — Next.js moves fast; this syllabus tracks the current model

[App Router](#)
[Explicit Caching](#)
[Partial Prerendering](#)
[Full-Stack](#)
[Auth & Proxy](#)
[Migration](#)

In short: Ship full-stack React apps that are fast because of how they're architected, not because someone bolted on caching at the end. This course teaches the framework and the application around it — our React.js Development course teaches React itself, and is a prerequisite here, not a competitor; take them in order. The syllabus tracks the current model, not the tutorials the framework has since reversed: the **inverted caching model** — everything dynamic unless you explicitly opt in, with cache lifetimes and tag-based invalidation — gets a full module because it is the biggest mental-model change and the most common source of bugs in migrated apps. **Partial Prerendering** ends the static-vs-dynamic binary with a static shell plus streamed dynamic segments. **Turbopack** is the default bundler for new projects. And **migration is a visible track**: the middleware-to-proxy rename, async route params, removed legacy features, and an exercise upgrading an implicit-caching-era app. Across 9 modules you ship a PPR site, a full-stack SaaS dashboard, a Server Actions mutation build, a caching workshop, an e-commerce flow, a performance rescue and a deployed capstone. The AI question is answered honestly: AI scaffolds framework code fast and gets caching, rendering boundaries and data flow wrong in ways that only show up under load — the value is architecture, review and debugging.

Duration	8 weeks · ~60 hours · live + hands-on	Format	Live online · build-along · deployed production projects
Prerequisite	Working React and TypeScript. If you can't build components, use hooks and type props yet, take React.js Development first.	Curriculum	9 modules, from rendering models to deployment, migration and capstone
Portfolio	7 deployed, production-style projects plus a defended capstone	Ideal for	React developers architecting whole applications, full-stack aspirants, and teams migrating implicit-caching-era apps

Why Learn Next.js When AI Scaffolds Apps?

AI generates framework code fast and gets caching, rendering boundaries and data flow wrong in ways that only show up under load. The value is architecture, review and debugging — knowing why the page is dynamic when it should be cached, why the client bundle grew, why the mutation double-fires. You'll use AI tooling as an accelerator you review, including the framework's own MCP-based devtools integration for debugging workflows, and learn the failure patterns generated code repeats.

Course Highlights

- Every module leads with what you build and ship, not what you "learn about"
- The explicit caching model taught properly — dynamic by default, cache as a decision
- Partial Prerendering: static shell instantly, dynamic segments streamed
- Full-stack in one framework: typed database layer, Server Actions, auth, the proxy boundary
- A migration track for the implicit-caching era — the reversal that breaks upgrades
- Performance measured, not guessed: bundle analysis, Core Web Vitals, a rescue exercise
- Deployment trade-offs taught honestly: managed platform, self-hosted, containers, edge
- AI tooling as an accelerator you review — including MCP-based devtools debugging

Hands-on Projects & Portfolio

- **Partial Prerendering Site** — instant static shell, streamed personalised sections
- **Full-Stack SaaS Dashboard** — authentication, role-based access and a typed database layer
- **Mutation-Heavy Interface** — Server Actions with optimistic updates and real error states
- **Caching Workshop** — take a page with wrong caching behaviour and make it correct and explicit, with evidence

- **E-Commerce Flow** — static catalogue, dynamic cart and server-side checkout
- **Performance Rescue** — before/after Core Web Vitals measurements on a slow app
- **Migration Exercise** — upgrade an app written against the implicit-caching model and fix what breaks
- **Capstone** — deployed, monitored, production-grade application defended on architecture

Tools & Stack

Next.js · React 19 · TypeScript · Turbopack · Tailwind CSS · Prisma or Drizzle · PostgreSQL · Auth.js · Zod · React Hook Form · TanStack Query · Playwright · Vitest · Storybook · Vercel · Docker · Node.js · Redis · Git · GitHub Actions

Career Outcomes

Next.js is where production React increasingly lives, and job descriptions frequently name it directly rather than saying React. The framework's full-stack model lets one developer own an entire product surface — from database to deployed UI — which is why it is disproportionately common in startups, agencies and independent work. This course targets:

- Full-Stack Developer · Frontend Engineer · React/Next.js Developer · Product Engineer
- Web Platform Engineer · Technical Founder or freelance developer shipping complete products · Solutions Engineer

Who This Course Is For

- React developers who can build components but have never architected a whole application
- Final-year students and fresh IT graduates targeting full-stack roles
- Frontend developers moving into backend and full-stack work
- Backend developers who need to own the web layer as well
- **Teams maintaining Next.js apps written against the old implicit-caching model** — the migration track was built for you
- Freelancers and founders who need to ship complete products alone

Module 1: Framework Architecture & Rendering Models

6 Hours

YOU SHIP

- A working app whose routes deliberately span the rendering spectrum — static, dynamic, streamed, client — with a written per-route justification

TOPICS

- What a meta-framework actually gives you; the App Router model
- The full rendering spectrum and how to choose per route; the server/client boundary as an architectural decision rather than a syntax rule
- The build pipeline with the current default bundler — and the compatibility story for custom Webpack configuration

Assessment: Quiz + Lab

Module 2: Routing & Layouts in Depth

7 Hours

YOU SHIP

- A route architecture that scales to hundreds of pages — nested layouts, route groups, parallel and intercepting routes powering modals, per-segment loading and error boundaries

TOPICS

- Nested layouts and templates; route groups; dynamic and catch-all segments
- Parallel and intercepting routes for modals and complex UI; loading and error boundaries per segment; not-found handling
- Metadata and SEO APIs; structuring routes for an app that grows

Assessment: Quiz + Lab

Module 3: The Caching & Rendering Model

9 Hours

YOU SHIP

- The caching workshop: a page with wrong caching behaviour made correct and explicit — plus a Partial Prerendering build serving a static shell with streamed dynamic holes

TOPICS

- What is dynamic by default; explicit opt-in caching; cache lifetimes and tag-based invalidation
- On-demand and path-based revalidation; incremental static regeneration
- Partial Prerendering — static shell plus streamed dynamic holes; debugging a page that caches when it shouldn't

Assessment: Quiz + Lab**Module 4: Data Fetching & Mutations**

7 Hours

YOU SHIP

- A mutation-heavy interface on Server Actions — optimistic updates, validation, pending states and real error handling — with fetching parallelised where it matters

TOPICS

- Server-side data access patterns; request deduplication; parallel versus sequential fetching; Suspense and streaming UI
- Server Actions for mutations; form handling and pending states; optimistic updates; validation and error handling
- Where a client-side query library still earns its place

Assessment: Quiz + Lab**Module 5: Full-Stack Next.js**

7 Hours

YOU SHIP

- A full-stack feature slice: typed ORM schema and migrations, route handlers, file uploads to object storage, a webhook consumer and a scheduled job — with server code that never leaks into the client bundle

TOPICS

- Route handlers and API design; connecting to databases with a typed ORM; schema and migrations
- File uploads and object storage; background and scheduled work; webhooks and third-party integrations
- Structuring server code that doesn't leak into the client bundle

Assessment: Quiz + Lab**Module 6: Authentication, Authorisation & the Network Boundary**

7 Hours

YOU SHIP

- A secured application: OAuth sign-in, role-based access, protected routes and server actions, hardened headers — with boundary logic living in the proxy layer where it belongs

TOPICS

- Session and token strategies; auth providers and OAuth; protecting routes and server actions; role-based access
- The proxy/middleware layer and what belongs at that boundary versus in a component
- CSRF and common web vulnerabilities; secrets handling; security headers

Assessment: Quiz + Lab**Module 7: Performance & Core Web Vitals**

6 Hours

YOU SHIP

- A performance rescue on a slow app — bundle analysis, image/font/script optimisation, streaming strategy — with before/after Core Web Vitals measurements

TOPICS

- Bundle analysis and what actually ships to the client; image, font and script optimisation
- Prefetching behaviour; streaming strategy; third-party script cost
- Measuring rather than guessing

Assessment: Quiz + Lab

Module 8: Testing, Quality & Developer Experience

5 Hours

YOU SHIP

- A tested feature slice — server components and actions covered, end-to-end flows in Playwright, type safety across the boundary — running as CI gates

TOPICS

- Testing server components and actions; integration testing; end-to-end coverage with Playwright
- Type safety across the server/client boundary; linting and CI gates; monorepo considerations
- Framework devtools including MCP-based debugging workflows

Assessment: Quiz + Lab

Module 9: Deployment, Observability & Capstone

6 Hours

YOU SHIP

- The migration exercise — an implicit-caching-era app upgraded and fixed — and your capstone: one deployed, monitored, production-grade application defended on architecture

TOPICS

- Deployment targets and their genuine trade-offs (managed platform, self-hosted Node, containers, edge); environment configuration; caching at the CDN layer
- Logging, error tracking and analytics; cost
- The upgrade and migration track for apps built on the older implicit-caching model; capstone design, build, deploy and defense

Assessment: Capstone Project + Review