Skills Network

Passing Data and States Between Components

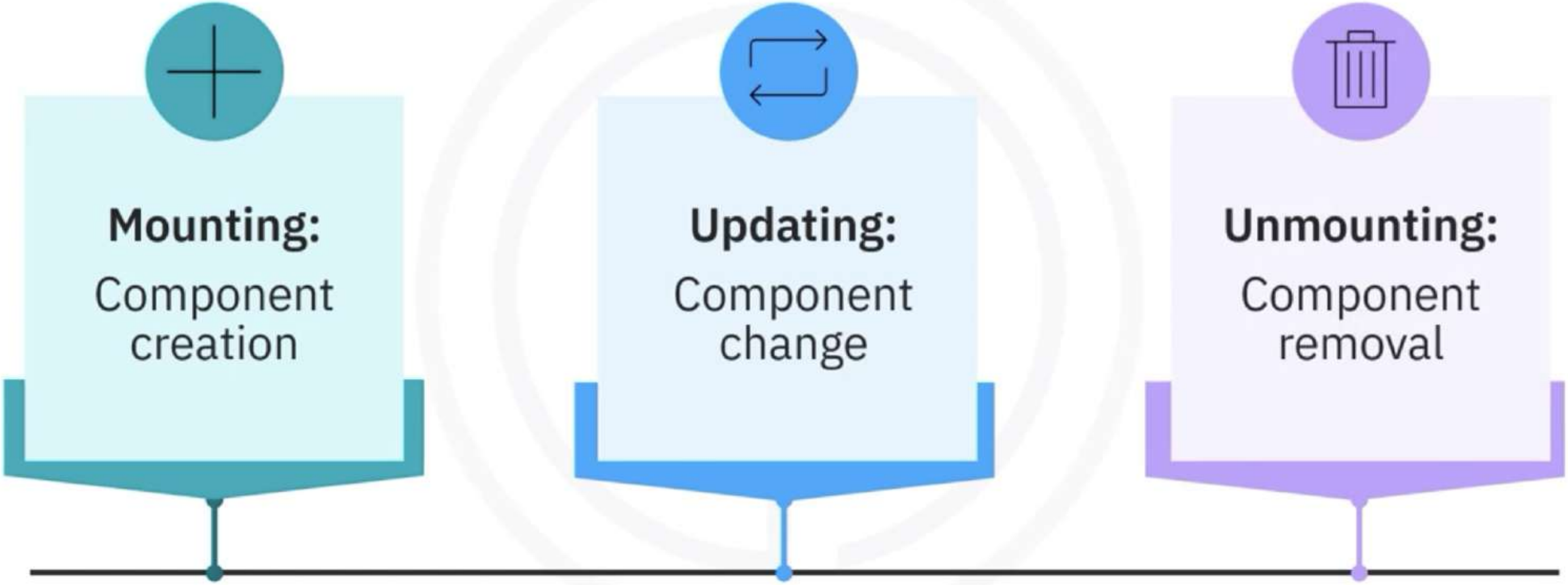
IBM

What you will learn

Describe the lifecycle of React components

Explain how to pass data and states to components

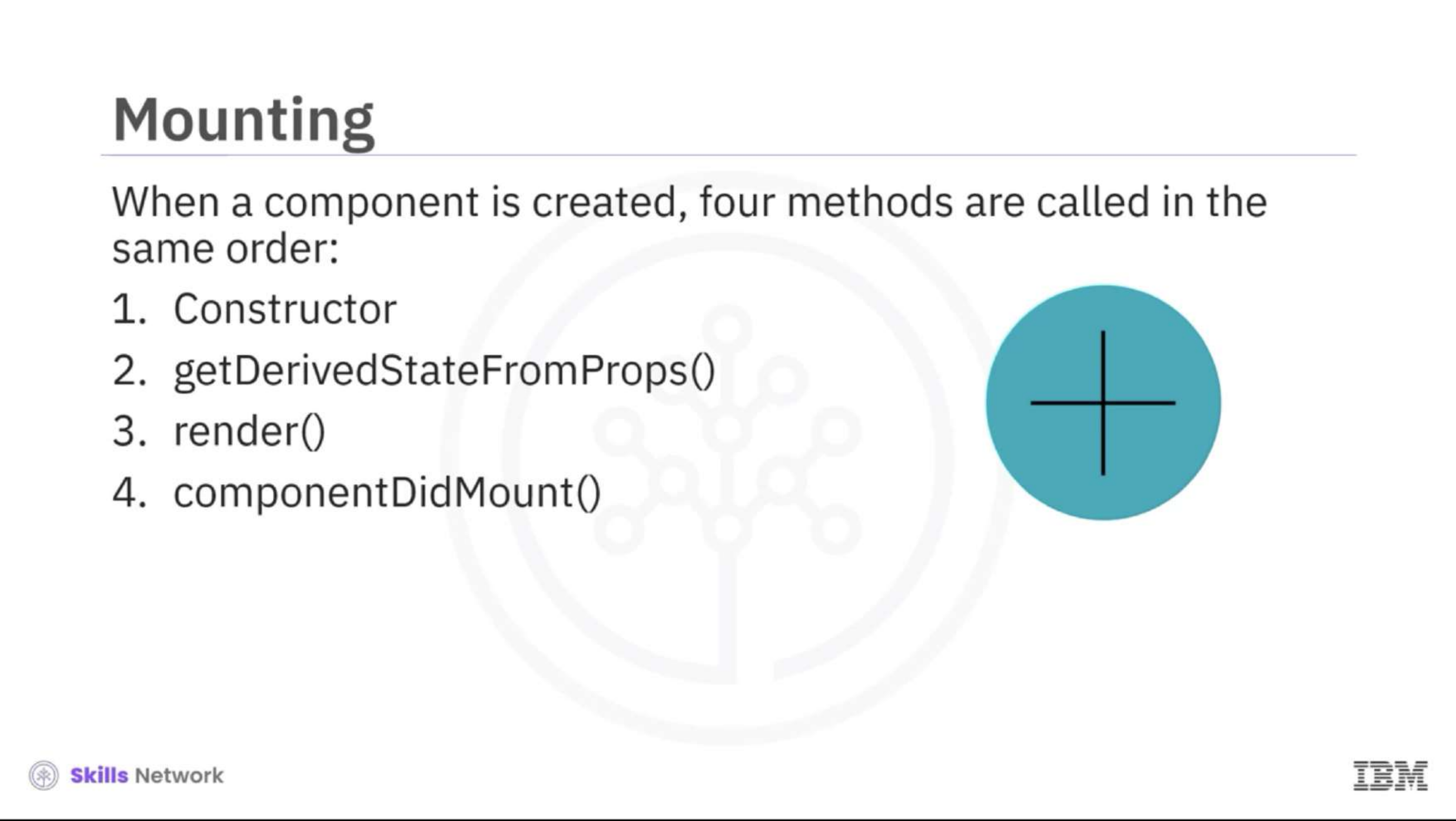
Component phases



Mounting:
Component creation

Updating:
Component change

Unmounting:
Component removal



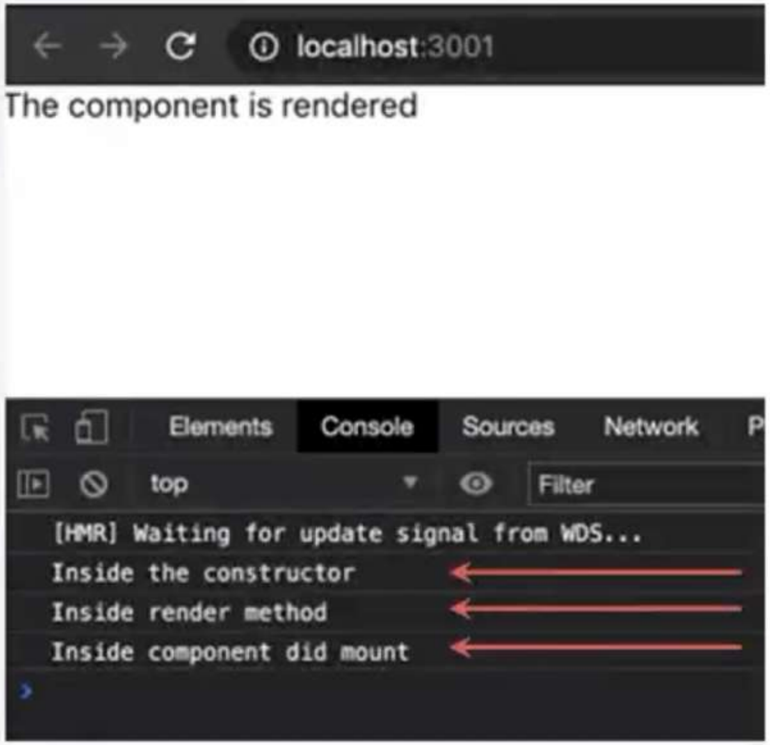
Mounting

When a component is created, four methods are called in the same order:

1. Constructor
2. getDerivedStateFromProps()
3. render()
4. componentDidMount()

Mounting

```
import React from 'react';
class App extends React.Component{
  constructor(props){
    super(props)
      console.log("Inside the constructor")
  }
  componentDidMount = ()=>{
    console.log("Inside component did mount")
  }
  render(){
    console.log("Inside render method")
    return(
      <div> The component is rendered </div>
    )
  }
}
export default App;
```



localhost:3001

The component is rendered

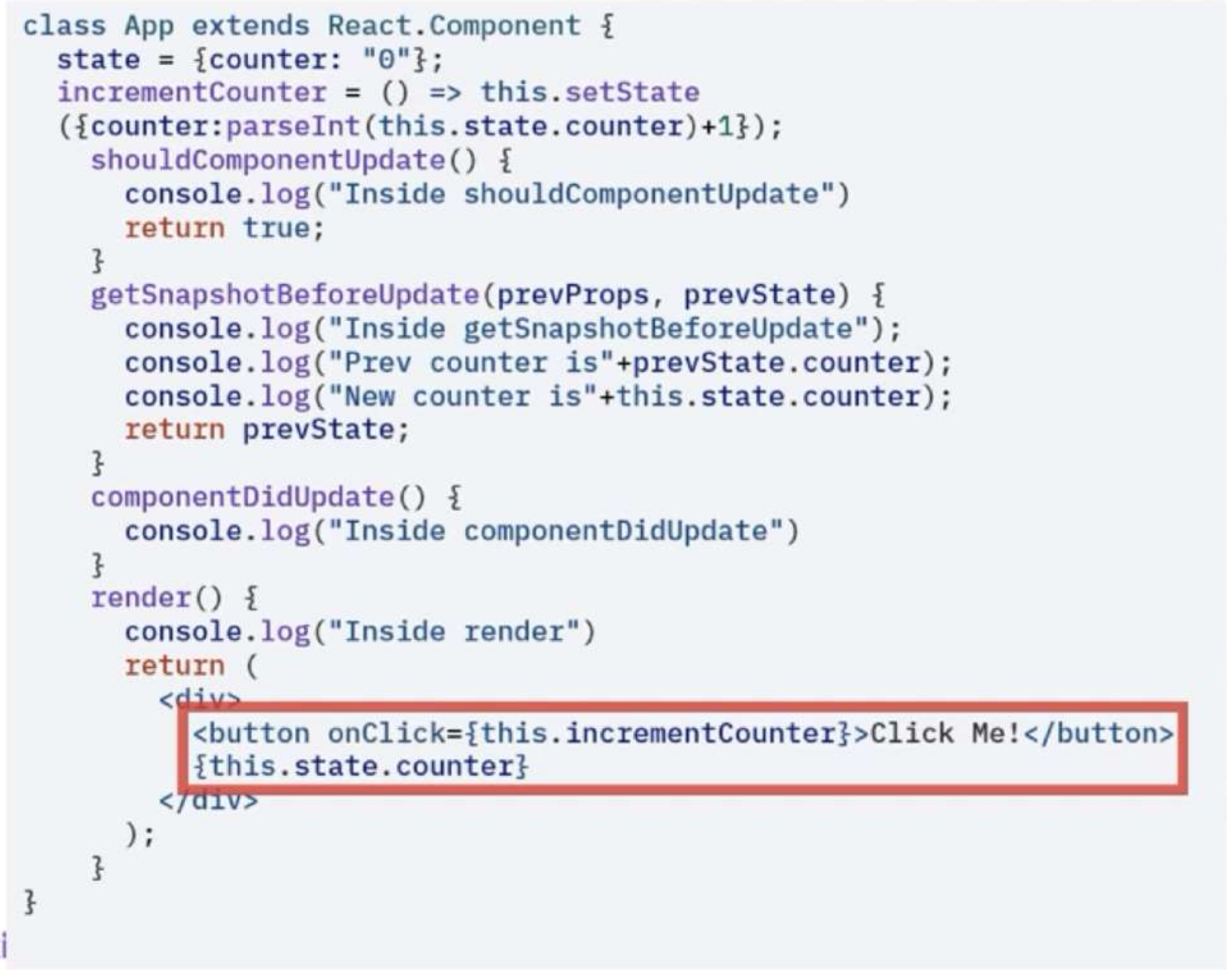
```
[HMR] Waiting for update signal from WDS...
Inside the constructor
Inside render method
Inside component did mount
```

Updating

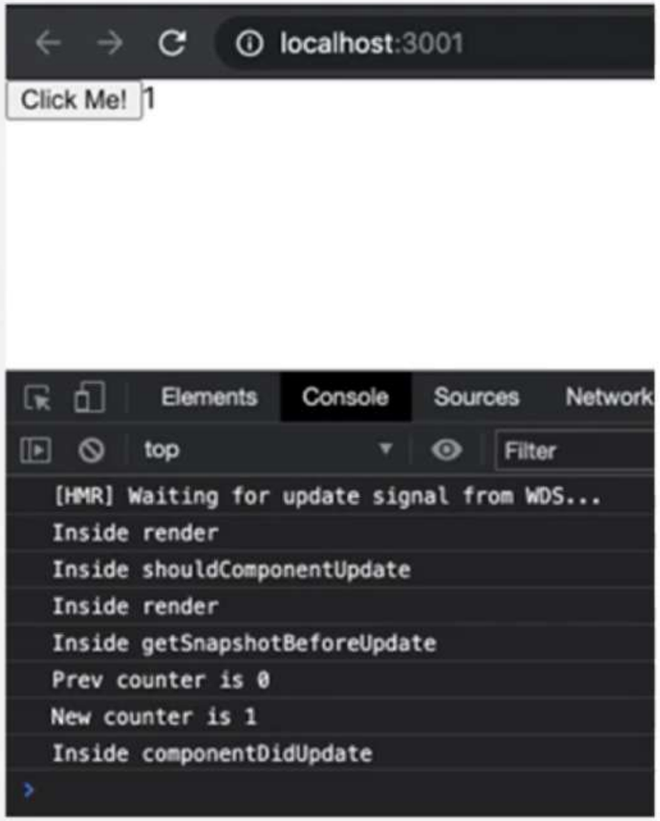
When a component is created, five methods are called in the same order:

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

Updating

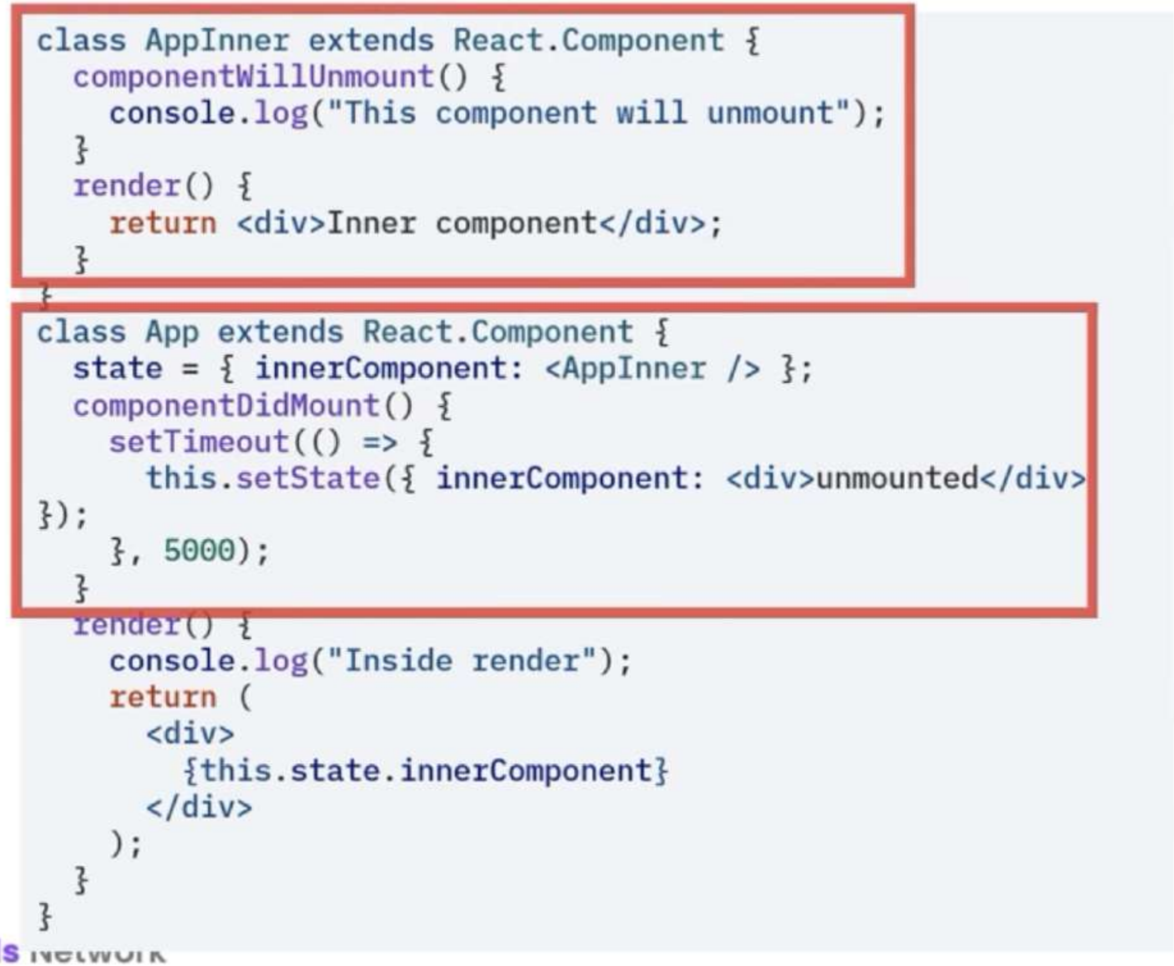


```
class App extends React.Component {
  state = {counter: "0"};
  incrementCounter = () => this.setState
  ({counter:parseInt(this.state.counter)+1});
    shouldComponentUpdate() {
      console.log("Inside shouldComponentUpdate")
      return true;
    }
    getSnapshotBeforeUpdate(prevProps, prevState) {
      console.log("Inside getSnapshotBeforeUpdate");
      console.log("Prev counter is"+prevState.counter);
      console.log("New counter is"+this.state.counter);
      return prevState;
    }
    componentDidUpdate() {
      console.log("Inside componentDidUpdate")
    }
    render() {
      console.log("Inside render")
      return (
        <div>
          <button onClick={this.incrementCounter}>Click Me!</button>
          {this.state.counter}
        </div>
      );
    }
}
```



```
[HMR] Waiting for update signal from WDS...
Inside render
Inside shouldComponentUpdate
Inside render
Inside getSnapshotBeforeUpdate
Prev counter is 0
New counter is 1
Inside componentDidUpdate
```

Unmounting

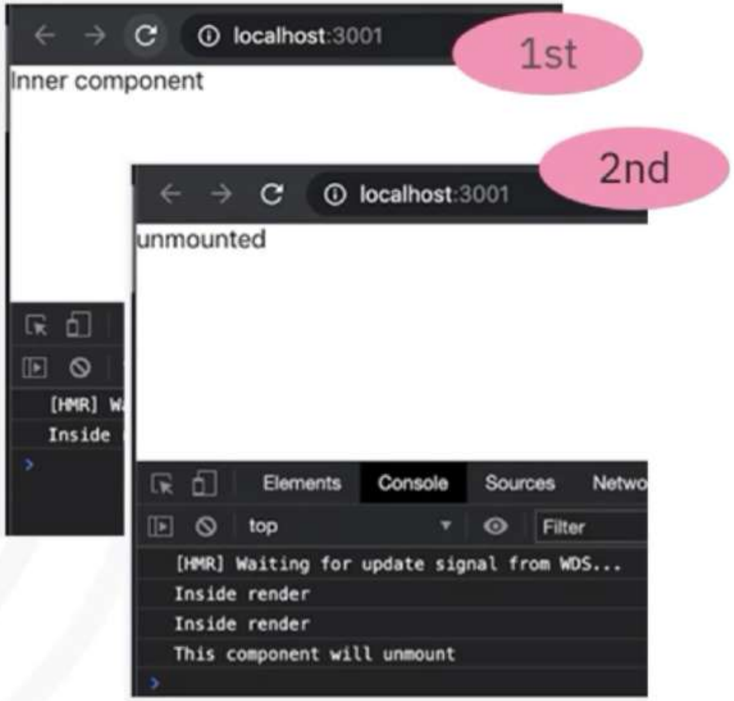


```
class AppInner extends React.Component {
  componentWillUnmount() {
    console.log("This component will unmount");
  }
  render() {
    return <div>Inner component</div>;
  }
}
class App extends React.Component {
  state = { innerComponent: <AppInner /> };
  componentDidMount() {
    setTimeout(() => {
      this.setState({ innerComponent: <div>unmounted</div>
});
    }, 5000);
  }
  render() {
    console.log("Inside render");
    return (
      <div>
        {this.state.innerComponent}
      </div>
    );
  }
}
```

When a component is unmounted, the componentWillUnmount() method is called

Unmounting

```
class AppInner extends React.Component {
  componentWillUnmount() {
    console.log("This component will unmount");
  }
  render() {
    return <div>Inner component</div>;
  }
}
class App extends React.Component {
  state = { innerComponent: <AppInner /> };
  componentDidMount() {
    setTimeout(() => {
      this.setState({ innerComponent: <div>unmounted</div>
});
    }, 5000);
  }
  render() {
    console.log("Inside render");
    return (
      <div>
        {this.state.innerComponent}
      </div>
    );
  }
}
```



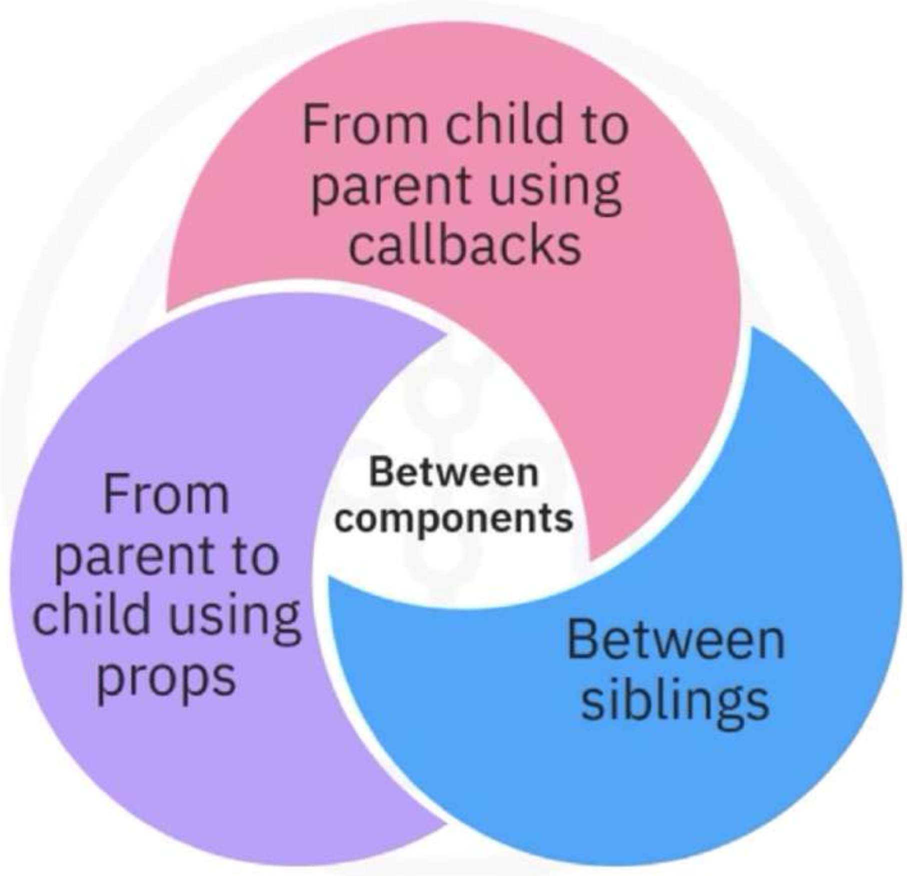
Inner component

```
[HMR] W
Inside
```

unmounted

```
[HMR] Waiting for update signal from WDS...
Inside render
Inside render
This component will unmount
```

Passing data between components



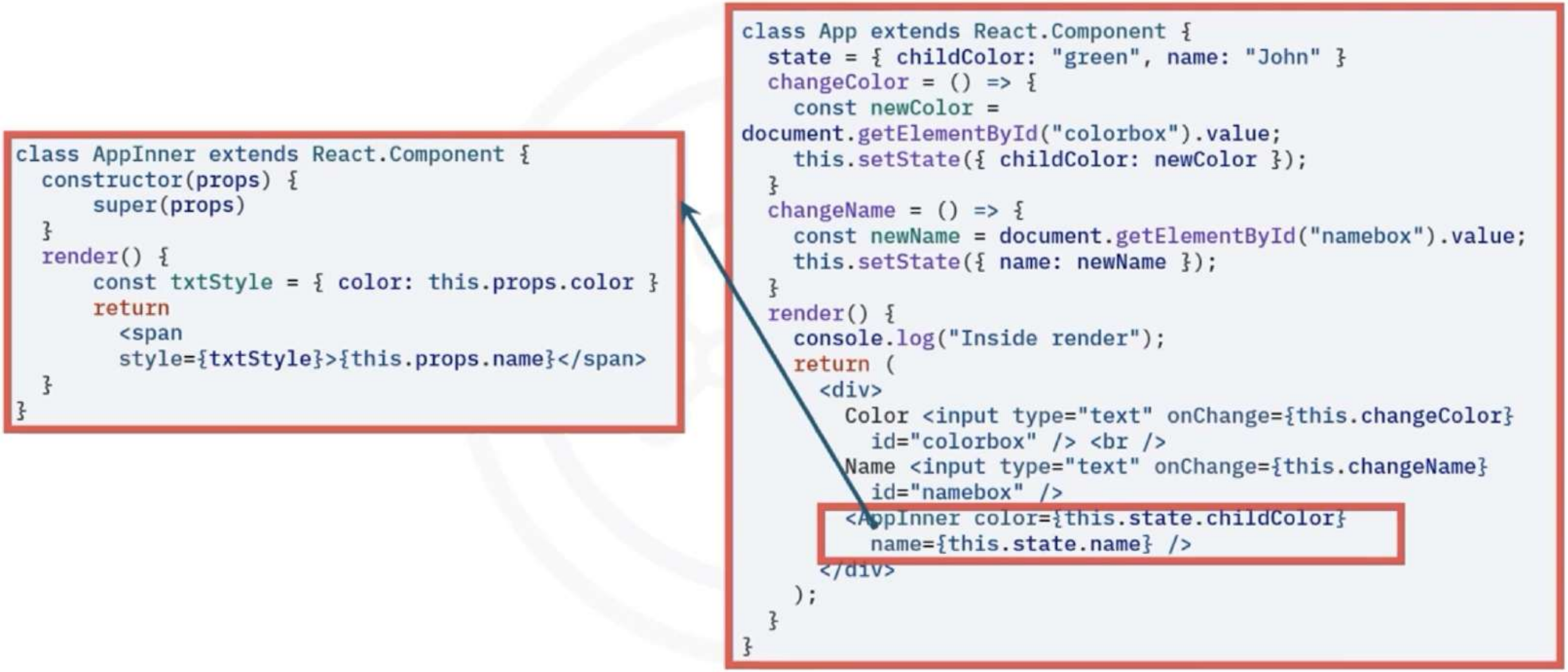
From child to parent using callbacks

From parent to child using props

Between components

Between siblings

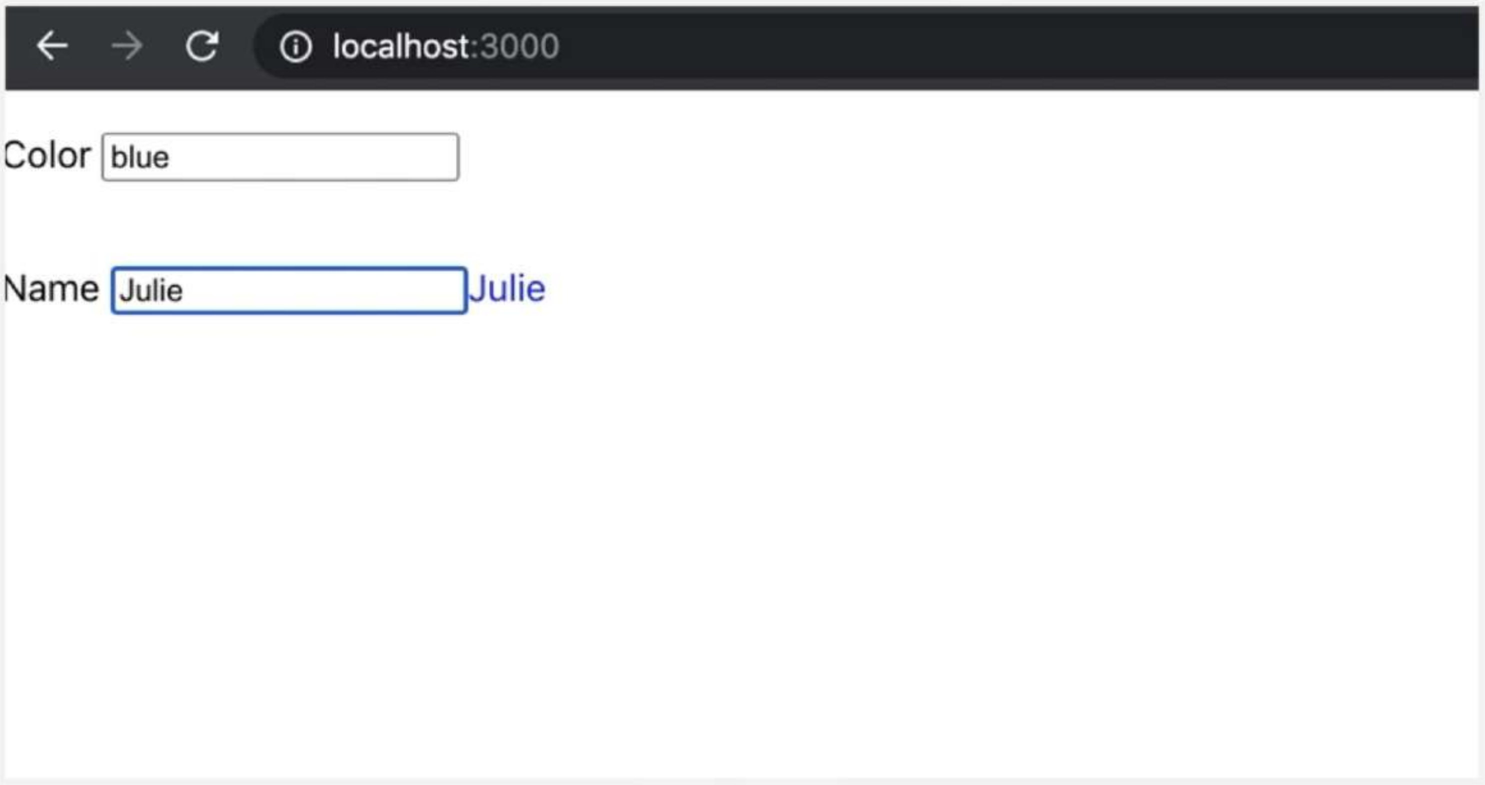
Passing data from parent to child



```
class AppInner extends React.Component {
  constructor(props) {
      super(props)
  }
  render() {
      const txtStyle = { color: this.props.color }
      return
        <span
        style={txtStyle}>{this.props.name}</span>
  }
}
```

```
class App extends React.Component {
  state = { childColor: "green", name: "John" }
  changeColor = () => {
    const newColor =
document.getElementById("colorbox").value;
    this.setState({ childColor: newColor });
  }
  changeName = () => {
    const newName = document.getElementById("namebox").value;
    this.setState({ name: newName });
  }
  render() {
    console.log("Inside render");
    return (
      <div>
        Color <input type="text" onChange={this.changeColor}
          id="colorbox" /> <br />
        Name <input type="text" onChange={this.changeName}
          id="namebox" />
        <AppInner color={this.state.childColor}
          name={this.state.name} />
      </div>
    );
  }
}
```

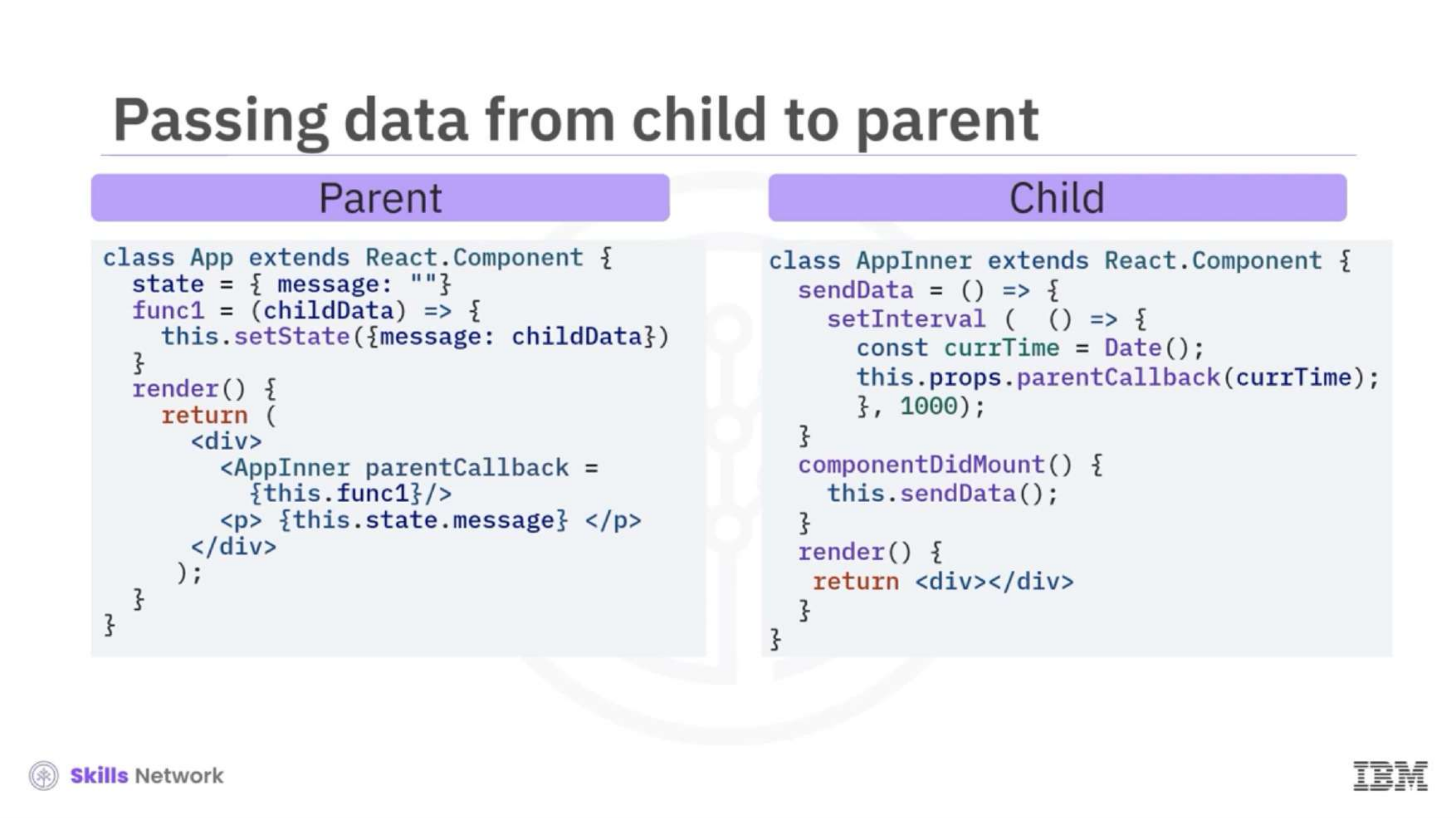
Passing data from parent to child



localhost:3000

Color: blue

Name: Julie Julie



Passing data from child to parent

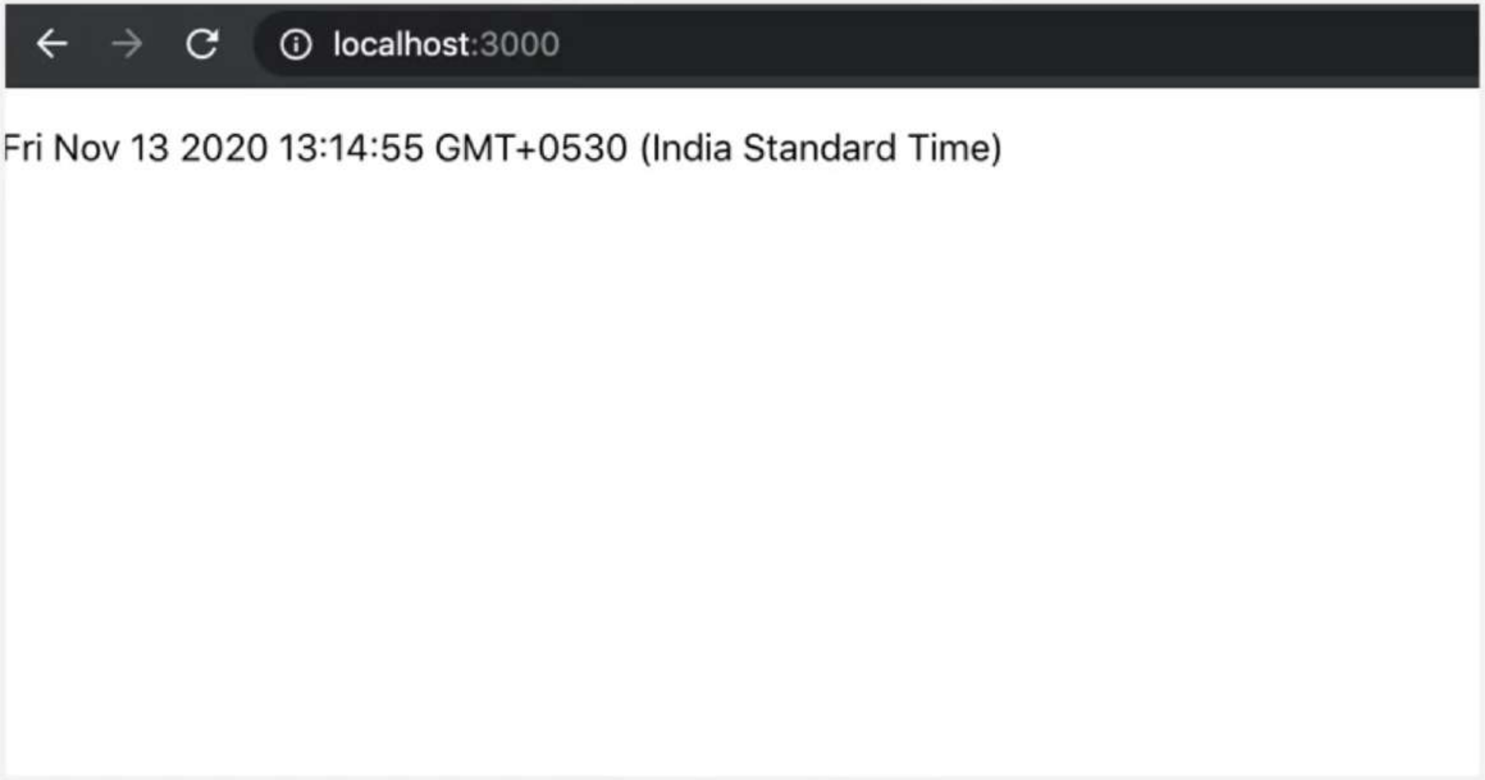
Parent

```
class App extends React.Component {
  state = { message: ""}
  func1 = (childData) => {
    this.setState({message: childData})
  }
  render() {
    return (
      <div>
        <AppInner parentCallback =
          {this.func1}/>
        <p> {this.state.message} </p>
      </div>
     );
  }
}
```

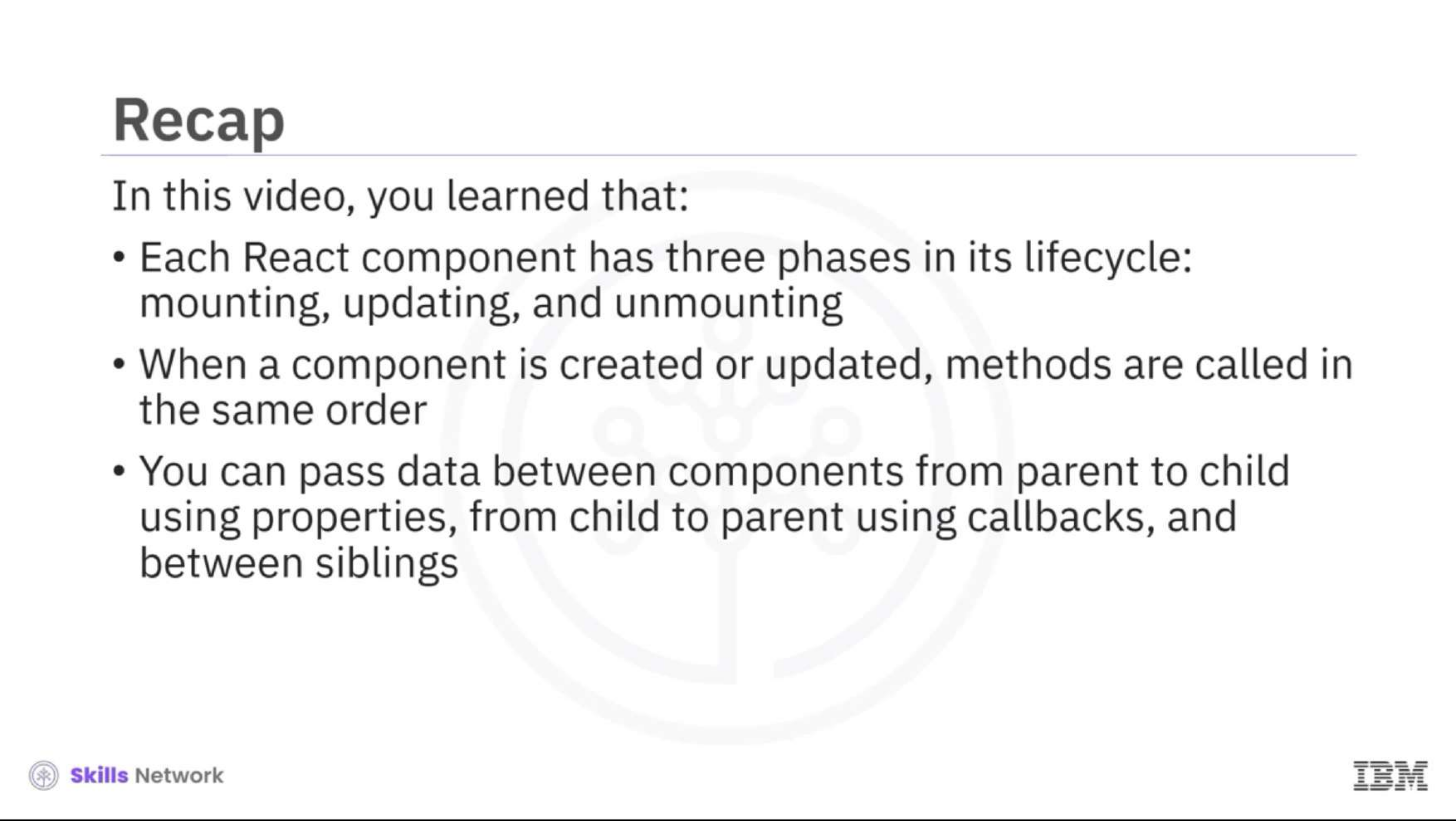
Child

```
class AppInner extends React.Component {
  sendData = () => {
    setInterval (  () => {
      const currTime = Date();
      this.props.parentCallback(currTime);
      }, 1000);
  }
  componentDidMount() {
    this.sendData();
  }
  render() {
   return <div></div>
  }
}
```

Passing data from child to parent



```
Fri Nov 13 2020 13:14:55 GMT+0530 (India Standard Time)
```



Recap

In this video, you learned that:

- Each React component has three phases in its lifecycle: mounting, updating, and unmounting
- When a component is created or updated, methods are called in the same order
- You can pass data between components from parent to child using properties, from child to parent using callbacks, and between siblings