03. REACT JSX.

React uses **JSX** for templating instead of regular JavaScript. It is not necessary to use it, however, following are some pros that come with it.

- It is **faster** because it performs optimization while compiling code to JavaScript.
- It is also **type-safe** and most of the errors can be caught during compilation.
- It makes it **easier and faster to write templates**, if you are familiar with HTML.

3.1. Using JSX

JSX looks like a regular HTML in most cases. We already used it in the Environment Setup chapter. Look at the code from ***App.jsx*** where we are returning div.

Example: *App.jsx*

```
import React from 'react';

class App extends React.Component {
   render() {
      return (
         <div>
            Hello World!!!
         </div>
      );
   }
}
export default App;
```

Even though it's similar to HTML, there are a couple of things we need to keep in mind when working with JSX.

3.2. Nested Elements

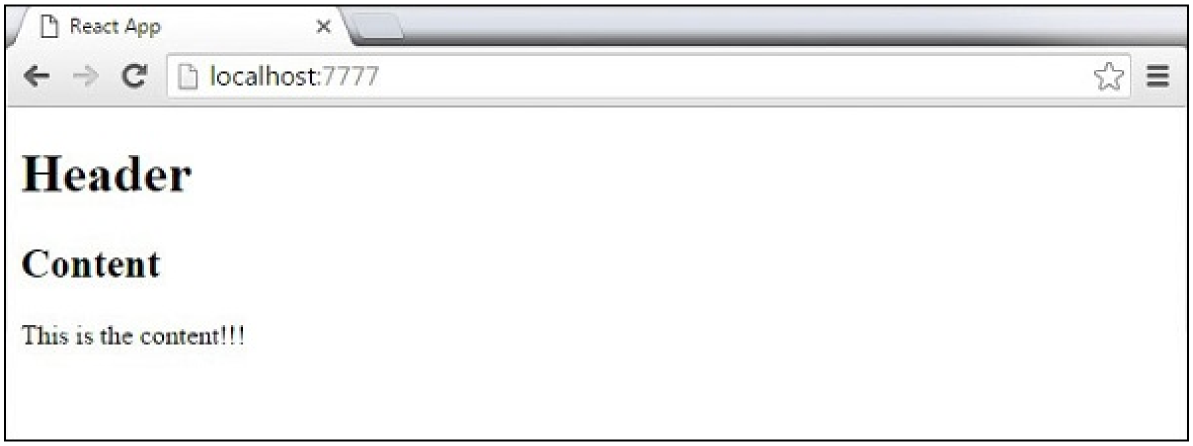
If we want to return more elements, we need to wrap it with one container element. Notice how we are using div as a wrapper for h1, h2 and p elements.

Example:

```
import React from 'react';

class App extends React.Component {
   render() {
      return (
         <div>
            <h1>Header</h1>
            <h2>Content</h2>
            <p>This is the content!!!</p>
         </div>
      );
   }
}
export default App;
```

Ouput:



Header

Content

This is the content!!!

3.3. Attributes

We can use our own custom attributes in addition to regular HTML properties and attributes. When we want to add custom attribute, we need to use data- prefix. In the following example, we added ***data-myattribute*** as an attribute of p element.

Example:

```
import React from 'react';

class App extends React.Component {
   render() {
      return (
         <div>
           <h1>Header</h1>
           <h2>Content</h2>
           <p data-myattribute = "somevalue">
                    This is the content!!!
           </p>
         </div>
      );
   }
}
```

```
export default App;
```

3.4. JavaScript Expressions

JavaScript expressions can be used inside of JSX. We just need to wrap it with curly brackets **{}**. The following example will render 2.

Example:

```
import React from 'react';

class App extends React.Component {
   render() {
      return (
         <div>
            <h1>{1+1}</h1>
         </div>
      );
   }
}
export default App;
```

Output:

```
2
```

We cannot use if else statements inside JSX, instead we can use conditional (ternary) expressions. In the following example, variable i equals to 1 so the browser will render true, If we change it to some other value, it will render false.

Example:

```
import React from 'react';

class App extends React.Component {
   render() {
      var i = 1;
      return (
         <div>
            <h1>{i == 1 ? 'True!' : 'False'}</h1>
         </div>
      );
   }
}
export default App;
```

Output:

```
True!
```

3.5. Styling

React recommends using inline styles. When we want to set inline styles, we need to use *camelCase* syntax. React will also automatically append px after the number value on specific elements. The following example shows how to add ***myStyle*** inline to h1 element.

Example:

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

```
class App extends React.Component {
   render() {
      var myStyle = {
         fontSize: 100,
         color: '#FF0000'
      }
      return (
         <div>
            <h1 style = {myStyle}>Header</h1>
         </div>
      );
   }
}
export default App;
```

Output:

```
Header
```

3.6. Comments

When writing comments, we need to put curly brackets {} when we want to write comment within children section of a tag. It is a good practice to always use {} when writing comments, since we want to be consistent when writing the app.

Example:

```
import React from 'react';

class App extends React.Component {
   render() {
      return (
         <div>
            <h1>Header</h1>
            {//End of the line Comment...}
            {/*Multi line comment...*/}
         </div>
      );
   }
}
export default App;
```

3.7. Naming Convention

HTML tags **always use lowercase tag names**, while React components start with Uppercase.

Note − You should use ***className*** and ***htmlFor*** as XML attribute names instead of ***class*** and ***for***.

This is explained on React official page as −

> *Since JSX is JavaScript, identifiers such as class and for are discouraged as XML attribute names. Instead, React DOM components expect DOM property names such as className and htmlFor, respectively.*