Xtern

BOOTCAMP

Week 2 | Day 1

May 22, 2017

Intro to React

What is React?

A *declarative*, *efficient*, and *flexible* JavaScript library for building user interfaces. It encourages the creation of *reusable UI components* which present data that changes over time.

Declarative vs Imperative

Imperative - describe *how* to do something

Declarative - describe *what* you want to do

Declarative vs Imperative

Example One - Arriving at a restaurant

Example Two - Driving a car

Declarative vs Imperative

```
function multiplyByThree(originalArray) {
  let multipliedArray = []

  for (let i = 0; i < originalArray.length; i++) {
    let multipliedNumber = originalArray[i] * 3
    multipliedArray.push(multipliedNumber)
  }

  return multipliedArray
}
```

Declarative vs Imperative

```
function multiplyByThree(originalArray) {
  return originalArray.map(number => number * 3)
}
```

What is React?

A *declarative*, *efficient*, and *flexible* JavaScript library for building user interfaces. It encourages the creation of *reusable UI components* which present data that changes over time.

Efficiency

DOM manipulation and rendering is slow

Because React represents the DOM through JavaScript Objects (Virtual DOM), it can compare the current state with the new state when things change so that it only re-renders what has changed.

This process of determining what has changed is called *diffing*, and the process of making the necessary changes is *reconciliation*.

4 Things Needed for a React App

1. An HTML page with at least one empty element
2. The React library
3. One or more React Components
4. A JavaScript call to attach the React component to the empty HTML element from step 1

Components

Some typical HTML

```
<header>
  <h1>My Site is the best</h1>
  <h2>Welcome to my site, Steve</h2>
  <ul class="nav-links">
    <li><a href="home">Home</a></li>
    <li><a href="about">About</a></li>
    <li><a href="archives">Archives</a></li>
    <li><a href="contact">Contact</a></li>
  </ul>
</header>
```

Wouldn’t it be cool…

```
<AppHeader name=”Steve" />
```

As a React Component

```
class AppHeader extends React.Component {
  render() {
    return (
      <header>
        <h1>My Site is the best</h1>
        <h2>Welcome to my site, {this.props.name}</h2>
        <ul className="nav-links">
          <li><a href="home">Home</a></li>
          <li><a href="about">About</a></li>
          <li><a href="archives">Archives</a></li>
          <li><a href="contact">Contact</a></li>
        </ul>
      </header>
    )
  }
}
```

JSX - JavaScript XML

```
class AppHeader extends React.Component {
  render() {
    return (
      <header>
        <h1>My Site is the best</h1>
        <h2>Welcome to my site, {this.props.name}</h2>
        <ul className="nav-links">
          <li><a href="home">Home</a></li>
          <li><a href="about">About</a></li>
          <li><a href="archives">Archives</a></li>
          <li><a href="contact">Contact</a></li>
        </ul>
      </header>
    )
  }
}
```

Without JSX

```
React.createElement('header', null,
  React.createElement('h1', null, 'My Site is the best'),
  React.createElement('h2', null, 'Welcome to my site’, props.name),
  React.createElement('ul', null,
    React.createElement('li', null,
      React.createElement('a', {href: 'home'}, 'Home'),
      // etc, etc...
    )
  )
)
```

Props

Component Props

Conceptually, a React component can be thought of as a JavaScript function. They accept arbitrary inputs called ‘props’, and return React elements describing what should appear on the screen

Component Props

Like function arguments, props can be just about anything:

- Strings
- Numbers
- Functions
- Objects
- Booleans
- Arrays
- Symbols

Component Props

```
function alertMe() {
  alert('ALERT!!!')
}

class MyComponent extends React.Component {
  render() {
    return (
      <div>
        <button onClick={() => this.props.alertMe()}>ALERT ME</button>
        <p>My name is {this.props.name}</p>
        <p>My favorite boolean is: {this.props.myFav}</p>
      </div>
    )
  }
}

<MyComponent alertMe={alertMe} myFav={true} name="Emilio" />
```

Component Props

Props are received from outside the component, and cannot be changed from within the component. They are *immutable*.

State

Component State

If a component needs to keep track of its own data and also be able to modify that data, we use *state.*

State is initialized in the component's constructor and is modified using the 'setState' method.

Component State

```
class ClickyCounter extends React.Component {
  constructor() {
    this.state = { count: 0 }
  }

  updateCount() {
    this.setState({ count: this.state.count + 1 })
  }

  render() {
    return (
      <div>
        <button onClick={() => this.updateCount()}>Click to count</button>
        <p>Button has been clicked {this.state.count} times</p>
      </div>
    )
  }
}
```

Attaching to the DOM

Attaching to the DOM

```
<!doctype html>
<html lang="en">
  <head>
    <title>React App</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>

class App extends Component {
  render() {
    return (<div className=“App”>React App</div>);
  }
}

ReactDOM.render(<App />, document.getElementById('root'));
```