Server-Side Rendering

12 min

While client-side rendering allows websites to be dynamic and interactive, constructing and loading the page all on the client's hardware can be an expensive endeavor. A slow internet connection or slow client hardware can compound the wait time for the user, who is busy staring at nothing as the browser assembles the webpage.

In server-side rendering (SSR), the webpage is assembled on the server. Offloading the rendering to the server leverages the more capable server infrastructure, reducing the load on the client's hardware.

Server-side rendering is ideal for web applications that need a lot of data fetching, search engine optimization (SEO), and speed. By moving the fetching

Preview: Docs Loading link description

requests

closer to the database, developers reduced the latency of these requests. Sending fully rendered pages also means users can view them immediately when visiting a website, regardless of their hardware's capabilities. The rendered pages can then be crawled and indexed by search engine bots, leading to better SEO.

By default, Next.js renders components on the server side. You can define a React component without additional configurations, as Next.js automatically handles the server-side rendering.

Within server-side rendering itself, Next.js offers three distinct approaches: static rendering, dynamic rendering, and streaming. In a later lesson on Server Components, we will delve deeper into these subsets and their specific applications.

At this stage, while the page is visible and contains elements like buttons and form fields, they are not fully interactive. In a sense, it is like a fresh first layer of paint on our page. In the next exercise, we'll be discussing what comes after to make the web application interactive.

Instructions

1. Checkpoint 1 Passed

1.

In the current code, we have a function that fetches the current time when the page is requested. Let's implement another component that will display that value.

In page.tsx, under the fetchCurrentTime function, define a new component with the name TimePage.

Hint

A React function component can be declared with the following syntax:

function ComponentName() {}

2. Checkpoint 2 Passed

2.

In the component, get the value of fetchCurrentTime and save it to a variable named currentTime.

3. Checkpoint 3 Passed

3.

Return a <p> tag with the text "The current time is:" and the currentTime.

Then, default export the component.

Hint

Use JSX where necessary to display the currentTime variable.

4. Checkpoint 4 Passed

4.

Run the code using npm run dev in the terminal.

Once you see a console message indicating “Ready,” click the Refresh Browser button in the workspace browser bar.

Notice that the time does not update. It is a static representation of the time when the page was first requested and rendered by the server.

Note that the server uses the UTC timezone, so it may not reflect your local time.

Hint

Enter the command inside the terminal window.

The window can be refreshed when it displays “Ready.”

```
> dev
> next dev -p 4001

  ▲ Next.js 14.0.1
  - Local:        http://localhost:4001

 ✓ Ready in 1156ms
```

page.tsx

```
import React from 'react'

const fetchCurrentTime = (): string => {
  return new Date().toLocaleString();
};

function TimePage() {
  const currentTime = fetchCurrentTime();
  return (
    <p>The current time is: {currentTime}</p>
  )
}

export default TimePage;
```