COURSEWARE

Components

Contents

Overview

In this module, we will be discussing React Components.

What is a Component?

React is fundamentally about Components - Developers need to think in Components when working with ReactJS!

The official documentation for React states; “React is a library for building composable user interfaces.
It encourages the creation of reusable UI components which present data that changes over time.”

React Components should be created as a function, and are the building blocks of any React app.
These components will return a React element that describes how a section of the UI (User Interface) should appear.

JavaScript Syntax Extension

We can write our components in raw JavaScript, but the readability suffers; instead, we use JSX.

JSX, or JavaScript Syntax Extension, is written specifically for ReactJS and helps integrate HTML and JavaScript.

Essentially, it converts the HTML in the return of a component into a React element.

```
  <div className="header">
    <h1>My React App</h1>
  <div>
```

In JSX, this would be written as;

```
  React.createElement("div", {className: "header"}, React.createElement("h1",
null, "My React App"));
```

JSX can be used as an expression.

After compilation, they become regular JS function calls that evaluate to JS objects.

We can also assign variables to it and use expressions within these variables; for example:

```
const title = `My React App`;
const el = <h1>Welcome to {title}</h1>
```

This would render: **Welcome to My React App**

JSX Functions as Components

Where possible, components should be Functions.

Functions must return either;

- a **representation of a single native DOM component** (e.g.)
- a **custom composite component** defined by the developer
- **null or false** to indicate that the component should not be rendered

This return does not modify a component's state, and returns the same result each time it is invoked.

```
// Component with single native DOM element
const MyComponent = props => {
  return (
    <h1>Hello World</h1>
  );
};

export default MyComponent;
```

JSX Classes as Components

Components can also be created as a class.

The release of React version 16.8 introduced functionality to the library that means they are no longer necessary in most cases, but Classes will always be supported in React!

Much like Function components, Class components must have a render method that must return:

- a **representation of a single native DOM component** (e.g.)
- a **custom composite component** defined by the developer
- **null or false** to indicate that the component should not be rendered

```
// Component as a class with single-native DOM element
class MyClassComponent extends Component{
    render() {
        return (
            <div>
              <h2>I am a class component</h2>
              <p>I work similarly to Function components</p>
            </div>
        );
    }
};

export default MyClassComponent;
```

React Fragments and Importing

React fragments allow us to return multiple elements without having to add extra nodes to the DOM.

This is done through the use of empty tags encapsulating the elements; <> and </>

```
const Component = () => {
  return (
    <>
      <p>lorem ipsum…</p>
      <p>lorem ipsum…</p>
    </>
  );
};

export default MyComponent;
```

Tutorial

1. Within your src folder of your React app, create a new file for your component, with the file extension of `.jsx`, called `MyComponent.jsx`
2. Create a const called `MyComponent`, as an arrow function that takes no arguments:

```
const MyComponent = () => {}
```

3. Make the function return a single <h1> with the text Hello World:

```
return (
    <h1>Hello World</h1>
);
```

4. Export `MyComponent` as default:

```
export default MyComponent;
```

5. Within the App.js file in the same folder, delete EVERYTHING in its return and replace it with `MyComponent` as an element, ensuring it is imported at the top of the file:

```
import './App.css';
import MyComponent from './MyComponent';

const App = () => {
  return (
    <MyComponent />
  );
}

export default App;
```

6. Save all files and run the application – use `npm start`

Exercises

Edit your App.js page, using the general layout from the example in the module, to render some HTML on the page with some CSS styling.