What is Next.js and why does it exist?

## Traditional React vs. Next.js: An Analogy 🏠

Let’s break down the difference between traditional React and Next.js using a simple analogy: building a house.

### The Traditional React Way: Client-Side Rendering (CSR)

When you use traditional React, which relies on **Client-Side Rendering (CSR)**, the process is like this:

1. **Order the Blueprint**: Your browser requests a webpage and receives a very basic, empty HTML file. Think of this as an empty plot of land with just a foundation.
2. **Receive the Entire Toolkit**: The browser then has to download a large JavaScript bundle. This is like a massive toolkit containing all the tools and raw materials (the React library, your application code, etc.) needed to build the house.
3. **Build the House On-Site**: Once all the tools and materials are downloaded, the browser’s JavaScript engine gets to work, building the entire house from scratch on the client’s device. It constructs the walls, adds furniture, and paints everything.

This approach has some significant limitations:

* **Slow Initial Load Time**: The user sees a blank screen or a loading spinner for a while because the browser has to download the entire toolkit *and then* build the house. This can lead to a poor user experience, especially on slower networks.
* **Poor Search Engine Optimization (SEO)**: Search engine crawlers (like Googlebot) might arrive at the empty plot of land and not wait for the entire house to be built. If they only see the initial, empty HTML file, they can’t index the actual content, which can hurt your search rankings.
* **Performance Bottlenecks**: For large applications, the JavaScript “toolkit” can become huge, increasing download and execution time. This slows down the **Time to Interactive (TTI)**, the point at which a user can actually interact with the page.

### The Next.js Solution: Pre-Fabricated Homes

Now, let’s introduce **Next.js**. Think of Next.js as a specialized, high-tech construction company that builds parts of the house, or the entire house in a factory before delivering it.

Next.js is a React framework that provides different rendering strategies to overcome the limitations of CSR. The two most powerful are:

* **Server-Side Rendering (SSR)**: With SSR, the complete, fully-furnished house (a fully-formed HTML page) is built on the server for each request. The browser receives a ready-to-display page.
  + **Benefit**: The user sees content almost instantly, dramatically improving perceived performance.
  + **Benefit**: Search engine crawlers get a complete HTML file, making it easy for them to read and index the page content for good SEO.
* **Static Site Generation (SSG)**: This is a form of pre-building. With SSG, the house (the HTML page) is pre-built at build time (when you deploy your app) and stored, ready to be served instantly from a Content Delivery Network (CDN). Think of this as mass-producing a popular model of a house and having it ready for immediate delivery.
  + **Benefit**: This approach is extremely fast because there’s no server-side computation needed when a user requests the page. It’s perfect for content that doesn’t change often, like blog posts, documentation, or marketing pages.

# What is Next.js and why does it exist?

Next.js is a flexible React framework that gives you building blocks to create fast, full-stack web applications. –> Framework: By framework, we mean Next.js handles the tooling and configuration needed for React, and provides additional structure, features, and optimizations for your application.(Vercel, 2024)

Server-side rendering (SSR) & static site generation (SSG)​ Optimized for performance and SEO​ Full-stack capabilities with API routes

websites built with Next.js load faster and rank better in search results (Sanity, 2024)

# References

Sanity. (2024). *Next.js overview*. <https://www.sanity.io/glossary/next-js>

Vercel. (2024). *Next.js documentation*. <https://nextjs.org/docs>