JSX and you...

What is JSX?

Put simply, **JSX allows you to create JavaScript objects using HTML syntax.**

Remember in the last lesson we created our first component? The render method looked like this:

```
render() {
  return (
    React.createElement('div', { className: 'container' },
      'Hello React!'
    )
  );
}
```

We can use JSX to transform that render method to look like this:

```
render() {
  return (
    <div className='container'>
      Hello React!
    </div>
  );
}
```

Why use JSX?

There are several great reason for using JSX over the JavaScript syntax for creating React views. More casual developers like designers will be much more comfortable with JSX, as it is easily recognizable as HTML. Also, because JSX has opening and closing tags like HTML, it makes large trees of UI components much, much easier to read and reason about.

Rendering JSX

React can render either HTML tags or React components. There is an important distinction to make however. If you use lowercase letters React will render an `html` tag. So the following:

```
ReactDOM.render(<div className='container'></div>, document.getElementById('react'));
```

Will render the following in the DOM:

```
<div class='container'></div>
```

However, when you use a **capitalized** tag name when rendering, React will attempt to render a component with that name:

```
ReactDOM.render(<MyComponent />, document.getElementById('react'));
```

The above code will render the `MyComponent` class and all of it's children.

HTML attributes in JSX

You may have noticed that in our above examples we're using `className` instead of `class` in our HTML markup. Because JSX is just JavaScript, we have to avoid using reserved identifiers such as `class` and `for` in our JSX. Instead, we have to use name like `className` and `htmlFor` respectively for these attributes.

For a more complete list of JSX gotchas visit this page.

And for a list of differences between JSX and the DOM check here.

JavaScript Expressions

It's absolutely possible and acceptable to use JavaScript expressions inside of JSX.

Attribute Expressions

If you'd like some logic inside one of your JSX attributes, simply wrap it in curly braces `{}`:

```
<div className={this.props.valid ? 'valid' : 'invalid'}></div>
```

Boolean attributes

If the value of an attribute is omitted, it is assumed to be true, so the following are equivalent:

```
<button disabled>Submit</button>
<button disabled={true}>Submit</button>
```

Child expressions

You can also use JavaScript expressions to render children:

```
<Nav>{ window.isLoggedIn ? <Dashboard /> : <Login /> }</Nav>
```