React conventions

We've already discussed some React conventions like using `className` in JSX attributes. We're going to expand on that briefly just to give you a quick overview of how developers in the community organize their code and write reusable components.

propTypes

We'll go into this in more detail in the next section, but React provides a `propTypes` declaration that you should use to define what props a component may receive, their types, and if they're required or optional:

```
propTypes = {
  foo: React.PropTypes.boolean,
  bar: React.PropTypes.string.isRequired
}
```

This really helps your code to be self documenting, and will help future developers out a lot when it's time for them to work on your codebase.

JSX structure

Making your JSX well formed and structured is essential to writing clean, easy to read code.

Multi-line components

It's good practice to keep JSX elements on separate lines in nested components.

This is bad:

```
<div><Component1 /><Component2 /></div>
```

This is better:

```
<div>
  <Component1 />
  <Component2 />
</div>
```

See how much easier to read and reason about the second version is? Always put your components on their own lines. Multi-line blocks of JSX must be encapsulated in a pair of curly braces.

Conditional elements

If you have elements that will be rendered only if a specific condition is met, then it's best practice to define those elements in a variable rather than putting them inline in the JSX.

Bad:

```
render() {
  return (
    <div>
      { window.loggedIn ? <SecretComponent /> : null }
    </div>
  );
}
```

Good:

```
class MyComponent extends React.Component {
  render() {
    return (
      <div>
        {this.renderSecretComponent()}
      </div>
    );
  }

  renderSecretComponent() {
    return (window.loggedIn) ? <SecretComponent /> : null;
  }
}
```

Again, much nicer, easier to read and reason about. This is only a simple contrived example, but you can imagine with more complicated state and components this could get really ugly.

Component attribute indentation

When there are more than 3 or so attributes on a component, it is a good idea to move them to their own line and use indentation to keep them organized:

```
<Component
  show={true}
  classes='foo bar'
  otherAttribute='baz'
/>
```

Wrapping up

This is by no means an exhaustive list of best practices and conventions, but it will get you started on writing good, clean, reusable, maintainable, and fun React components.