Import React

Wooo! Your first React component!

In the last exercise, we started by importing from react. The line that did this is:

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

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

Later, we'll go over where the React library is imported from, and how the importing process works. For now, just know that this is how we import the React library.

You've already seen one of the methods contained in the React library: React.createElement(). Recall that when a JSX element is *compiled*, it transforms into a React.createElement() call.

For this reason, you *have to* import the React library, and save it in a variable named React, before you can use any JSX at all. React.createElement() must be available in order for JSX to work.

Import ReactDOM

In order to create our first component, we next imported the ReactDOM:

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

This line of code is very similar to line 1.

Both import JavaScript objects. In both lines, the imported object contains React-related methods.

However, there is a difference!

The methods imported from 'react-dom' are meant for interacting with the DOM. You are already familiar with one of them: ReactDOM.render().

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.

To clarify: the 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.

🧿 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. Let's talk about these instructions!

For starters, these instructions should take the form of a class declaration body. That means that they will be delimited by curly braces, like this:

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

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>;
    }
}
```

Of course, none of this explains the *point* of a render method. All you know so far is that its name is render, it needs a return statement for some reason, and you have to include it in the body of your component class declaration. We'll get to the 'why' of it soon!