XTERN

BOOTCAMP

Week 2 : Day 1

Introducing

React

What is React?

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

Declarative vs. Imperative

Imperative: Describe how to do something

Declarative: Describe what you want to do

EXAMPLE 1

ARRIVING AT A RESTAURANT

Example 2
Driving a car

Example 3

Making a dang web page

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

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

  return multipliedArray
}
```

```
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 that present data that changes over time.

EFFICIENCY
THE VIRTUAL DOM

Reusable components

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="navLinks">
    <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

```
<h1>My site is the best</h1>
```

becomes

```
React.createElement(
   'h1',
   null,
   'My site is the best'
)
```

```
<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>
```

```
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

```
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 and modify its own data, we use state.

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

```
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

```
<!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'));
```