Deeksha Sharma

Access Data from an External API into a React Component

Deeksha Sharma

May 27, 2020 • 9 Min read • 26,104 Views

May 27, 2020 • 9 Min read • 26,104 Views

Web Development | Front End Web Dev... | Client-side Framew... | React

Introduction

44

Introduction

Fetching data from an external API and rendering React components with that data is a common task in building React apps. In this guide you will learn how to fetch JSON data from the GitHub Users API and render that data inside a React component. This will help you make asynchronous requests initiated by the browser (including those across the network) for fetching resources.

Approaches

There are two approaches you will leverage to get the data from the network.

- Fetch, a Web API available in browsers to fetch network resources.
- Axios, a `Promise` based `npm` library for browser and node.js.

You will fetch data from the GitHub Users API for a specific username and render a React component that displays the user's details, including the name, location, blog URL, and company name for that user.

Set up a React App

The first step is to set up a React app. Open your terminal and run these commands to get a sample Create React App (CRA) running on your machine.

sh

```
npx create-react-app access-api-react

```

```
cd access-api-react

yarn start
```

This starts your app in development mode. To check if the sample app is running, open http://localhost:3000 in your browser. You should see a React logo.

Come back to your terminal and add the `npm` dependency for `axios` inside the root of the project. You will need this library to send a request to the GitHub API.

sh

```
yarn add axios
```

Modify the App Component

Open the project in your favorite editor and remove all the boilerplate code from the `<App>` component. You will find the file inside your `src/` directory. Add the code below in your component.

js

```
import React, { useState, useEffect } from "react";
import "./App.css";

```

```
4     const gitHubUrl = "https://api.github.com/users/deekshasharma";
5
6     function App() {
7       const [userData, setUserData] = useState({});
8
9       useEffect(() => {
10        getGitHubUserWithFetch();
11      }, []);
12
13      const getGitHubUserWithFetch = async () => {};
14
15      return (
16        <div className="App">
17          <header className="App-header">
18            <h2>GitHub User Data</h2>
19          </header>
20          <div className="user-container">
21            <h5 className="info-item">{userData.name}</h5>
22            <h5 className="info-item">{userData.location}</h5>
23            <h5 className="info-item">{userData.blog}</h5>
24            <h5 className="info-item">{userData.company}</h5>
25          </div>
26        </div>
27      );
28    }
29
30    export default App;
```

In the code above, `gitHubUrl` is a variable containing the GitHub API URL for the username *deekshasharma*. It's declared as a `const` because this value will not change.

The State Hook below will allow the use of state in the `<App>` function component without writing a separate class component. `userData` is an object that is initially empty (`useState({})`). Once the data is fetched from the network, it will contain the user data (name, location, blog, and company). `setUserData` is equivalent to writing `this.setState()` to update the component state with the value of `userData`.

```js
const [userData, setUserData] = useState({});
```

Next is the Effect Hook, which will allow you to perform side effect operations such as fetching data, clean up, or DOM manipulation. `useEffect()` takes as argument a function that will execute after the first render and after every component update. So this function is an apt place to call the `getGitHubUserWithFetch()` function, whose job is to get the user data from GitHub API and update the component. Passing a second argument to `useEffect` is optional. Passing an empty array `[]` ensures this effect will run just once; otherwise, it will run after every render.

```js
useEffect(() => {
  getGitHubUserWithFetch();
}, []);
```

`getGitHubUserWithFetch()` is an `async` function without implementation. We will write code inside this function shortly.

Finally, the `<App>` component returns a header containing the text "GitHub User Data" and the user data (name, location, blog, and company) rendered inside a `<div>` element.

```js
return (
  <div className="App">
    <header className="App-header">
      <h2>GitHub User Data</h2>
    </header>
    <div className="user-container">
      <h5 className="info-item">{userData.name}</h5>
      <h5 className="info-item">{userData.location}</h5>
      <h5 className="info-item">{userData.blog}</h5>
      <h5 className="info-item">{userData.company}</h5>
    </div>
  </div>
);
```

Replace the contents in your `src/App.css` with the code below. The existing styles are slightly changed and new ones are added to display user data on the web page.

```css
.App {
  text-align: center;
}
```

```
4
5       .App-header {
6         background-color: #282c34;
7         min-height: 10vh;
8         display: flex;
9         flex-direction: column;
10        align-items: center;
11        justify-content: center;
12        font-size: calc(10px + 2vmin);
13        color: white;
14      }
15
16      .user-container {
17        display: flex;
18        justify-content: center;
19        align-items: center;
20        flex-direction: column;
21      }
22
23      .info-item {
24        width: 15vw;
25        height: 5vh;
26        padding: 2em;
27        border: 2px solid rgb(159, 12, 22);
28        align-self: center;
29        background-color: #63b7af;
30        border-radius: 2em;
31      }
```

Get Data Using Fetch API

Here is how to implement the `getGitHubUserWithFetch()` function. The code below uses `async/await` so that asynchronous code is readable and appears to be executing synchronously.

Though `async` functions always return a `Promise`, there is no need to return a value in this case since `setUserData()` is called with the resulting `jsonData` within the function itself.

Now comes the `fetch()` API, which takes as argument a URL, makes a network request to that URL, and returns a `Promise` that resolves to a `response` object.

The `await` keyword in front of the `fetch()` function pause the code until `fetch()` returns a fulfilled `Promise`, after which it returns the resolved value `response`.

The `response` is then parsed into a JSON using the `json()` method, which also returns a `Promise` that resolves to the JSON object `jsonData`. The use of `await` yet again pauses the code until the response is parsed to JSON. Then `setUserData()` is called with the resulting JSON object `jsonData`. It updates the state of the `<App>` component with GitHub user data.

```js
const getGitHubUserWithFetch = async () => {
  const response = await fetch(gitHubUrl);
  const jsonData = await response.json();
```

```
    setUserData(jsonData);
  };
```

To verify your code changes, open your browser to http://localhost:3000 and you should be able to view the GitHub user details rendered as React components for the username *deekshasharma*.

Get Data Using Axios

The second approach to fetch the data from the GitHub API is by using the Axios library. You already installed the dependency in your project. All you need now is to import it inside `src/App.js`. So go ahead and add this import.

```
import axios from "axios";
```
js

Add another function, `getGiHubUserWithAxios()`, in your `<App>` component, which use Axios. Since it is a `Promise`-based library, you can seamlessly replace `Promise.then()` with `async/await`.

the `get()` method of the library takes as argument a URL and makes an `http` request to that URL. It then automatically transforms the response to JSON, which you can get from its `data` property. Once the data is received, the state of the component is updated via the `setUserData()` function.

```js
const getGiHubUserWithAxios = async () => {
  const response = await axios.get(gitHubUrl);
  setUserData(response.data);
};
```

Inside your `<App>` component, make sure you call `getGiHubUserWithAxios()` instead of `getGitHubUserWithFetch()` inside `useEffect()`.

```js
useEffect(() => {
  getGiHubUserWithAxios();
}, []);
```

Verify your code changes in the browser at http://localhost:3000, and you should see the data rendered in a React component for the username *deekshasharma*.

Conclusion

You now understand how to get data for your React components from an external API using `fetch` and `axios`. Both achieve the same objective and you can choose either of the two.

However, keep in mind that `axios` should be added as an `npm` dependency, whereas `fetch` is available out of the box as a web API. Another difference is that `axios` returns you the transformed JSON data, but with `fetch` you need to parse the response and extract the JSON before passing it to React components.

The code for this guide is available at GitHub.

These additional resources are worth reading if you want to learn more:

- Understanding Effect Hook in React
- Working with State Hook in React
- How Fetch API works
- Axios on Browser