✅ How It Works

1. **State Initialization**:
   - `this.state = { count: 0 }` initializes the state.
2. **State Modification**:
   - `this.setState()` updates `count` whenever a button is clicked.
   - `prevState.count + 1` ensures the latest state is used (to avoid async state issues).
3. **Component Lifecycle (`componentDidMount`)**:
   - Runs when the component mounts, useful for API calls, logs, or subscriptions.
4. **Reactivity**:
   - Clicking a button updates the state, **triggering a re-render** with the new `count` value.

✅ Why Use State in Class Components?

Feature	Benefit
Encapsulated Data	State is local to the component, preventing unintended modifications.
Dynamic UI Updates	Changes in state automatically trigger UI re-renders.
Complex State Handling	Useful for multi-step forms, interactive UIs, etc.
Works with Lifecycle Methods	Can be combined with `componentDidMount()`, `componentDidUpdate()`, etc.

🚀 Real-World Use Cases for Class-Based State

- **Legacy React applications** that predate Hooks
- **Dashboards** where widgets update dynamically
- **Forms with multiple steps** that track user input
- **Interactive UI components** like counters, sliders, and toggles