How does SSR(Server-Side Rendering) differ from CSR(client-side rendering) ?

Last Updated : 16 Apr, 2024

Server-Side Rendering (SSR) and **Client-Side Rendering (CSR)** are two different approaches used in web development to **render web page**s to users. Each approach has its own set of advantages and disadvantages.

In this article, we will learn about the difference between SSR and CSR with examples and features.

Table of Content

Server-Side Rendering (SSR)

Server-side rendering is the process of **rendering(loading)** the web pages on the **server side** and sending the fully rendered HTML to the client. In this, the server generates HTML dynamically based on the requested URL and data then sends it to the client.

Features:

- It reduces the time to load initial page by delivering pre-rendered HTML directly to the client.
- Search engines can easily crawl and index pages rendered on the server side.
- It is rendered on server side so users can see content quicker, especially on slower connections or devices.
- It ensures that basic content is available to users even if JavaScript is disabled or fails to load.

- Devices with limited processing power benefit from SSR as it reduces the amount of client-side computation required to render the page.

Client-Side Rendering (CSR)

Client side rendering is the process of rendering web pages on the client side using JavaScript after the initial HTML is loaded. In this, the browser loads a minimal HTML document then JavaScript retrieves data from the server and generates the HTML dynamically.

Features:

- It allows for dynamic content loading without refreshing the entire page.
- In this, the web applications can provide highly interactive user interfaces and create a complex interactions such as drag-and-drop, real-time updates etc..
- Once the initial page is loaded, subsequent interactions typically result in faster response times since only the necessary data is fetched from the server
- It allows us to do asynchronous data loading.

Steps to Initialize Node Application and install required modules

Step 1: Create a NodeJS application using the following command:

```
npm init -y
```

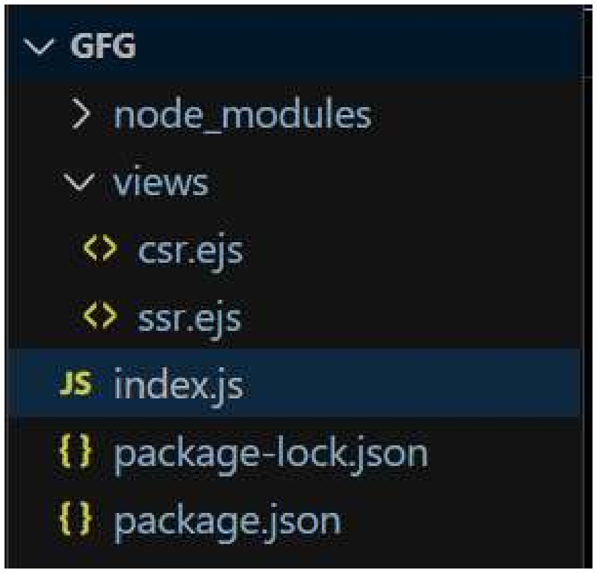
Step 2: Install required Dependencies:

```
 npm i ejs express
```

The updated dependencies in package.json file will look like:

```
"dependencies": {
    "ejs": "^3.1.9",
    "express": "^4.19.2"
}
```

Folder Structure:



```
GFG
  node_modules
  views
    csr.ejs
    ssr.ejs
  index.js
  package-lock.json
  package.json
```

Example: The below example demonstrate the SSR and CSR.

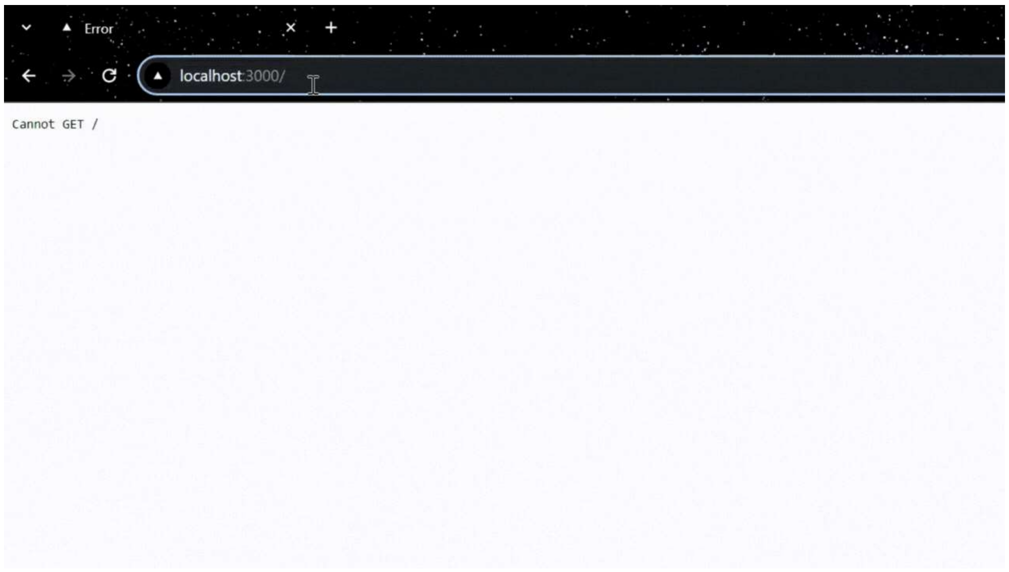
HTML HTML JavaScript

```
<!-- File path: views/csr.js -->

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSR Example</title>
</head>
<body>
    <h1 style="color: green;">GeeksForGeeks | Client Side Rendering</h1>
    <button onclick="showData()">Show Data</button>
    <div id="dataContainer"></div>

    <script>
        async function showData() {
            const response = await fetch('/api/data');
            const data = await response.json();
            document.getElementById('dataContainer').innerText =
JSON.stringify(data);
        }
    </script>
</body>
</html>
```

To run the application use the following command

```
node index.js
```

Output: Now go to **http://localhost:3000/ssr** and **http://localhost:3000/csr** in your browser:



```
Cannot GET /
```

Difference between SSR and CSR

SSR	CSR
SSR stands for Server-Side Rendering	CSR stands for Client-Side Rendering
It renders the page at server side	It renders the page at client side
It is a more SEO friendly	It is a less SEO friendly
User interactivity is Limited	User interactivity is Highly interactive
It consumes the server resources	It consumes the client resources
It gives better performance on low Powered Devices	It may not give better performance on low Powered Devices
It may require more server resources to handle rendering tasks.	It doesn't require more server resources to handle rendering tasks.