

# Next.js roadmap



next js full roadmap

Here's a **complete Next.js roadmap (2025 edition)** — designed to take you from **beginner** → **advanced** → **production-ready developer** 🚀

It's structured step-by-step with **topics, explanations, and learning goals** — ideal if you already know **HTML, CSS, JS, and React** (which you do 🙌).



## PHASE 0: Prerequisites

Before diving into Next.js, make sure you are confident with:



### Web Fundamentals

- HTML5 (semantic tags, forms, accessibility)
- CSS3 (Flexbox, Grid, Responsive design)
- JavaScript (ES6+ features: promises, async/await, destructuring, modules)



### React Essentials

- Functional components & hooks (`useState`, `useEffect`, `useContext`)
- Component lifecycle & props
- Conditional rendering & lists
- Basic project setup with Vite or CRA



**Goal:** Be able to create and deploy a basic React app confidently.



## PHASE 1: Next.js Basics (App Router)

Learn the fundamentals of Next.js (v14 or newer)

### ♦ 1. Introduction

- What is Next.js? Why use it over React?
- Core features: Server-Side Rendering (SSR), Static Site Generation (SSG), API Routes, Routing, Image Optimization.

### ♦ 2. Project Setup

- Install:

```
bash

npx create-next-app@latest my-next-app
```

- Understand the project structure (**app/**, **pages/**, **public/**).

### ♦ 3. Routing System (App Router)

- **File-based routing:** `/app/page.tsx` → `/`
- **Nested routes:** `/app/blog/page.tsx` → `/blog`
- **Dynamic routes:** `/app/blog/[slug]/page.tsx`
- **Catch-all routes:** `[...slug].tsx`

 Learn difference between:

- **pages/** (legacy routing)
- **app/** (new app router with React Server Components)

## PHASE 2: Rendering & Data Fetching

Understand how data is fetched and rendered on the server/client.

### ♦ 1. Types of Rendering

- **Static Site Generation (SSG)** — pre-rendered at build time
- **Server-Side Rendering (SSR)** — rendered at request time
- **Client-Side Rendering (CSR)** — rendered in browser
- **Incremental Static Regeneration (ISR)** — revalidate pages after deployment

### ♦ 2. Data Fetching Methods

In **app/** router:

- `fetch()` with async components
- `getStaticProps`, `getServerSideProps` (legacy pages)
- `generateStaticParams` for dynamic routes


 Learn: How to cache, revalidate, and stream data.

## PHASE 3: Styling & UI

Multiple ways to style Next.js apps.

## ◆ 1. CSS Options

- Global CSS & CSS Modules
- SCSS/SASS
- Tailwind CSS
- Styled Components / Emotion
- Shadcn UI (modern component library)

 Goal: Be able to create a consistent, responsive UI.

---

## PHASE 4: Navigation & State Management

### ◆ 1. Routing Features

- `<Link>` & `useRouter()`
- `usePathname`, `useSearchParams`
- Loading UI: `loading.tsx`
- Error UI: `error.tsx`
- Layouts: `layout.tsx` (nested layouts)

### ◆ 2. Global State

- Context API
  - Zustand / Redux Toolkit
  - Server Actions (new feature for form handling without API routes)
- 

## PHASE 5: Authentication & Authorization

### ◆ 1. Authentication Options

- NextAuth.js (best for most apps)
- Clerk / Supabase Auth
- Custom JWT-based Auth

### ◆ 2. Protecting Routes

- Middleware (`middleware.ts`)
  - Conditional Rendering (client + server)
- 

## PHASE 6: Database & Backend

### ◆ 1. API Routes

- `/app/api/route.ts` Or `/pages/api/`
- Handle GET, POST, PUT, DELETE
- Return JSON responses

## ♦ 2. Database Integration

- Prisma (ORM) + PostgreSQL / MongoDB
- PlanetScale / Supabase for hosted DBs
- Server Actions with Prisma (no API routes)

## PHASE 7: File Uploads & Cloud Integrations


- Image Upload (Cloudinary / AWS S3)
- File storage & access control
- Edge functions & caching

## PHASE 8: Performance & Optimization

- Image optimization (`next/image`)
- Font optimization (`next/font`)
- Code splitting & lazy loading
- Metadata & SEO (`metadata` API)
- Lighthouse performance testing

## PHASE 9: TypeScript Integration

- Convert files to `.tsx`
- Define prop types & interfaces
- Use `NextPage`, `AppProps`, and `InferGetServerSidePropsType`

 Goal: Write strongly typed, maintainable Next.js code.

## PHASE 10: Deployment & DevOps







- Vercel (most seamless)
- Netlify / Render / Railway
- Environment variables management
- Build optimization & CI/CD

## PHASE 11: Advanced Concepts

- Middleware for route guards
- Edge & Serverless Functions
- Server Actions (v14+)
- Streaming with React Suspense
- Internationalization (i18n)
- Analytics with Vercel / PostHog

## PHASE 12: Build Real Projects

Here are **project ideas** to apply your skills:

1.  **E-commerce App** (with products, cart, checkout)
2.  **Blog Platform** (SSG + MDX + comments)
3.  **Portfolio Website**
4.  **Chat App** (Next.js + Socket.io + Prisma)
5.  **Expense Tracker** (Clerk + PostgreSQL + Server Actions)
6.  **AI Route Finder or Smart Dashboard** (for your ambulance project idea)

## Suggested Timeline (8–10 Weeks)

Week	Focus	Topics
1–2	Fundamentals	Next.js basics, routing, layouts
3–4	Data & UI	Fetching, SSR/SSG, styling
5	Auth	NextAuth / Clerk
6	API & DB	API routes, Prisma, CRUD
7	Advanced	Middleware, ISR, SEO
8–10	Projects	Build + deploy 2 real-world projects

Would you like me to make this into a **visual roadmap (diagram)** or a **Notion checklist** you can follow step-by-step?