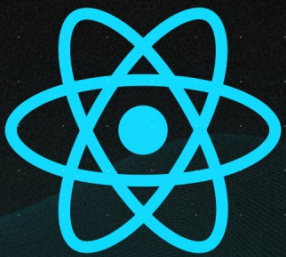
New API: ‘use’

follow me for other new react features



REACT 19

coming soon...

codewithsloba.com

Swipe →

Purpose

The use API in React 19 simplifies fetching data and working with asynchronous resources directly within a component's render function.

This eliminates the need for separate lifecycle methods or complex state management for loading and error states.

Swipe for code

How it works

1. **Import Suspense**: We import Suspense for handling the loading state.
2. **Define Async Function**: We define an async function `fetchData` that fetches data from the API.
3. **Call use**: Inside the component's render function, we call `use` with fetchData as an argument.
4. **Suspense Wrapper**: We wrap the content with Suspense and provide a fallback message ("Loading data...") while data is fetched.
5. **Render Data**: Once data is available, data from `use` is used to render the content (message in the example).

Swipe for example code

```jsx
import { Suspense } from 'react';

async function fetchData() {
  const response = await
fetch('https://api.example.com/data');
  return await response.json();
}

function MyComponent() {
  const data = use(fetchData);

  return (
    <Suspense fallback={<div>Loading data...</div>}>
      <div>
        <h1>My Data</h1>
        <p>{data.message}</p>
      </div>
    </Suspense>
  );
}
```

Benefit

Cleaner Code: The use API keeps your component logic concise and focused on UI rendering. It removes the boilerplate code typically needed for handling asynchronous operations.

Improved Readability: By integrating with React's Suspense mechanism, the use API makes the flow of data fetching and rendering more explicit, leading to easier code comprehension.

Reduced Errors: The automatic suspension during data fetching helps prevent rendering issues that might occur when data isn't available yet.

Real life Applications

Fetching User Data: The use API can be used to fetch user data from an API and display it on a profile page. The component suspends rendering until the user data is available, ensuring a smooth user experience.

Loading Comments: Imagine a blog post component that fetches comments from an API. The use API can handle this, suspending the rendering of comments until they are loaded, while showing a loading indicator in the meantime.

Real-time Data Updates: The use API can be used with libraries like WebSockets to fetch real-time data updates. The component suspends until the update arrives, then rerenders with the latest information.

codewithsloba.com

Get a weekly digest of my tips and tutorials by subscribing now.