React Class Components

All React websites are made up of components and this lesson gives an overview of how a basic React component is made.

What are Components?

Components are simply the building blocks that make-up what a website looks like. For example, Facebook's like button is a component. Let's try to code up a component.

```
import React from 'react';

export default class App extends React.Component {
  render() {
    return (
      <div> This is a component </div>
    );
  }
}
```

Explanation:

Line 1 of the index.js file imports the React library; this allows you to use all the functionality and features of React. Revisit the modules chapter if you want a refresher on how this works.

Line 3 of the app.js file creates a default export `App` class that inherits the `React.Component` class, again if any of these concepts are unclear, revisit the class inheritance lesson.

Line 4 is the `render()` function. That is what displays the content onto the screen. That's why the render() method is mandatory in React class components: The React Component from the imported React package instructs you to use it for displaying something in the browser. Furthermore, without extending from the React Component, you wouldn't be able to use other lifecycle methods including the `render()` method. For instance, there wouldn't be a `componentDidMount()` lifecycle method, because the component would be

an instance of a vanilla JavaScript class. And not only the lifecycle methods would go away, but React's state methods such as `this.setState()` for local state management wouldn't be available as well.

Right now, a simple 'This is a component' would get printed. Now to actually *use* your component, write the name of your component in tags as on *line 8* of the index.js file, yep, this is just like HTML! This makes your components reusable anywhere on your website.

However, as you have seen, using a JavaScript class is beneficial for extending the general class with your specialized behavior. Thus you can introduce your own class methods or properties. For example, in the following, we add a `getGreeting()` method as on *line 4* in the app.js file and render it with the `this` notation on *line 11*. Also notice that the `render()` function only returns *one* JSX/HTML element which is usually the `<div> ... </div>` element. Everything has to be nested within it. If you try to return more than one element, you'll get an error.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

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

Using components within components

React allows you to use your custom components within other components. For example, in the following, `App` is used within the component `OtherApp` on *line 8*. By doing this, you can build relatively bug-free complex components with ease. Also, note that **all React components need to start with a capital letter**.

```
import React from 'react';

export default class OtherApp extends React.Component {
  render() {
    return (
      <h1> other app </h1>
```

```
    );
  }
}
```

Caveats

JavaScript classes welcome using inheritance in React, which isn't the desired outcome for React because React favors composition over inheritance. So the only class you should extend from for your React components should be the official React Component.

In the next lesson, we'll see how we can make components more configurable using React Props.