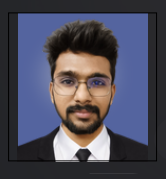
Comprehensive Guide to Client-Side Rendering (CSR) in Next.js

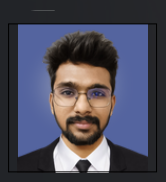
: A Complete Overview 🚦⏭️



Jimmy Ramani
@jimmyramani

What is Client-Side Rendering (CSR)?

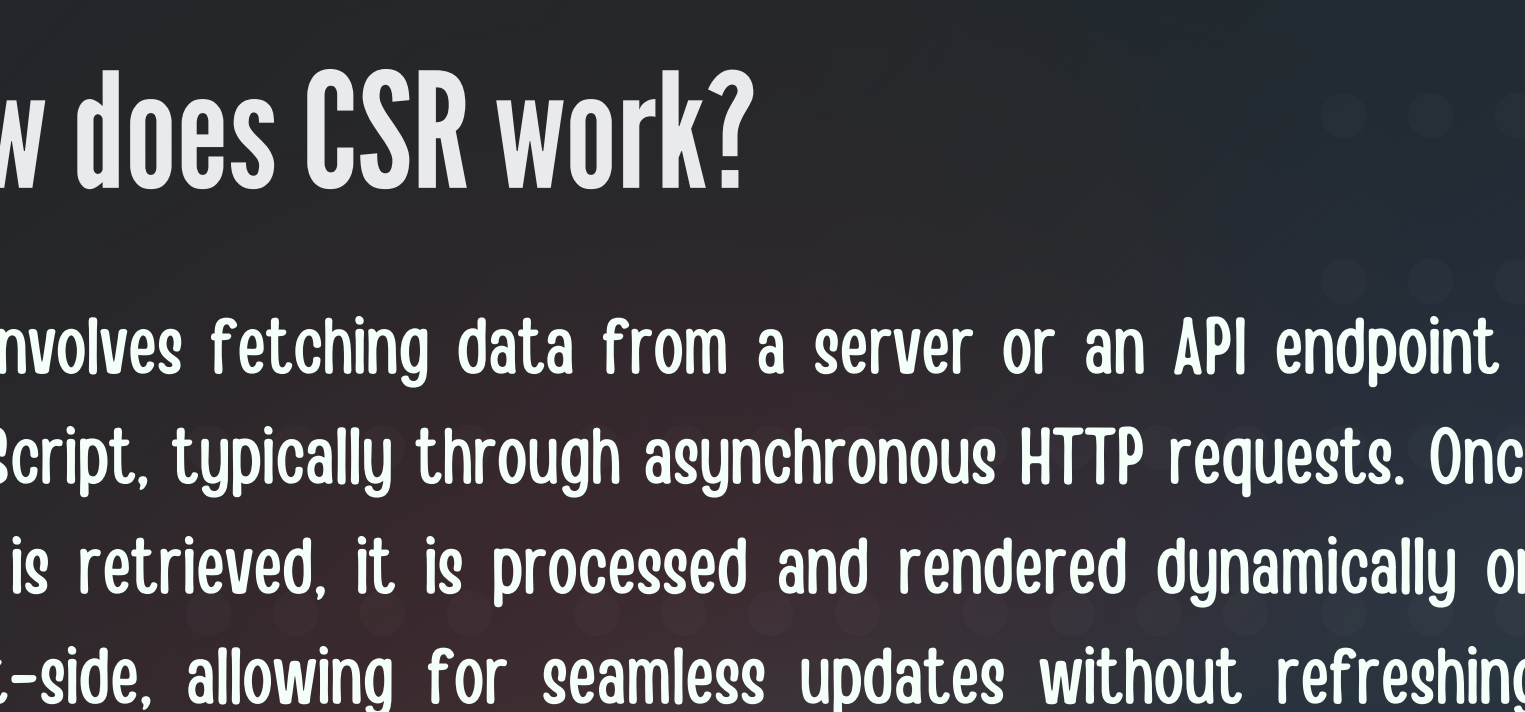
Client-Side Rendering (CSR) is a technique where the browser is responsible for generating the HTML content of a web page dynamically using JavaScript. Unlike server-side rendering (SSR) or static site generation (SSG), where HTML is generated on the server, CSR relies on JavaScript to fetch data from an API and manipulate the DOM to render content on the client-side.



Jimmy Ramani
@jimmyramani

How does CSR work?

CSR involves fetching data from a server or an API endpoint using JavaScript, typically through asynchronous HTTP requests. Once the data is retrieved, it is processed and rendered dynamically on the client-side, allowing for seamless updates without refreshing the entire page. This approach provides a more interactive user experience but may result in slower initial page load times compared to SSR or SSG.



Jimmy Ramani
@jimmyramani

Example

Let's create a simple example to illustrate CSR in Next.js.

```
// pages/index.js

import { useState, useEffect } from "react";

export default function Home() {
  const [posts, setPosts] = useState([]);

  useEffect(() => {
    fetchPosts();
  }, []);

  const fetchPosts = async () => {
    try {
      const response = await fetch("https://api.example.com/posts");
      const data = await response.json();
      setPosts(data);
    } catch (error) {
      console.error("Error fetching posts:", error);
    }
  };

  return (
    <div>
      <h1>Latest Posts</h1>
      <ul>
        {posts.map((post) => (
          <li key={post.id}>{post.title}</li>
        ))}
      </ul>
    </div>
  );
}
```

In this example, we use React's useState and useEffect hooks to fetch posts from an API endpoint (https://api.example.com/posts) when the component mounts. The fetched data is then stored in the component's state and rendered dynamically in the DOM.

Jimmy Ramani
@jimmyramani

When to use CSR?

- Highly Interactive Applications:
  CSR is ideal for applications with dynamic user interfaces that require frequent updates and interactions, such as single-page applications (SPAs) and progressive web apps (PWAs).

- Real-Time Data Updates:
  If your application relies on real-time data updates or user interactions, CSR allows for seamless updates without reloading the entire page.

- Asynchronous Content Loading:
  CSR is suitable for scenarios where content is fetched asynchronously from multiple sources or APIs.

Jimmy Ramani
@jimmyramani

Real-life Use Cases :

- Social Media Platforms:
  Social media platforms like Twitter and Facebook use CSR to deliver real-time updates on user feeds, notifications, and messages.

- Web-based Email Clients:
  Web-based email clients such as Gmail use CSR to load emails and update the interface dynamically without reloading the entire page.

- Data Dashboards:
  Data visualization dashboards often rely on CSR to fetch and display real-time data from various sources.

Jimmy Ramani
@jimmyramani

Jimmy Ramani

🚀 Join ME on a journey of Full Stack Development ! 🌍