Unit 2: JSX, Components, Component Life Cycle

Why JSX:

JSX 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.

JSX makes it easier and faster to write templates if you are familiar with HTML.

JSX

Syntax:

```
import React from 'react';

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

export default App;
```

Notes when using JSX:

1. Nested Elements:

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

JSX

```
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;
```

Notes when using JSX:

2. JavaScript Expressions:

JavaScript expressions can be used inside of JSX. You just need to wrap it with curly brackets **{}**. Example below will render **2**.

You can not use **if else** statements inside JSX but you can use **conditional (ternary)** expressions instead.

```
{i == 1 ? 'True!' : 'False'}
```

JSX

```
import React from 'react';

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

export default App;
```

JSX

```
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;
```

Notes when using JSX:

3. Styling:

React recommends using inline styles. When you want to set inline styles, you need to use **camelCase** syntax. React will also automatically append **px** after the number value on specific elements. You can see below how to add **myStyle** inline to **h1** element.

JSX

```
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;
```

Notes when using JSX:

4. Comments:

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

JSX

```
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;
```

Components

Stateless Example: (eg in code sample)

Component Life Cycle

Mounting

These methods are called when an instance of a component is being created and inserted into the DOM:

- `constructor()`
- `componentWillMount()`
- `render()`
- `componentDidMount()`

Component Life Cycle

Updating

An update can be caused by changes to props or state. These methods are called when a component is being re-rendered:

- `componentWillReceiveProps()`
- `shouldComponentUpdate()`
- `componentWillUpdate()`
- `render()`
- `componentDidUpdate()`

Component Life Cycle

Unmounting

This method is called when a component is being removed from the DOM:

```
componentWillUnmount()
```