React Extras

Dev vs. Prod

- By default React performs extra checking of your code
  - such as checking **`propTypes`**
  - helpful when developing an app
  - useful messages appear in browser console
- Disable for better production performance
  - set **`NODE_ENV`** environment variable to "**`production`**"
  - enable these Babel optimizations in **`.babelrc`**
    - **`optimization.react.constantElements`**
    - **`optimization.react.inlineElements`**

Rendering HTML Strings

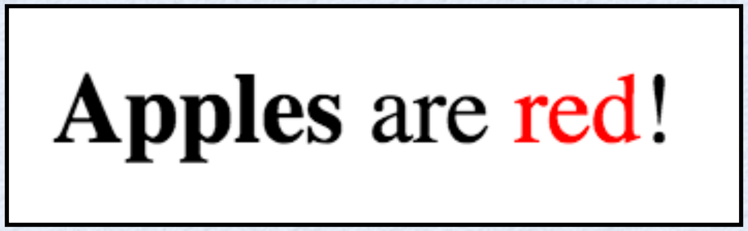
- Need to trust that string doesn't contain anything dangerous

```
import React from 'react';
import ReactDOM from 'react-dom';

const htmlString =
  '<b>Apples</b> are <span style="color: red">red</span>!';

const MyComponent = () =>
  <div dangerouslySetInnerHTML={{__html: htmlString}}/>;

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



Apples are red!

Higher Order Components

- A “higher order component factory” is a function that takes a component and returns a decorated version of it
  - can use this function as an ES7 decorator
  - an example is adding validation to an input component
- Pattern

```
const HOComponent = Component =>
  class extends React.Component {
    constructor(props) {
      super(props);
      // Can bind methods.
      // Can set initial state.
    }
    // Can define lifecycle methods.
    // Can define render method that renders
    // the passed component and more.
    render() {
      return <Component props {...this.props}/>
    }
  };

const NewComponent = HOComponent(OldComponent);
// Now render <NewComponent/> somewhere.
```

interesting that a function can create a class this way (anonymous class)

can apply this to many components

PureRenderMixin

- Provided shortcut for simple `shouldComponentUpdate` checking
- Only for components that render the same DOM given the same props and state (pure components)
  - should do this!
- Can only use in components defined using `React.createClass`
  - not ES6 classes or stateless functions
  - use react-addons-shallow-compare for those (see next slide)

```
import PureRenderMixin from 'react-addons-pure-render-mixin';
React.createClass({
  mixins: [PureRenderMixin],

  render: function() {
    ...
  }
});
```

Shallow Compare

- Helper function that is an alternative to `PureRenderMixin` for components defined with an ES6 class
- https://facebook.github.io/react/docs/shallow-compare.html
- Steps to use
  - `npm install react-addons-shallow-compare`
  - require the add-on
  - use in `showComponentUpdate` method
- Only compares top-level properties
- Returns true if any diffs are found
- Alternative
  - for components that only get data from immutable props

Both **only** perform **a shallow compare** of the property values of props and state.
This **can miss some changes**.
Another approach is to write your own utility method.
See `todo-redux-rest/public/deep-equal.js`.

```
const shallowCompare =
  require('react-addons-shallow-compare');
class MyComponent extends React.Component {
  shouldComponentUpdate(nextProps, nextState) {
    return shallowCompare(
      this, nextProps, nextState);
  }
}
```

```
class MyComponent extends React.Component {
  shouldComponentUpdate(nextProps, nextState) {
    return nextProps.propName !== this.props.propName;
  }
}
```

Rendering to a String

- Only for use on server
- `ReactDOMServer.renderToString(`*`jsx`*`)`
  - returns a string of HTML that includes React-specific attributes
- `ReactDOMServer.renderToStaticMarkup(`*`jsx`*`)`
  - similar to previous function, but doesn't include React-specific attributes

```
import ReactDOMServer from 'react-dom/server';

const jsx = <h1>Hello, World!</h1>;

console.log(ReactDOMServer.renderToString(jsx));
// <h1 data-reactid=".0" data-react-checksum="60427361">Hello, World!</h1>

console.log(ReactDOMServer.renderToStaticMarkup(jsx));
// <h1>Hello, World!</h1>
```

Undo

- Two approaches
- Both are mostly only useful for development
  - because they don't address undoing changes to persistent stores like databases
- 1) All app state owned by top-level component
  - keep **stack of state objects**
    - only those to be undone; ex. maybe not each keystroke in an input
  - on undo, pop last state off stack and restore new, last state on stack
  - gift app uses this approach
- 2) All app state owned by Redux
  - can use same approach
  - alternative is to keep **stack of** all **actions** that have been applied
  - on undo, pop last action off stack and replay all remaining actions from the beginning
- Also see https://github.com/gaearon/react-hot-loader

Using Bootstrap ...

gift app uses this

- Provides
  - good, default **styling** - CSS classes like `form-inline`, `btn`, `btn-default`, `form-control`, ...
  - some **widgets** - buttons, dropdowns, modals, tooltips, breadcrumbs,
    tabs, tables, form elements, carousels, glyphicons, alerts, progress bars, ...
  - **responsiveness** - jump start on making apps compatible with mobile devices
- Can choose version where CSS is based on LESS or Sass
- Steps to use with webpack
  - `npm install --save-dev bootstrap-loader file-loader resolve-url-loader url-loader`
  - `npm install --save bootstrap-sass jQuery react-bootstrap`
  - import bootstrap and custom CSS

    ```
    import 'bootstrap-loader';
    import './my-app.scss';
    ```
  - update `webpack.config.js`
    - see next slide

... Using Bootstrap

- **`webpack.config.js`** updates

```
  plugins: [
    new webpack.ProvidePlugin({
      $: 'jquery',
      jQuery: 'jquery'
    })
  ],
```

```
  module: {
    loaders: [
      ... existing loaders here ...
      {                                                    why?
        test: /\.(ttf|eot|svg|woff(2)?)(\?[a-z0-9]+)?$/,
        loader: 'file-loader'
      },
      {test: /\.css$/, exclude: /node_modules/, loader: 'style!css'},
      {test: /\.scss$/, exclude: /node_modules/, loader: 'style!css!sass'}
    ]
  }
}
```

react-addons-perf ...

- Helps find performance bottlenecks
  - and identifies components that would benefit from implementing `shouldComponentUpdate`
- Logs results in dev tools console
- Use in development, not production
- To install
  - `npm install --save react-addons-perf`
- See example usage in todo-redux-rest app

... react-addons-perf

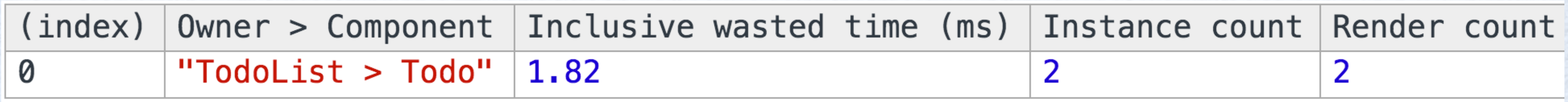
```
// In code for main component ...
import Perf from 'react-addons-perf';

// At top of an event handling method ...
Perf.start();

// After render of main component ...
Perf.stop();
const measurements =
  Perf.getLastMeasurements();
// Choose the statistics to output.
Perf.printInclusive(measurements);
Perf.printExclusive(measurements);
Perf.printWasted(measurements);
Perf.printDOM(measurements);
```

- Inclusive
  - prints overall time taken in a table
- Exclusive
  - same as Inclusive, but doesn't include times taken to mount components
    - processing props, `getInitialState`, `componentWillMount`, `componentDidMount`, etc.
    - will get an empty array if none
- Wasted
  - time spent on components that didn't render
    - render didn't change, so DOM wasn't modified
    - can indicate components that would benefit from `shouldComponentUpdate`
    - will get an empty array if none

In this example, there were two todos and a third was added.
It wasted time determining if the original 2 needed to be re-rendereed.



(index)	Owner > Component	Inclusive wasted time (ms)	Instance count	Render count
0	"TodoList > Todo"	1.82	2	2

- DOM
  - lists DOM manipulations that were performed

react-intl

- Provides React components for internationalization and localization
  - for the differences between these, see https://www.w3.org/International/questions/qa-i18n
- From Yahoo
  - see http://formatjs.io/react/
- Features
  - displays **numbers** with separators in a locale-specific way
  - displays **dates** and **times** in a locale-specific way
  - displays **dates relative to "now"**
  - **pluralizes** labels in **strings**
  - lookup and format **language-specific strings**
  - supports over 150 languages

React Native

- Combines JavaScript, React, and a native component library to build native Android and iOS apps
  - uses native components, not a web view
- See https://facebook.github.io/react-native/

Web Components

- Can use React components inside Web Components makes less sense to do this
  - use `ReactDOM.render` to render JSX for a React component to an element inside Web Component HTML
- Can use Web Components inside React components makes more sense to do this
  - embed instances of Web Components in JSX returned by the `render` method or by a stateless functional component
- Web components often expose an imperative API
  - ex. a video component could expose `play` and `pause` methods
- To enable calling these, attach a ref to the component and interact with the DOM node directly
  - recommended solution is to write a React component that behaves as a wrapper for the web component
  - attach event handlers to the Web Component inside the React wrapper
- For more detail, see https://facebook.github.io/react/docs/webcomponents.html