SECTION 8:
REACT TESTING

REACT TESTING

- Lets say we want to test Couple components:

```
import React from 'react';

class Label extends React.Component {
  render() {
    return <span>Hello {this.props.name}</span>;
  }
}

export default Label;
```

```
import React from 'react';
import Label from './Label';

class Button extends React.Component {
  render() {
    return <div><Label name={this.props.name}
/></div>;
  }
}

export default Button;
```

- It’s pretty easy! Lets take a look

REACT TESTING

- First of all you will need to install tool created by facebook: react-addons-test-utils
- This module contains useful helpers which cover almost everything you need to write unit test
- Example:

```
import expect from 'expect';
import {createRenderer} from 'react-addons-test-utils';

import Button from './Button.js';
import Label from './Label.js';

describe('Button', () => {
  it('works', () => {
    let renderer = createRenderer();
    renderer.render(<Button name="John" />);
    let actualElement = renderer.getRenderOutput();
    let expectedElement = <div><Label name="John" /></div>;
    expect(actualElement).toEqual(expectedElement);
  });
});
```

REACT TESTING

- As a result of getRenderOutput() method you get the result of its render function (a ReactElement object)
- At that point you can inspect its .type and .props to verify that the correct result was rendered
- Why there's Label name="John" instead of Lable component content?
- Because React Test Utils createRenderer under the hood uses **Shallow Rendering technique**

REACT TESTING: SHALLOW RENDERING

- Renders a component "one level deep”
- Assert facts about what its render method returns, without worrying about the behavior of child components, which are not instantiated or rendered
- Does not require a DOM.
- Errors in children components wouldn’t propagate to top level components, making our tests more isolated and reliable

REACT TESTING: DOM RENDER

- But sometimes we actually want component to be rendered in DOM. Why?
  - Because we are planning to use ref
  - We are planning to test event listeners
  - We need to test children component structure
- You can use renderIntoDocument method from Test Utils.
- Obviously this method requires some DOM to be rendered in.
- Result of this method is ReactElement object, but with all nested children

REACT TESTING: DOM RENDER

- Example. We will make our previous example more complicated - we add click handler.

  How to test it?

```
class Button extends React.Component({
  getInitialState:function () {
    return {event: ''}
  },
  handleClick:function(){
    this.setState({event:"click"});
  },
  render() {
    return <div ref="button" onClick={this.handleClick}>
                        <Label name={this.props.name} />
              </div>;
  }
});
```

REACT TESTING: DOM RENDER

- Test for Button will look this way:

```
it('should have click event state', function (done) {
  var button = ReactTestUtils.renderIntoDocument(Button);
  ReactTestUtils.Simulate.click(button.refs.button.getDOMNode());
  button.state.event.should.equal('click');
  done();
});
```