React 101
Codecademy
React Components

Our First React Component

Hello World, Part 2… The Component

React applications are made out of components.

What's a component?

A component is a small, reusable chunk of code that is responsible for one job. That job is often to render some HTML.

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

class MyComponentClass extends React.Component {
  render() {
    return <h1>Hello world</h1>;
  }
};

ReactDOM.render(
  <MyComponentClass />,
  document.getElementById('app')
);
```

Import React

```
import React from 'react';
```

This creates an object named React which contains methods necessary to use the React library.

Import ReactDOM

In order to create components we import ReactDOM.

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

The methods imported from `react-dom` are meant for interacting with DOM.

The methods imported from `react` don't deal with the DOM at all. They don't engage directly with anything that isn't part of React.

DOM is used in React applications, but it isn't part of React. After all, the DOM is also used in countless non-React applications. Methods imported from 'react' are only for pure React purposes, such as creating components or writing JSX elements.

Create a Component Class

We know a React component is a small, reusable chunk of code that is responsible for one job, which often involves rendering HTML.

Here's another fact about components: we can use a JS class to define a new React component. We can also define components with JS functions, but we'll focus on class components first.

All class components will have some methods and properties in common. Rather than rewriting those same properties over and over again every time, we extend the `Component` class form the React library. This way, we can use code that we import from the React library, without having to write it over and over again ourselves.

After we define our class component, we can use it to render as many instances of that component as we want.

What is `React.Component`, and how do you use it to make a component class?

`React.Component` is a JS class. To create your own component class, you must subclass React.Component. You can do this by using the sytax

```
Class YourComponentNameGoesHere extends React.Component {}.
```

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

class MyComponentClass extends React.Component {
  render() {
    return <h1>Hello world</h1>;
  }
}

ReactDOM.render(
    <MyComponentClass />,
    document.getElementById('app')
);
```

On line 4, you know that we are declaring a new *component class*, which is like a factory for building React components. You know that `React.Component` is a class, which you must subclass in order to create a component class of your own. we also know that `React.Component` is a property on the object which was returned by `import React from 'react'` on line 1.

Name a Component Class

Subclassing `React.Component` is the way to declare a new component class.

When we declare a new component class, we need to give that component class name. On our finished component, our component class's name was `MyComponentClass`:

```
class MyComponentClass extends React.Component {
  render() {
    return <h1>Hello world</h1>;
  }
}
```

Component class variable names must begin with capital letters!

This adheres to JS's class syntax. It also adheres to a broader programming convention in which class names are written in UpperCamelCase.

There is also a React-specific reason why component class names must always be capitalized.

Component Class Instructions

- On line 1, `import React from 'react'` creates a JS object. This object contains properties that are needed to make React work, such as `React.createElement()` and `React.Component`.
- On line 2, `import ReactDOM from 'react-dom'` creates another JS object. This object contains methods that help React interact with the DOM, such as `ReactDOM.render()`.
- On line 4, by subclassing `React.Component`, we create a new component class. This is not a component! A component class is more like a factory that produces components. When we start making components, each one will come from a component class.
- Whenever we create a component class, we need to give that component class a name. That name should be written in UpperCamelCase. In this case, we chose name `MyComponentClass`.

Something we haven't talked about yet is the body of the component class: the pair of curly braces after `React.Component`, and all of the code between those curly braces.

Like all JSS classes, this one needs a body. The body will act as a set of instructions, explaining to your component class how it should build a React component.

The Render Function

A component class is like a factory that builds components. It builds these components by consulting a set of instructions, which you must provide.

These instructions should take the form of a class declaration body, i.e., they will be delimited by curly braces ({}).

```
class ComponentFactory extends React.Component {
  // instructions go here, between the curly braces
}
```

The instructions should be written in typical JS ES2015 class syntax.

There is only one property that you have to include in your instructions: a *render method*.

A *render method* is a property whose name is render, and whose value is a function. The term "render method" can refer to the entire property, or to just the function part.

```
class ComponentFactory extends React.Component {
  render() {}
}
```

A render method must contain a return statement. Usually, this return statement returns a JSX expression.

```
class ComponentFactory extends React.Component {
  render() {
    return <h1>Hello World</h1>;
  }
}
```

Create a Component Instance

`MyComponentClass` is now a working component class!

Let's make a React component!

```
<MyComponentClass />
```

To make a React Component, we write a JSX element. Instead of naming your JSX element something, give it the same name as a component class. Voila, there's a component instance!

JSX element can be either HTML, or component instances. JSX uses capitalization to distinguish between the two! That is the React-specific reason why component class names must begin with capital letters. In a JSX element, the first letter as a capital says that it will be a component instance and not a HTML tag.

Render A Component

We know that a component class needs a set of instructions, which tell the component class how to build components. When we make a new component class, these instructions are the body of our class declaration.

```
class MyComponentClass extends React.Component
{ // everything in between these curly-braces is instructions for how to build components

  render() {
    return <h1>Hello world</h1>;
  }
}
```

This class declaration results in a new component class, in this case named `MyComponentClass`. `MyComponentClass` has one method, named render. This all happens via standard JS class syntax.

When we make a component, that component inherits all of the methods of its component class. `MyComponentClass` has one method: `MyComponentClass.render()`. Therefore, `<MyComponentClass />` also has a method named render.

We can make as many different instances of a component and each will have the exact same methods.

To call a component's render method, we pass that component to `ReactDOM.render()`.

```
ReactDOM.render(
  <MyComponentClass />,
  document.getElementById('app')
);
```

`ReactDOM.render()` will tell `<MyComponentClass />` to call its render method.

`<MyComponentClass />` will call its render method, which will return the JSX element `<h1>Hello world</h1>`. `ReactDOM.render()` will then take that resulting JSX element, and add it to the virtual DOM.