In React, when you render the same component multiple times on one page, each of these component instances operates independently of the others. This is a fundamental principle of React and is primarily due to the concept of component instances, which are isolated and maintain their own state and props.

```jsx
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
    </div>
  );
}

function App() {
  return (
    <div>
      <h1>Multiple Counters</h1>
      <Counter />
      <Counter />
      <Counter />
    </div>
  );
}

export default App;
```

Here's why each Counter component operates independently

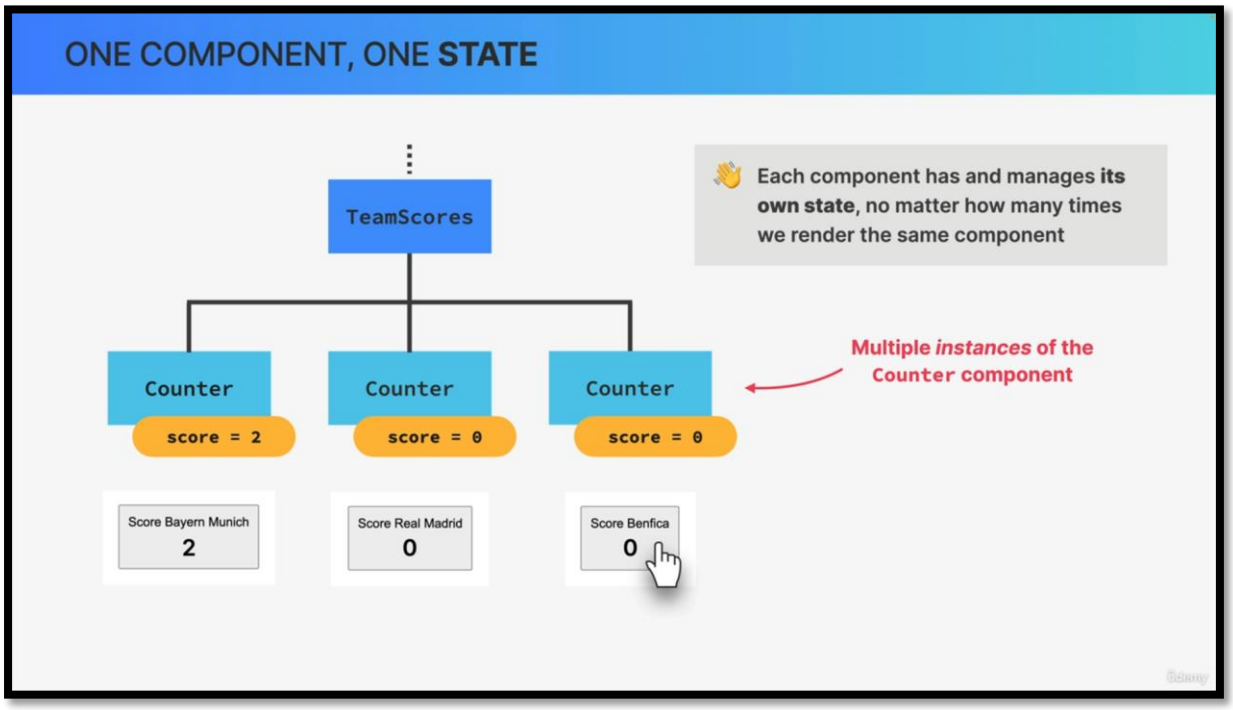
State Isolation
Each Counter component has its own state variable count created by useState. When you click the "Increment" button in one Counter, it only affects the state of that specific instance. Other Counter instances maintain their own separate state.

Props Independence
If you were to pass different props to each Counter instance, those props would also be specific to that instance. Props are not shared between component instances.

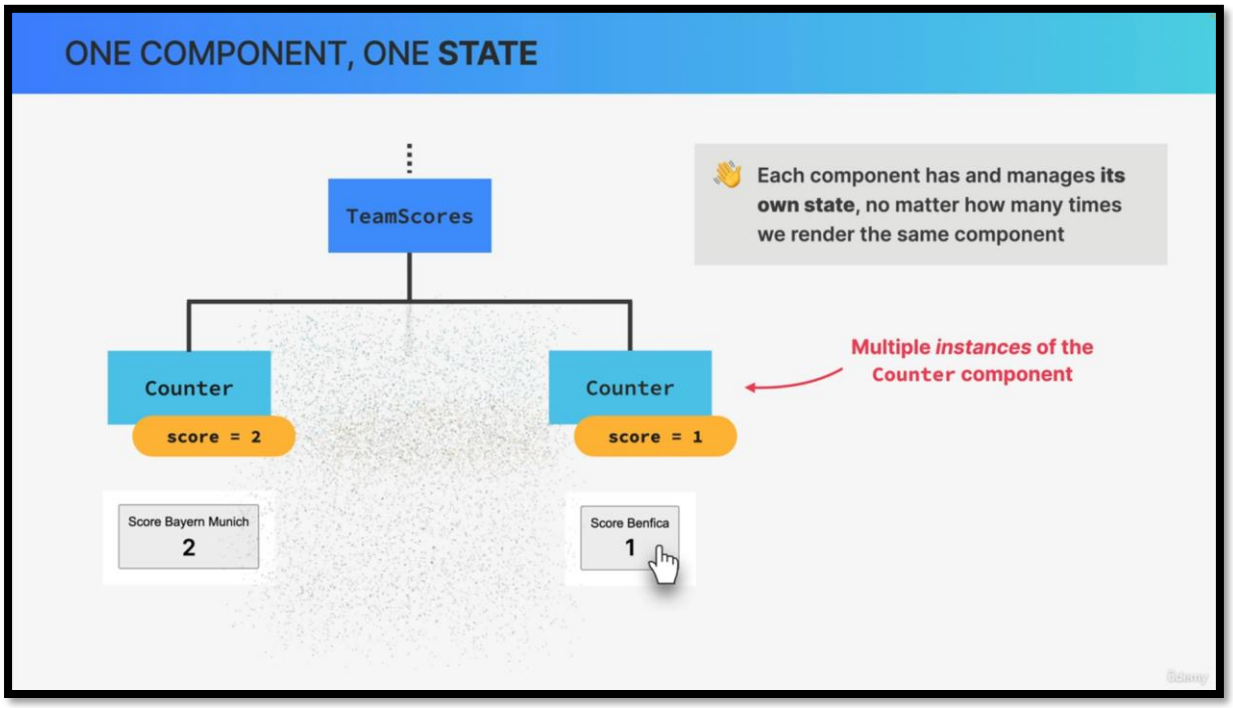
Event Handlers
Event handlers like increment in our example are defined within each component instance. When you click the button in one Counter, it calls the increment function of that instance and updates its state. It doesn't affect other instances.

Rendering Independence
Each component instance renders independently. Updates to one component don't trigger re-renders of the others unless there's a shared parent component or global state management involved.

In summary, React components are encapsulated and maintain their own state and props, making them operate independently when rendered multiple times on a page. This encapsulation and isolation are essential principles of React that contribute to the predictability and maintainability of your UI components.



Each component has and manages its own state.



So, even if we render the same component multiple times on one page, each of these component instances will operate independently from all the other ones.

So, in this example, the three counter components all start with a piece of state called "Score," which is set initially to zero.

If one of the buttons is clicked, that increases the score by one for each click but only in that component. The state in all the other components stays the same.

So, state really is isolated inside of each component.