Advanced React.js

Recap

- We learned about the anatomy of a React.js directory structures
  - And “scaffolding” a simple React.js app
- How to deploy a React app
- Using JSX and the advantages for UI development over regular JavaScript
- How to design a React app through Components
  - And passing data via props

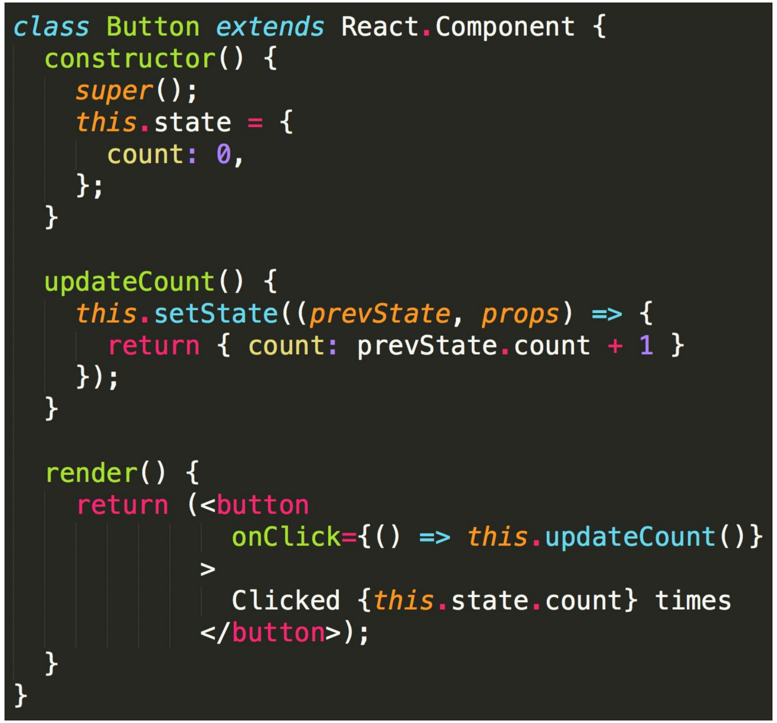
State

- State is very similar to props
- State is managed WITHIN the Component
- Whereas Props are passed to the Component (similar to function parameters)
- By default, a Component has no State but can easily be enabled

When to Use State?

- When a Component needs to keep track of information between renderings the Component itself can create update and use State
- State is very similar to an Instance Variable or a Class Variable in Object-Oriented Programming
- State will normally be “managed” in the Constructor

State Example



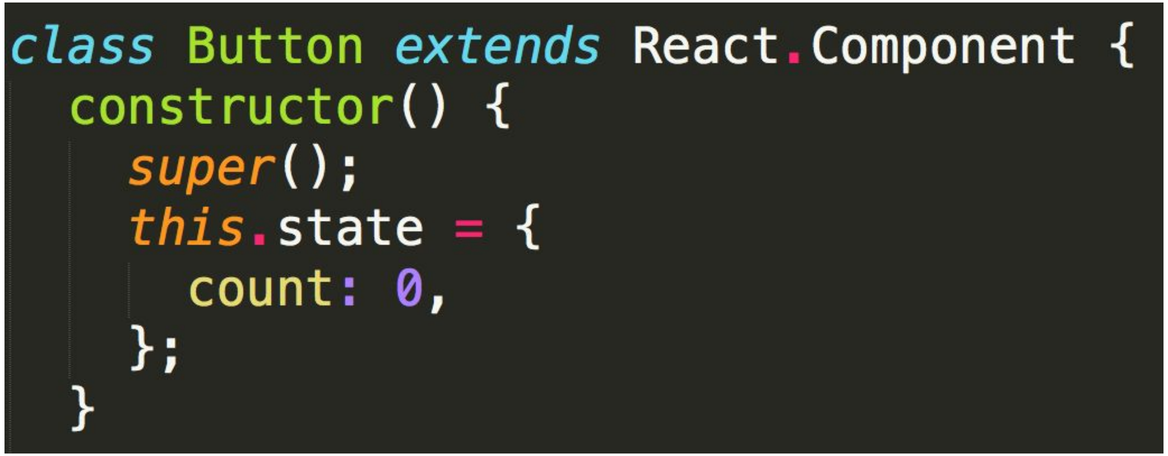
```
class Button extends React.Component {
  constructor() {
    super();
    this.state = {
      count: 0,
    };
  }

  updateCount() {
    this.setState((prevState, props) => {
      return { count: prevState.count + 1 }
    });
  }

  render() {
    return (<button
              onClick={() => this.updateCount()}
            >
              Clicked {this.state.count} times
            </button>);
  }
}
```

Button Component

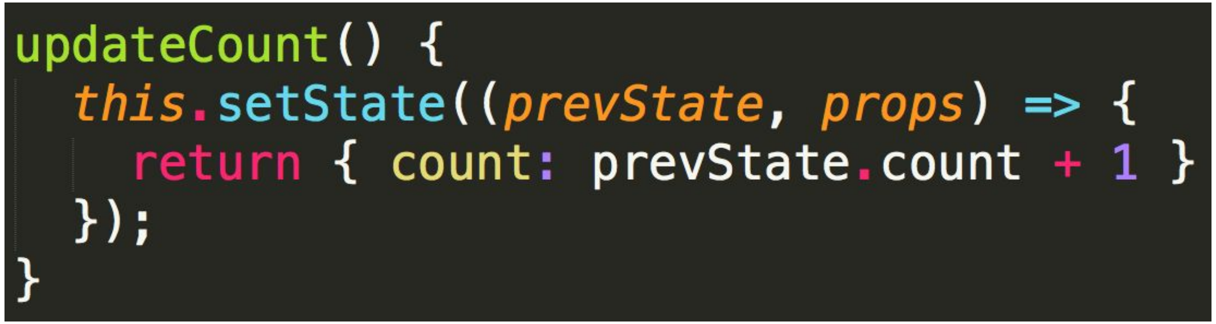
- Good to initialize State with a hardcoded value or a Prop value. Don't use null, undefined, and must have some initial value because it is essentially JSON (must have a value for a key)



```
class Button extends React.Component {
  constructor() {
    super();
    this.state = {
      count: 0,
    };
  }
```

State is Changeable!

- Below we are keeping track of total clicks
- setState takes a callback function with two parameters
  - prevState: Stores the previous value of State
  - Props: which can be ignored and is not needed



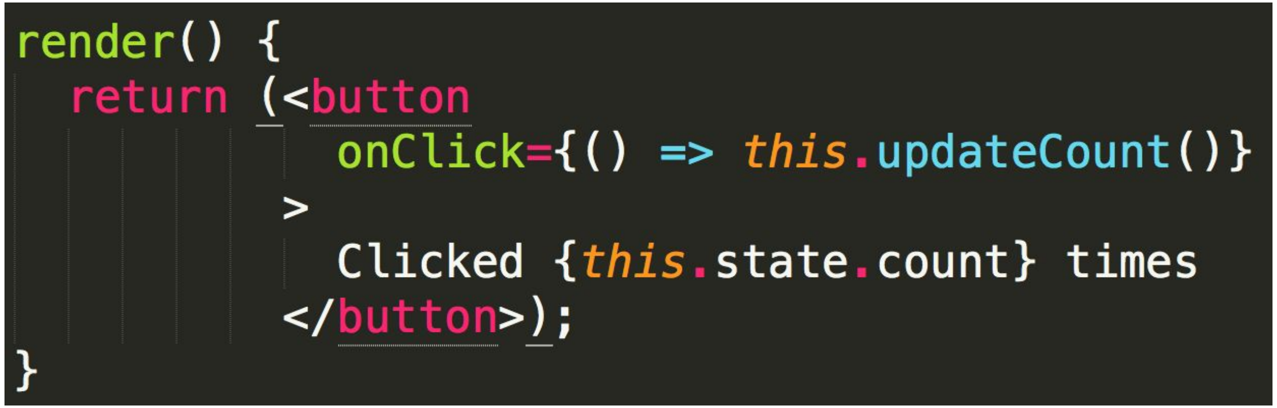
```
updateCount() {
  this.setState((prevState, props) => {
    return { count: prevState.count + 1 }
  });
}
```

DO NOT DO

- This.state.count = this.state.count + 1;
- Does not work because the Component does not re-render!

Click Event and Print

- When the button is clicked, the updateCount() function will be called
- The text is rendered in the button click and the web page will appear “Clicked x times”

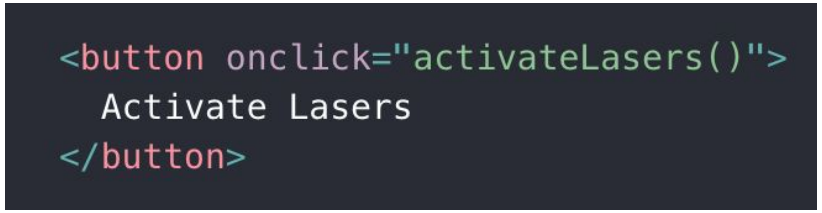


```
render() {
  return (<button
            onClick={() => this.updateCount()}
          >
            Clicked {this.state.count} times
          </button>);
}
```

Handling Events

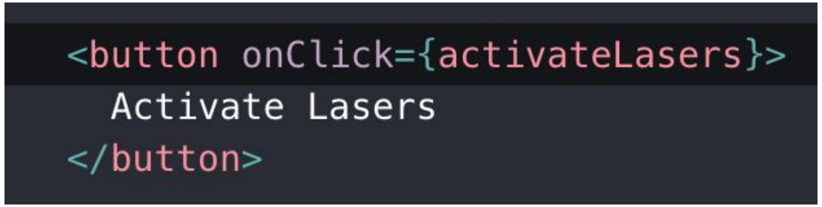
- From the React.js website: Handling events in React is very similar to handling events in HTML using the DOM

HTML



```
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

React



```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

Returning False on Event

- HTML

```html
<a href="#" onclick="console.log('The link was clicked.'); return false">
  Click me
</a>
```

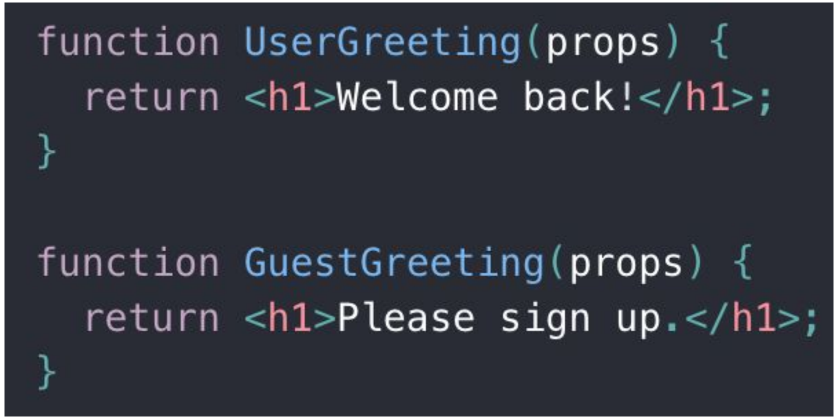
- React

```jsx
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('The link was clicked.');
  }

  return (
    <a href="#" onClick={handleClick}>
      Click me
    </a>
  );
}
```

Conditional Rendering

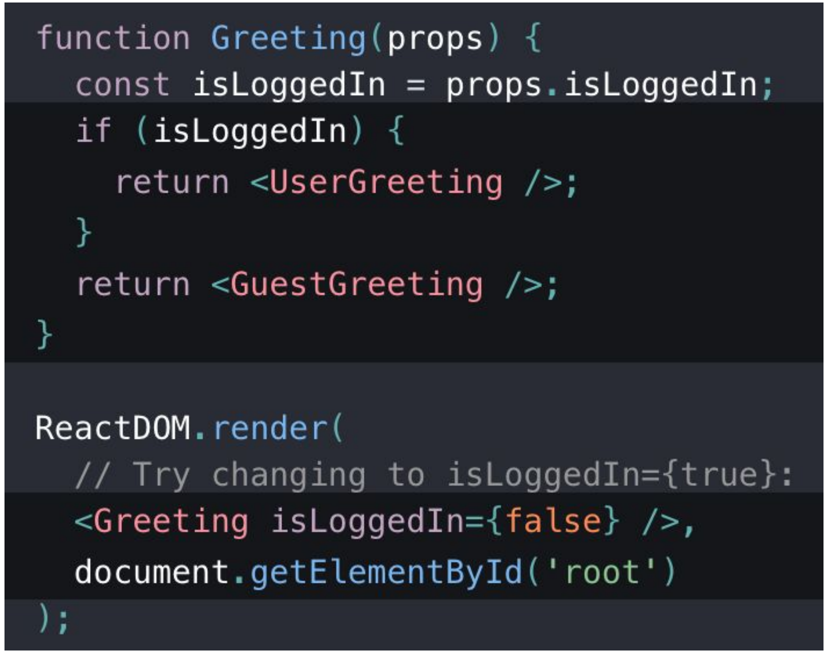
- Conditional Rendering in React works exactly the same way that conditionals work in JavaScript
- Based on some condition, certain components will get rendered



```
function UserGreeting(props) {
  return <h1>Welcome back!</h1>;
}

function GuestGreeting(props) {
  return <h1>Please sign up.</h1>;
}
```

Conditional Rendering (Cont.)



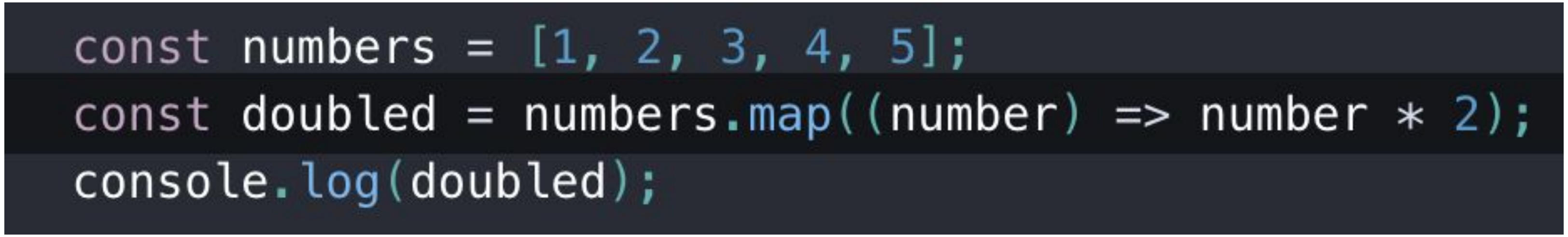
```
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}

ReactDOM.render(
  // Try changing to isLoggedIn={true}:
  <Greeting isLoggedIn={false} />,
  document.getElementById('root')
);
```

Rendering Multiple Components

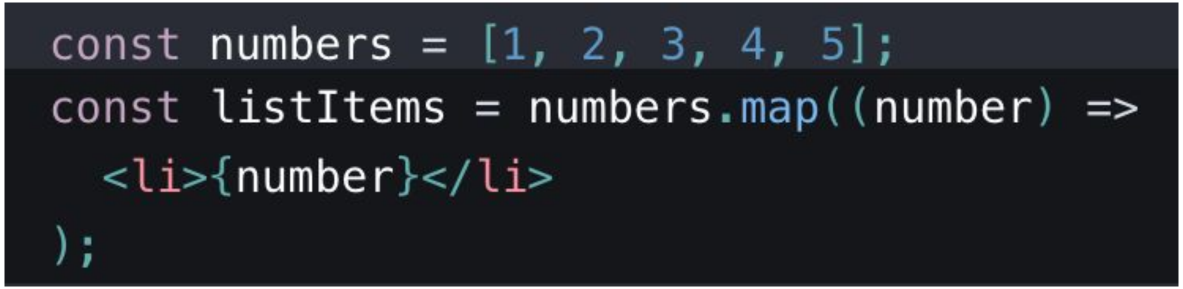
- Rendering multiple components is useful especially thinking about HTML elements
- Take a list for example:
    - Instead of hardcoding a bunch of <li> elements that are static

Given The Following...

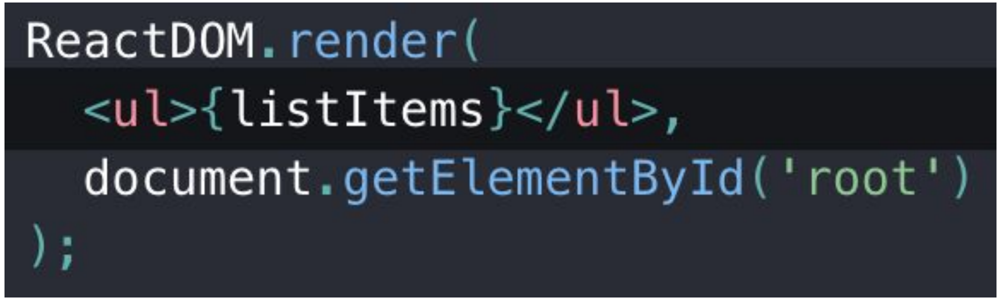


```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map((number) => number * 2);
console.log(doubled);
```

Render Multiple List Components



```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li>{number}</li>
);
```



```
ReactDOM.render(
  <ul>{listItems}</ul>,
  document.getElementById('root')
);
```