# 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)

# Chapter 3 —> Prerequisites

## Web Development Fundamentals

1. HTML -
    a. Structure
    <!DOCTYPE>,<html>,<head>,<body>
    b. Elements
    headings, paragraph, lists, <a>, <img>, <input>, <textarea>, <button>, <div>
    c. Semantics
    header, nav, main, section, aside, footer

```
<header>Site Logo/Navigation</header>
<nav>
  <a href="/">Home</a> | <a href="/about">About</a>
</nav>
<main>
  <section id="intro">
    <h2>Welcome</h2>
    <p>Introduction text...</p>
  </section>
  <aside>Related links (Content indirectly related to main content)</asi
```

```
de>
</main>
<footer>Copyright © 2024</footer>
```

d. Forms
handling user input, perform form validations by using form element and onSubmit event listener

```
<form onsubmit="validateForm()">
  <label for="name">Name:</label>
  <input type="text" id="name" required>

  <label for="email">Email:</label>
  <input type="email" id="email" required>

  <button type="submit">Submit</button>
</form>
```

2. CSS -

   a. Structure
   Box model - padding, margin, border
   Selectors - type, class, id, child, sibling
   Typography - font, size, weight, alignment
   Colors & Background - colors, gradients, background images

```
/* Box model */
div {
  width: 300px;
  padding: 20px;    /* Inner space */
  border: 2px solid black;
  margin: 30px;     /* Outer space */
}

/* Type */ h1 { color: blue; }
/* Class */ .btn { background: red; }
```

```css
/* ID */ #header { height: 80px; }
/* Child */ ul > li { list-style: none; }
/* Sibling */ h2 + p { margin-top: 0; }

body {
  font-family: 'Arial', sans-serif;
  font-size: 16px;
  line-height: 1.5;
  font-weight: 400/bold;
  text-align: center;
}

.element {
  color: #ffffff; /* Text color */
  background-color: rgba(0,0,0,0.5);
  /* A gradient is like a smooth blend of two or more colors. Instead of
  one solid color, the colors gradually change. */
  background: linear-gradient(to right/135deg, red, yellow);
  background-image: url('image.jpg');
}
```

b. Layout and Positioning
   Display - block, inline, inline-block
   Position - relative, absolute, sticky, fixed
   Flexbox & Grid

c. Effects
   Transitions - Learn to create smooth transitions using different CSS properties like delay, timing, duration, property, timing-function
   Transformations - Explore 2D and 3D transformations like scaling, rotating, translating elements
   Animations - Learn how to create animations using keyframes
   Shadows and Gradients - Explore with box shadows and linear or radial gradients

d. Advanced (Plus)
   Learn how to use CSS processors like sass or frameworks like

TailwindCSS for more powerful and efficient styling

3. JS -
    a. Variables and Data Types
    b. Operators
    c. Control Flow
    d. Functions
    e. DOM Manipulation