



useState in Functional Components (Hooks)

The `useState` hook allows you to manage **local state** in **function components** — a feature that was only available in class components before React 16.8.

Syntax

js

 Copy


 Edit


```
const [state, setState] = useState(initialValue);
```

- `state` → current value
- `setState` → function to update the value
- `initialValue` → starting value (e.g. `0`, `""`, `[]`, `{}`)

Simple Example

jsx

 Copy

 Edit

```
import React, { useState } from 'react'; function ClickCounter() { const [count, setCount] = useState(0); // initial value is 0 return ( <div> <h2>Clicked {count} times</h2> <button onClick={(() => setCount(count + 1))}>Click Me!</button> </div> ); }
```

Notes:

- `setCount` re-renders the component with the **new value**.
- You can use multiple `useState` calls for different variables.

Multiple States Example

jsx

 Copy Edit

```
function Profile() { const [name, setName] = useState("Vivek"); const [age, setAge] =
useState(22); return ( <div> <h3>{name}</h3> <p>Age: {age}</p> <button onClick={() => setAge(age +
1)}>Increase Age</button> </div> ); }
```

useState with Objects

jsx

 Copy Edit

```
const [user, setUser] = useState({ name: "Vivek", age: 22 }); const updateName = () => { setUser({
...user, name: "Rahul" }); // Spread keeps old values };
```

Common Mistake

Don't do this 🙅 — you'll lose other properties:

js

 Copy Edit

```
// This will remove "age" setUser({ name: "Rahul" });
```

Instead, always spread the old state unless you're replacing it all.