React in 30 Minutes

R. Mark Volkmann
Object Computing, Inc.
2/16/17

Slides and code are available at
`https://github.com/mvolkmann/react-swapper`

React

- A library for creating web applications

- One-way data flow makes applications easier to implement and understand

  - and the "virtual DOM" makes this very fast

- Components do

  - decide what to render when given certain data

  - decide what to do when events occur, like button clicks and input changes

    - often they just pass data to a function that was passed to the component

- Components do not

  - directly modify the DOM

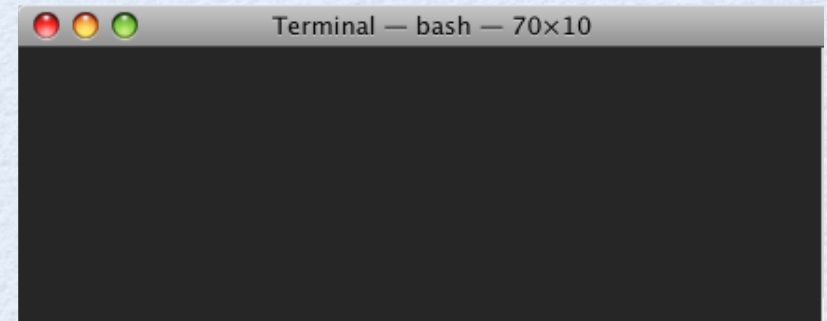 - modify the state of other components

Getting Started

- Browse `nodejs.org`

- Click "Latest Features" button
  to download installer for your platform

  <div style="border:2px solid #fff; background:#2e7d00; color:#fff; display:inline-block; padding:1em 2em; text-align:center;">

  ## v7.5.0 Current

  Latest Features

  </div>

- Double-click downloaded file and follow instructions
  - installs **Node.js** and **npm**

- Open a terminal or command prompt

- Enter `npm install -g create-react-app`

Creating an App

- Enter `create-react-app app-name`

  - takes about 20 seconds to complete because it downloads and installs many npm packages

- Enter `cd app-name`

- Enter `npm start`

- Starts local HTTP server

- Opens default browser to local app URL

Welcome to React

To get started, edit `src/App.js` and save to reload.

4

Benefits of create-react-app

- Creates directory structure and files including `package.json`

- Installs and configures many tools and libraries

- Provides a local web server for use in development

- Provides **watch and live reload**

- Uses Jest test framework which supports **snapshot tests**

- Lets Facebook maintain the build process
  - future benefits from future improvements

- "npm build" produces small production deploys

Notable Packages Installed

- **Babel** - JavaScript transpiler (ES6+ to ES5) and more

- **ESLint** - pluggable JavaScript linter

- **Istanbul** - code coverage tool

- **Jest** - JavaScript test framework supporting snapshot tests

- **Lodash** - JavaScript utility library

- **PostCSS** - tool for transforming styles with plugins

  - "can lint CSS, support variables and mixins,
    transpile future CSS syntax, inline images, and more"

- **React** - of course

- **ReactDOM** - provides DOM-specific methods

- **react-scripts** - scripts and configuration used by create-react-app

  - source of future benefits

- **SockJS** - WebSocket emulation (tries to use native WebSockets first)

- **UglifyJS** — JavaScript parser/compressor/beautifier

- **Webpack** - module and asset bundler

- **webpack-dev-server** - an Express server that servers a webpack bundle

- **whatwg-fetch** - polyfill for Fetch API used to make REST calls

create-react-app Scripts

- **`package.json`** file generated by create-react-app defines several scripts

- **`npm start`**

  - starts webpack-dev-server and opens browser to the app

- **`npm test`**

  - runs Jest-based tests under **`__tests__`** directory

- **`npm run build`**

  - creates a compressed, production build

  - produces many files in a new **`build`** directory

  - most important are
    `index.html`,
    `static/css/main.hash.css`, and
    `static/js/main.hash.js`

  - **`index.html`** refers to the **`.css`** and **`.js`** files

7

State and Props

- Two ways data is provided to a component

- Values of **props** never change

- Values of **state** can change over the life of components

- **Props** are passed to components via what look like HTML attributes

- The type of each prop can be described using `React.Proptypes`

  - provides error checking during development

- **State** is initially provided by component definitions
  and is updated by calls to `this.setState(newState)`

  - causes the component to re-render using Virtual DOM diffing

- Larger applications often use **Redux** to manage application state

  - overkill for small to medium sized applications

Component Types

- Stateless functional components
  - defined by a single function
  - get data from props
  - do not hold state

```
const Greeting = ({name}) =>
  <h1>Hello, {name}!</h1>;
```

- Class-based components
  - defined by a class that extends `React.Component`
  - get data from props
  - can hold state
  - can define lifecycle methods like `componentDidMount` and `shouldComponentUpdate`

```
class Greeting extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}!</h1>;
  }
}
```

9

JSX - JavaScript XML

- A syntax for specifying what components should render

- Looks very **similar to HTML**

- HTML element names must be lowercase

- Custom element names must start uppercase

- **Props** look like attributes and must use **camel-case**, not snake-case

  - ex. `onClick`, not `on-click`

- Values of event handling props must be references to functions

  - ex. `onClick={this.handleClick}`

- Converted to calls to JavaScript functions
  by Babel plugin "transform-react-jsx"

  - browsers do not understand JSX syntax

Time to Look at Code!

- **`Select`** in **`src/select.js`**

  - a list of strings that supports a single selection

  - implemented with a stateless functional component

- **`Swapper`** in **`src/swapper.js`**

  - renders two Select components with two buttons between them for moving items from one Select to the other

  - implemented with a class-based component

- **`App`** in **`src/App.js`**

  - renders a Swapper for selecting favorite ice cream flavors

- CSS in **`src/App.css`**

  - uses Flexbox to layout all components

Favorite Ice Cream Flavors

butter pecan
chocolate
chocolate chip
chocolate mint
cookie dough

⇒

⇐