Props, State, Lifecycle and Keys

Cory House
PRINCIPAL CONSULTANT
@housecor reactjsconsulting.com

Agenda

Props

State

Lifecycle methods

Keys

Plenty of coding!

Props and State

Props

- Data passed from parent component
- Look like HTML attributes

Props

```
function Avatar(props) {
  return <img src={"images/" + props.username} />;
}
```

```
<Avatar username="cory" />
```

```
<Avatar username={username} />
```

Props and State

Props

- Data passed from parent component
- Look like HTML attributes
- Immutable
  - Want to change? Call a function provided by the parent.

Props and State

State

- Holds *mutable* state
- Set via setState in class components

```
state.username
```

State

```
class Example extends Component {
  constructor(props) {
    super(props);
    this.state = { name: "" };
  }

  onChange(event) {
    this.setState({ name: event.target.value });
  }

  render() {
    return <input onChange={this.onChange}                         value={this.state.name} />;
  }
}
```

State

```
class Example extends Component {
  state = { name: "" };

  onChange(event) {
    this.setState({ name: event.target.value });
  }

  render() {
    return <input onChange={this.onChange}                    value={this.state.name} />;
  }
}
```

Lifecycle Methods (Class Components)

constructor

render

static getDerivedStateFromProps

componentDidMount

shouldComponentUpdate

componentDidUpdate

componentWillUnmount

componentDidCatch

constructor

When

Before the component is mounted

Why

Initialize state, bind event methods

render

When

Anytime state and props change

Why

To declare the markup your component outputs

componentDidMount

When

After component is mounted

Why

Access DOM, set up subscriptions, integrate with frameworks, set timers, make HTTP calls

shouldComponentUpdate

When

Before render when new props or state are being received.

Why

Performance. Return false to void unnecessary re-renders.

componentDidUpdate

When

After component's updates are flushed to the DOM.

Why

Work with the DOM after an update

componentWillUnmount

When

Immediately before component is removed from the DOM

Why

Cleanup

componentDidCatch

When

Anytime an error occurs in your component

Why

Catch errors and handle them as desired

Demo

Demo: State and lifecycle methods

Keys for Dynamic Children

Add a key to dynamic child elements

```
function Courses() {
  return courses.map(course => {
    return <div key={course.id}>{course.title}</div>;
  });
}
```

Demo

Keys

Summary

Props

- Pass data to child components

State

- Mutable data

Lifecycle Methods

- Run code at different times in classes

Keys

Next up: Hooks, component composition, and PropTypes