Skills Network

Introduction to States

IBM

What you will learn

Explain the use of states in class components

Explain the use of props in class components

Compare and contrast states and props

What is state?

Allows you to change data

An object used to track data

Built-in state objects

State change re-renders the component

Types of React state

Local state:

- Present only in a single component that needs it, such as hiding and showing information

Shared state:

- Shared by multiple components, such as an order application

States in React components

Represents information about the component's current situation

Determines how the component renders and behaves

Allows you to create dynamic and interactive components

States in class components

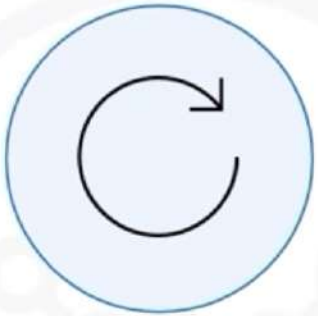
An instance with properties that control the behavior of the component

Managed and preserved in the React component

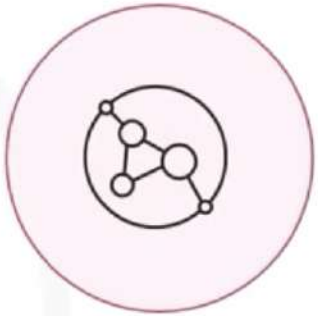
Holds some information that may change over the lifetime of the component

When to use states

Required when a component changes during a user interaction



Includes forms, buttons, and timers

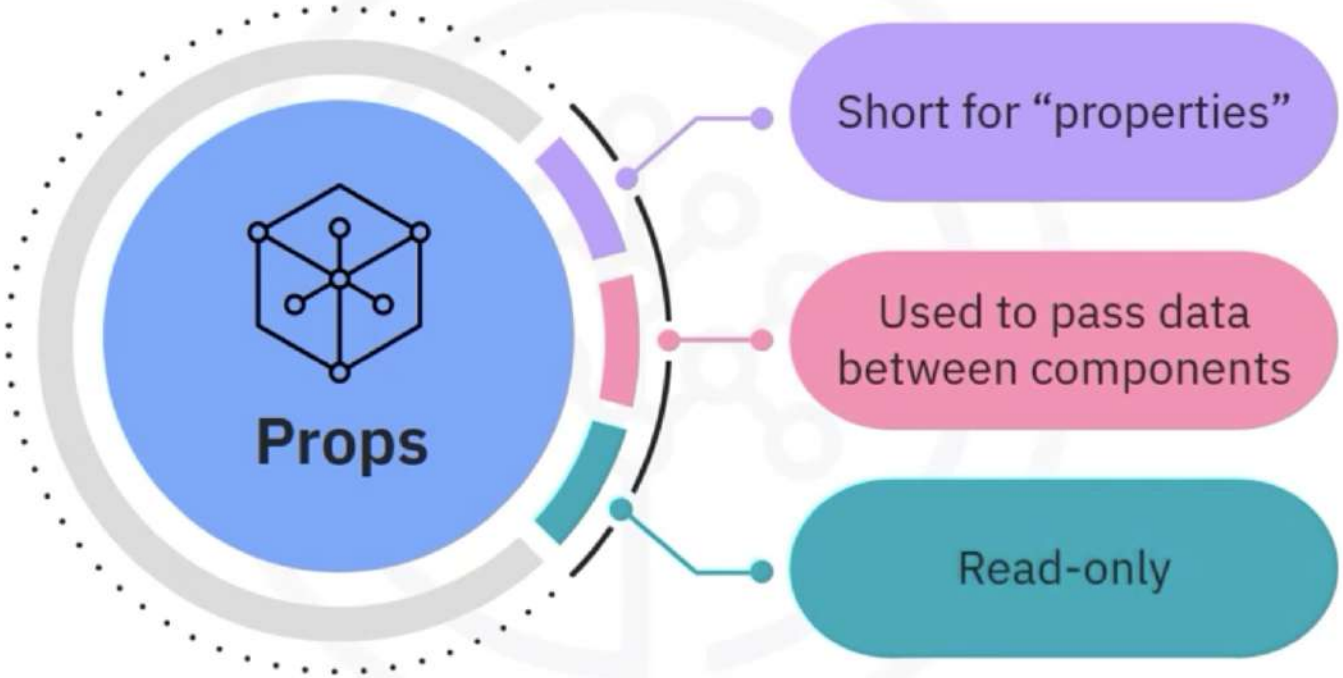


Example: Incrementing a counter when user presses a button

State example

```
class TestComponent extends React.Component {
  constructor(props) {
    super(props);
      this.state = {
        id: 1, name: "John", age: 28
      };
    }
    render() {
      return (
        <div>
          <p>{this.state.name}</p> <p>{this.state.age}</p>
         </div>
      );
    }
  }
  const root = ReactDOM.createRoot(document.getElementById('root'));
  root.render(<TestComponent />)
```

Props


Props
Short for “properties”
Used to pass data between components
Read-only

Props behavior

Store attribute values and work like the HTML attributes

Arguments that you pass between components

Immutable

Modify a variable in a components state

Allows child components access to methods defined in the parent

Props example

```
class TestComponent extends React.component {
  render() {
    return <div> Hi {this.props.name} </div>
  }

//passing the props as examples to the test component
<TestComponent name='John'/>
<TestComponent name='Jill'/>
```

State versus props

State	Props
Cannot access or modify data from outside	Receives data from parent in form of props
Components create and manage their own state data	Receives data from outside using props
Used to manage data	Used to pass data
Can modify its own data from within the component	Read-only data
Modify with `setState()` method	unidirectional flow from parent to child

Recap

In this video, you learned that:

- A state represents information about the component's current situation
- The state allows you to create interactive components
- Changes in a state cause the component to re-render
- Use props to pass data between components in a unidirectional flow
- Components create and manage their own data with their state, whereas they receive data from outside using props