March 15, 2023

The Future of Data Fetching in React

React has recently announced the development of a new hook that will provide first-class support for Promises. This means that fetching data from an API and rendering it with HTML in React will become even simpler.

Photo by Lautaro Andreani on Unsplash

In previous versions of React, the traditional way of fetching data involved using the `useState` and `useEffect` hooks to manage state and perform side effects. For example, the following code snippet shows how data was fetched and rendered in React 16+:

```
const Post = () => {
    const [post, setPost] = useState(null);
    const [loading, setLoading] = useState(true);
    useEffect(() => {
        axios
        .get('https://jsonplaceholder.typicode.com/posts/1')
        .then((res) => {
            setPost(res.data);
            setLoading(false);
        })
        .catch((err) => {
            console.log(err);
            setLoading(false);
        });
    }, []);
    return (
        <div>
        {loading ? (
            <div>Loading...</div>
        ) : (
            <div>
            <h1>{post.title}</h1>
            <p>{post.body}</p>
            </div>
        )}
        </div>
    );
};
// Parent component
<Post />
```

However, with React 18 and the introduction of `Suspense`, the code has become simpler. The following code snippet shows the updated code using React 18:

```
const Post = () => {
    const [post, setPost] = useState(null);
    useEffect(() => {
        axios
        .get('https://jsonplaceholder.typicode.com/posts/1')
        .then((res) => {
            setPost(res.data);
        })
        .catch((err) => {
            console.log(err);
        });
    }, []);
    return (
        <div>
            <h1>{post.title}</h1>
            <p>{post.body}</p>
        </div>
    );
};

// Parent component
<Suspense fallback="Loading...">
    <Post />
</Suspense>
```

Despite this improvement, the `useEffect` hook can still be a source of annoyance for developers. To address this, React is developing a new hook called `use`, which provides first-class support for Promises.

With the `use` hook, the code for fetching and rendering data can be further simplified. The following code snippet shows how the use hook can be used to fetch and render data:

```
const Post = () => {
    const { data } = use(axios.get('https://jsonplaceholder.typicode.com/posts/
    return (
        <div>
            <h1>{data.title}</h1>
            <p>{data.body}</p>
        </div>
    );
};

// Parent component
<ErrorBoundary fallback="Error">
  <Suspense fallback="Loading...">
      <Post />
  </Suspense>
</ErrorBoundary>
```

You can extend React 18s error boundaries to handle the errors. Your component is much simpler now.

In summary, while the traditional way of fetching and rendering data using `useState` and `useEffect` hooks has been effective, but it can be a source of annoyance for developers. React's new use hook provides a simpler and more efficient way to fetch and render data in React applications, making the process even more streamlined and straightforward.

Remember, `use` hook is still under development and not recommended to use in production. Here's a demo.

reactjs typescript promises

§

Tweet Share