SECTION 5:
MIXINS AND PURE RENDER

MIXIN

- Components are the best way to reuse code in React, but sometimes very different components may share some common functionality. These are sometimes called cross-cutting concerns. React provides mixins to solve this problem.
- Mixin has access to lifecycle methods
- Unfortunately ES6 launched without any mixin support. Therefore, there is no support for mixins when you use React with ES6 classes.
- React team is working on making it easier to support such use cases without resorting to mixins.

MIXINS

- How to create and use:

```
var LogMixin = {
    componentWillMount: function() {
        this.logs = [];
    },
    writeLog: function(txt) {
        this.logs.push(txt);
    },
    readLog: function() {
        console.log(this.logs.join('\n'))
    }
};
```

```
var UserName = React.createClass({
    mixins: [LogMixin], // Use the mixin
    getInitialState: function() {
        return {name: "paul"};
    },
    onClick: function() {
        this.setState({name: "victor"});
    },
    componentWillUpdate: function(nextProps, nextState){
        this.writeLog(nextState.name);
    },
    componentWillUnmount: function(){
        this.readLog();
    },
    render: function() {
        return (
            <p>
                Current user: {this.state.name}
                <button onClick={this.onClick}>Change user name</button>
            </p>
        );
    }
});
```

MIXINS

- **As a Mixins people usually use:**
  - Lifecycle Hooks and State Providers
  - Utility Functions
- But in most cases can be replaced by composition
- With ES7 coming, you can also use Decorators instead of mixins (@Decorator)

PURE RENDER MIXIN

- If your React component's render function is "pure" (in other words, it renders the same result given the same props and state), you can use this mixin for a performance boost in some cases.
- The PureRenderMixin is a mixin that overrides shouldComponentUpdate and only re-renders the component if the props or state have actually changed
- It is a pretty big optimization on top of React's already good performance.
- It also means you can call setState often without worrying about spurious re-renders
- No need to make checks like this:

```
if (this.state.someVal !== computedVal) {
   this.setState({someVal: computedVal})
}
```

PURE RENDER MIXIN

- To use pure render mixin your render must be pure.

```
render: function () {
  //...
  if (this._previousFoo !== this.props.foo) { // <-- IMPURE
    return renderSomethingDifferent();
  }
}
```

- Example:

```
var PureRenderMixin = require('react-addons-pure-render-mixin');
React.createClass({
  mixins: [PureRenderMixin],

  render: function() {
    return <div className={this.props.className}>foo</div>;
  }
})
```