Your first React Component

Hooray! If you've made it this far, you're ready to do some serious coding in React.

React components and ES6

We're going to be writing all of our React components as ES6 Classes.

ES6 classes are a simple sugar over the prototype-based OO pattern. Having a single convenient declarative form makes class patterns easier to use, and encourages interoperability. Classes support prototype-based inheritance, super calls, instance and static methods and constructors.

By convention, all React components that are created this way *must* inherit from the `React.Component` class. Let's make a new file in our `example` directory called `react-reading-time.jsx` and add some code to it.

The first thing we need to do is import both the `React` and `ReactDOM` classes. Add these lines to the top of that file:

```
import React from 'react';
import ReactDOM from 'react-dom';
```

It's conventional to include all `import` statements at the top of your file, as you can't use anything until you've imported it(plus it makes your component files much more neatly organized, and we all want to be good coders and make our code nicely organized and easy for other developers to read).

Ok great. Now we've imported the React modules that we'll need, let's start creating our component.

Wait a minute! What's the difference between `React` and `ReactDOM`?!?! With the most recent release of React, the team over at Facebook decided to split up the core application logic and the rendering logic. This decision was largely based on the creation of `react-native` and the need for a cross platform rendering engine.

So let's go ahead and get started with our component. Let's add the core of the component:

```
class ReactReadingTime extends React.Component {
  render() {
    return (
      React.createElement('div', {className: "container"},
        "Hello React!"
```

```
      )
    );
  }
}
```

Awesome! But what's going on here? As I stated before, **EVERY** React component must inherit from the `React.Component` class, and that's what we're doing here. Also, *EVERY* React component must have a `render` method that returns the React element to be rendered on the page.

Rendering the component

This is where the `ReactDOM` class comes in. It's time for use to put it to use to render our component. At the very bottom of the file, add this:

```
ReactDOM.render(<ReactReadingTime />, document.getElementById('react'));
```

When using the `render` function, the first argument is the component you would like to render. In this case it's our main app component that we just created called `ReactReadingTime`. The second argument is the DOM node that you would like to attach the component to. Remember in our `index.html` file we created that `div` with an id of 'react'. Well that's where this component is going to
mount. Save this file and let's reload our page at `localhost:8881/example` and see what we've got!