Introduction to JSX

Get to know the basics of JSX and its syntax in React. You will also learn more about the App component and where to use it.

As mentioned before, *create-react-app* has already bootstrapped a basic application for you, and all files come with their own default implementations.

For now, the only file we will modify is the `src/App.js` file.

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}

export default App;
```

Don’t worry if you’re confused by the import/export statements and class declaration now. These are features of JavaScript ES6 we will revisit in a later chapter.

In the file you should see a **React ES6 class component** with the name App. This is a component declaration. After you have declared a component, you can use it as an element anywhere in your application. It will produce an **instance** of your **component** or, in other words, the component gets instantiated.

```
// component declaration
class App extends Component {
  ...
}

// component usage (also called instantiation for a class)
// creates an instance of the component
<App />
```

The returned **element** is specified in the `render()` method. The components you instantiated earlier are made up of elements, so it is important to understand the differences between a component, an instance of a component, and an element.

You should see where the App component is instantiated, else you couldn't see the rendered output in a browser. The App component is only the declaration, but not the usage. You can instantiate the component anywhere in your JSX with `<App />`. You will see later where this happens in this application.

The content in the render block may look similar to HTML, but it is actually JSX. JSX allows you to mix HTML and JavaScript. It is powerful, but it can be confusing when you are used to separating the two languages. It is a good idea to start by using basic HTML in your JSX. Open the `App.js` file and remove all unnecessary HTML code as shown:

```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <h2>Welcome to the Road to learn React</h2>
      </div>
    );
  }
}

export default App;
```

Now, you only return HTML in your `render()` method without any JavaScript. Let's define the "Welcome to the Road to learn React" as a variable. A variable is set in JSX by curly braces.

```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
  render() {
    var helloWorld = 'Welcome to the Road to learn React';
    return (
      <div className="App">
        <h2>{helloWorld}</h2>
      </div>
    );
  }
}

export default App;
```

Start your application on the command line with `npm start` to verify the changes you've made.

You might have noticed the `className` attribute. It reflects the standard `class` attribute in HTML. JSX had replaced a handful of internal HTML attributes, but you can find all the supported HTML attributes in React's documentation, which all follow the camelCase convention. On your way to learn React, expect to run across more JSX specific attributes.

Exercises:

- Define more variables and render them in JSX
  - Use a complex object to represent a user with a first name and last name
  - Render the user properties in JSX

Further Readings:

- Read about JSX
- Read about React components, elements and instances