Slim

React js tips

Mastering useState in React:

A Beginner’s Guide

Slim toumi

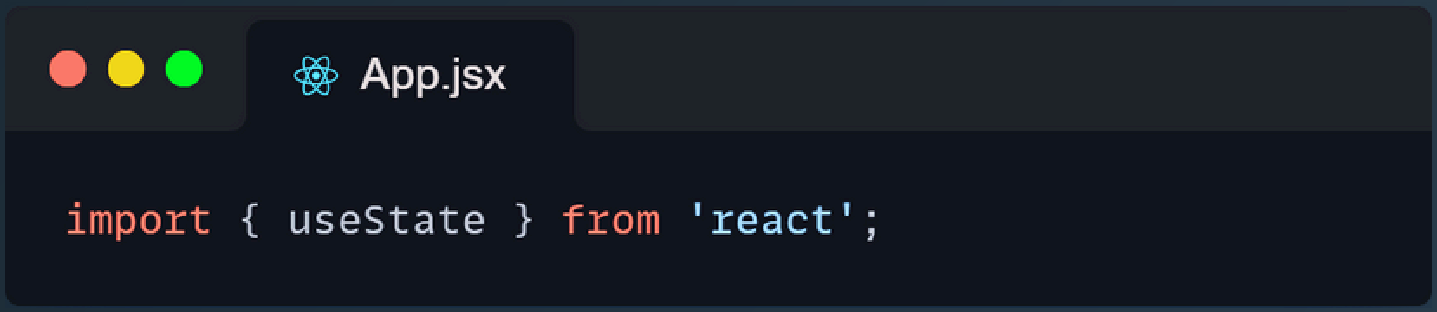
Slim toumi

What is useState ?

useState is a React hook that lets functional components manage state, a feature that was previously exclusive to class components.

Using useState

To start using useState, import it at the top of your component:

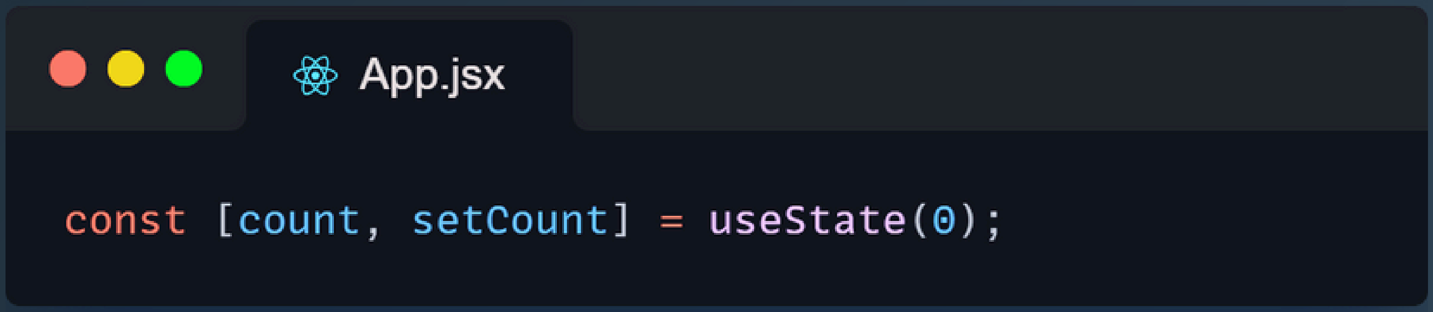


App.jsx

```
import { useState } from 'react';
```

Setting Up useState

Calling useState with an initial value creates state in a component. This returns an array with two elements



App.jsx

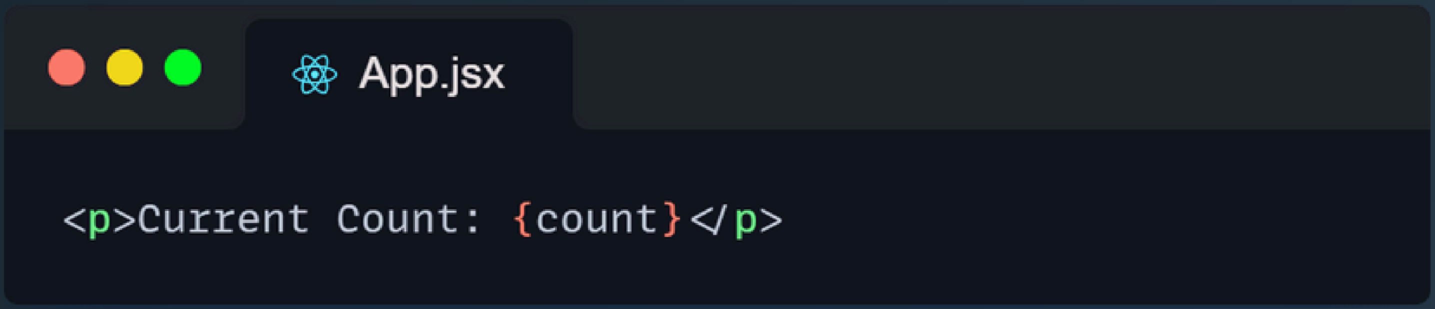
```
const [count, setCount] = useState(0);
```

Explanation :

- **count** : starts with an initial value of 0.
- **setCount** : is used to update count and re-render the component.

Displaying State in JSX

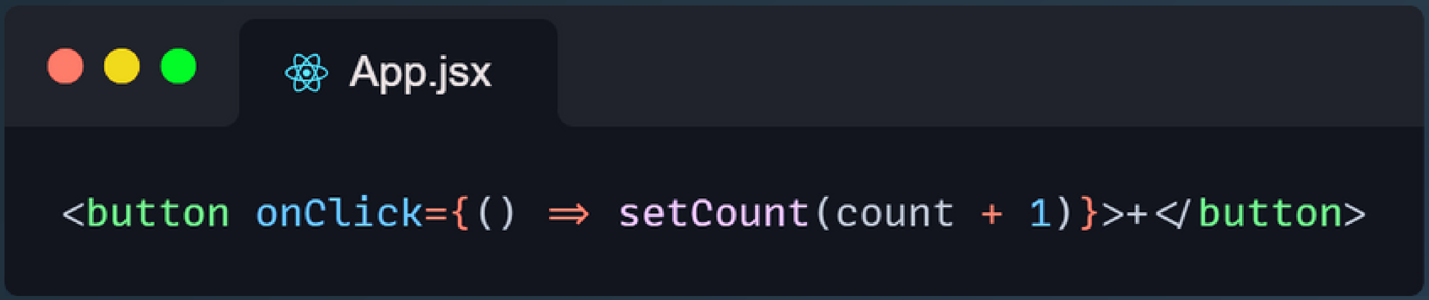
You can display the state directly in your JSX by referencing the state variable:



```jsx
<p>Current Count: {count}</p>
```

Updating State

To change the state value, call the updater function, **setCount**. This will schedule a re-render:



```jsx
<button onClick={() => setCount(count + 1)}>+</button>
```

Note:

setCount does not immediately update the state. Instead, it schedules an update, which is important if your updates depend on the previous state.

Functional State Updates

If the next state relies on the current state, use a functional update:

App.jsx

```
<button onClick={() => setCount(prev => prev + 1)}>+</button>
```

Note:
Using **prev** ensures that updates are accurate, especially when multiple updates are triggered in rapid succession.

Managing Multiple States

To manage different pieces of state, call useState multiple times:

App.jsx

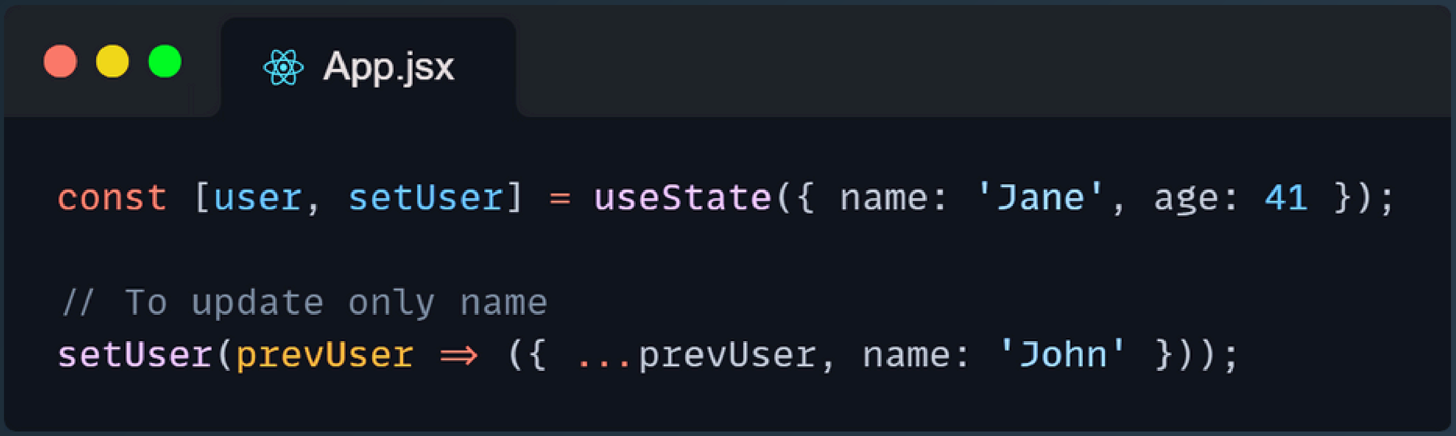
```
const [name, setName] = useState('');
const [age, setAge] = useState(0);
```

Note:
Each piece of state has its own variable and updater, keeping your code modular and clear.

Managing Complex State (Object Or Array)

When dealing with arrays or objects in state, avoid directly modifying them. Instead, use the spread operator to create a new version, which ensures React can track the update properly.

Example with an Object: Let’s say you have a state variable holding an object with user details, and you want to update just one part of it.


App.jsx

```jsx
const [user, setUser] = useState({ name: 'Jane', age: 41 });

// To update only name
setUser(prevUser => ({ ...prevUser, name: 'John' }));
```

Explanation :

- **...prevUser** copies all properties from the previous user state.
- **name: 'John'** updates the name while keeping the rest unchanged.

Note:
Why this is important: Directly modifying state can cause unexpected behavior because React may not detect the change. Always return a new object or array to let React know there’s an update.

Slim toumi
Slim toumi

Slim

Hopefully You Found It Usefull!

“Be sure to save this post so you can come back to it later”

like Comment Share

Slim toumi

Slim toumi