Next.js Notes

Chapter 1 —> Birth

JavaScript Evolution

- Created by Brendan Eich (1995) at Netscape.
- Next.js was created in **2016** by **Vercel** (led by Guillermo Rauch) to address React's limitations.
- Framework progression: jQuery → Angular → Node.js → React.js → Next.js

Hello World Example

- **Vanilla JS**: Verbose DOM manipulation.
- **jQuery**: Simplified syntax.
- **Angular/React**: More code for this example but scalable in "bigger picture" (component-based).

Why Modern Frameworks?

- **Component Architecture**: Reusable UI pieces (e.g., buttons).
- **Virtual DOM**: Efficient UI updates (only changes rendered).
- **Ecosystem**: Strong community, documentation, and tools.
- Modern frameworks improve efficiency, scalability, and performance.

Chapter 2 —> Introduction

Next.js is a **full-stack web framework** built on top of **React.js** or simply we can say it's a React framework. While React is a **UI library** that focuses on building components, Next.js extends it into a complete framework for building **production-grade web applications**.

What is a Framework?

- A framework serves as a tool equipped with predefined rules and conventions that offer a structured approach for building applications.
- Handles database integration, routing, authentication, etc.
- Helps developers focus on writing application logic rather than low-level setups.

Key features of Next.js:

1. Solves React limitations (SEO, routing, performance)
2. Built-in features:
    - File-based routing
    - Efficient code splitting
    - Hybrid rendering (SSR/SSG)
    - Built-in optimizations (images, fonts, SEO)
    - HMR (Hot Module Replacement)
    - API Routes (backend)
    - Built-in support for Sass
    - CSS modules
    - Data fetching choice (SSG, SSR, ISR)
    - Error handling
    - Metadata API (For SEO)
    - Internationalization(support for any spoken language), etc.

Why Use a React Framework like Next.js?

1. Less Tooling Time
    - No need to configure bundlers, compilers, formatters, etc.
    - Built-in support for routing, rendering, auth, and more.
    - Focus more on business logic and React code.

2. Easy Learning Curve
    - Easier to learn if you're already familiar with React.
    - Includes backend features but without complex setup (no routing config needed).
3. Improved Performance
    - Built-in SSR (Server-Side Rendering) & SSG (Static Site Generation).
    - Automatic code splitting for faster page loads and better UX.
    - React has introduced React Server Components for SSR, but Next.js automates the setup.

    > Follows "Convention over Configuration" = less boilerplate code.

4. SEO Advantage
    - React.js renders everything on the client side, sending a minimal initial HTML response from the server. The server sends a minimal HTML file code and a JavaScript file that the browser executes to generate the HTML —hard for search engines to crawl.
    - Next.js sends **full HTML file** and minimal JavaScript code to render only the content requiring client-side interaction.
    - This improves:
        - Visibility
        - Ranking
        - Traffic
        - User trust

When to Use Next.js over React

Choose **Next.js** when:

- You care about **SEO**

- You want **fast page loads** (via SSR/SSG)
- You don’t want to configure everything yourself
- You want an all-in-one full-stack React framework
- You need **routing, data fetching**, and **backend API** in one codebase

Choose **React (only)** when:

- You're building a **simple SPA or PWA**
- You need complete control over the setup
- You’re integrating into an existing app (e.g., with a non-React backend)