REACT.JS WORKSHOP

Pratik Patel @prpatel

LAB0

- Install Node.js ^4.6.0 https://nodejs.org/
- Clone Git repo:
  - https://github.com/prpatel/connect.tech-react-workshop
- cd connect.tech-react-workshop && npm install
- cd Lab1; npm start
- open browser to http://localhost:8080/
- approximate start time 9:15AM

@prpatel

- FOCUS ON “MODERN” WEB DEV
- ES6
- ADVANCED BUILD TOOLS

@prpatel

- UI ONLY LIBRARY
- DEVELOPED AT FACEBOOK
- POWERS INSTAGRAM.COM

@prpatel

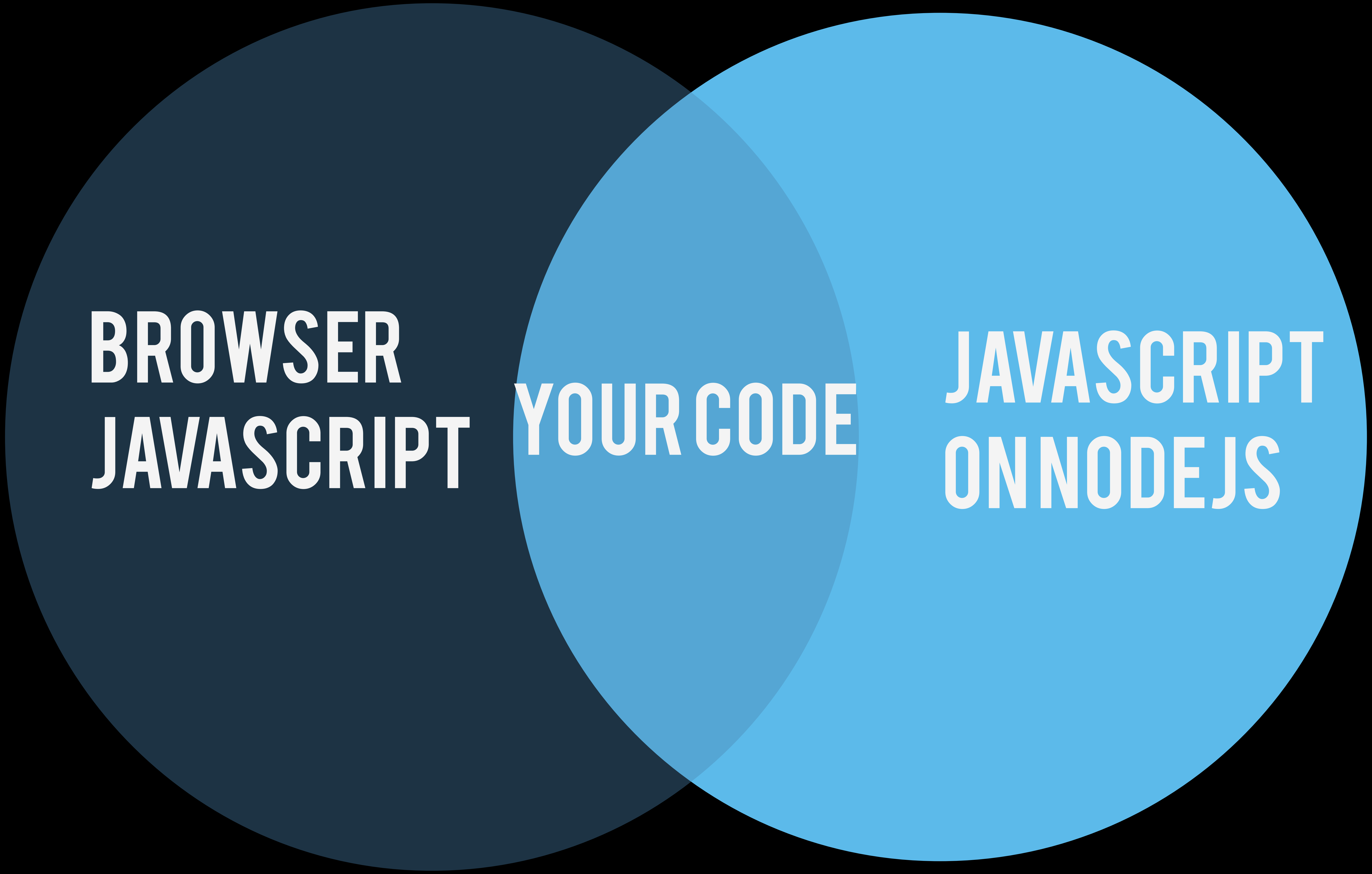
REACTJS CONCEPTS

CODE RUNS ON THE BROWSER

__OR__

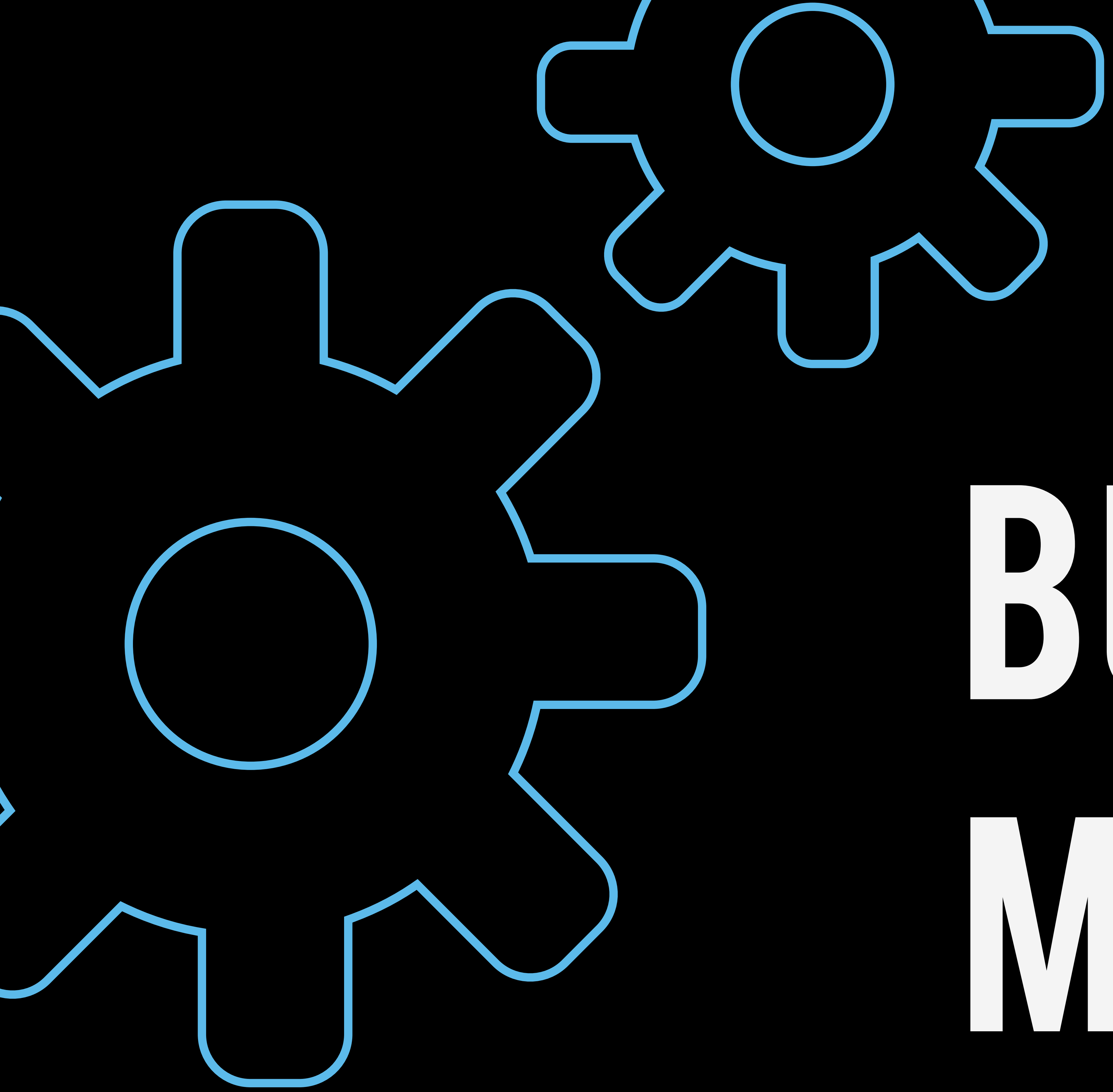
THE SERVER

ISOMORPHIC



@prpatel

Allows great flexibility and performance management



BUT THAT’S NOT THE MAIN REASON REACTJS IS FAST

- VIRTUAL DOM
- SMART DIFF'ING OF DOM
- BATCHED DOM UPDATES

OTHER FRAMEWORKS ARE / WILL BE USING VIRTUAL DOM

@prpatel

YES, MITRHIL

@prpatel

TYPICAL BROWSER UPDATE

@prpatel

PARSE
MARKUP
&
CSS

CREATE
RENDER
TREE

REFLOW
&
REPAINT

@prpatel

REFLOW
&
REPAINT

DO THIS ALOT = SLOW

@prpatel

ANY DOM UPDATE

HIDE DOM NODE

MOVE / ANIMATE DOM

STYLE CHANGES

MANAGE DOM STATE

VIRTUAL DOM

REDUCE NUMBER OF CHANGES TO BE APPLIED

SMART DIFF'ING OF DOM

REDUCE NUMBER OF REFLOWS / REPAINTS

BATCHED DOM UPDATES

REACT BASICS

@prpatel

COMPONENTS

- BASIC BUILDING BLOCKS
- JSX
- JAVASCRIPT XML SYNTAX

@prpatel

I know a name is just a sound somebody makes when they need you...

- Jared

@prpatel

```
<script type="text/jsx">
/** @jsx React.DOM */
    ReactDOM.render(
    <h1>Hello, world!</h1>,
    document.getElementById('myDiv')
    );
</script>
```

@prpatel

- HTML-LIKE
- JSX DIFFERS FROM HTML
- [HTTP://FACEBOOK.GITHUB.IO/REACT/DOCS/JSX-GOTCHAS.HTML](http://facebook.github.io/react/docs/jsx-gotchas.html)

@prpatel

WHY WRITE IN HTML-LIKE WHEN YOU CAN JUST WRITE CODE?

@prpatel

```
ReactDOM.render(
  React.DOM.h1(null, 'Hello, world!'),
  document.getElementById('myDiv')
);
```

@prpatel

- REACT.RENDERCOMPONENT
- 1: COMPONENT TO RENDER
- 2: WHERE TO MOUNT

@prpatel

```
ReactDOM.render(
 component, whereToAttachToDOM
);
```

@prpatel

CUSTOM COMPONENTS: createClass

@prpatel

```
class HelloWorld extends
React.Component({
    render: function(){
        return (<h1>Hello, world!</h1>);
    }
});
```

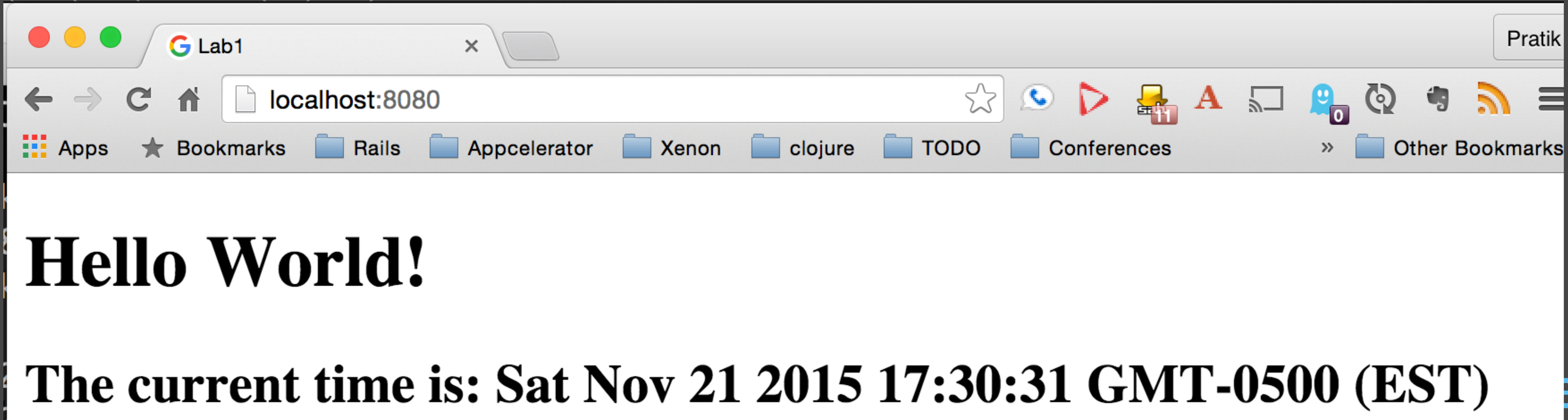
```
// use component
ReactDOM.render(
   <MyComponent/>,
   document.getElementById('myDiv')
);
```

Lab 1: Hello World

@prpatel

Lab 1: Hello World

- cd Lab1; npm start
- open http://localhost:8080/
- Inside of JSX use: {dateVar} to show date string
- edit app.js (notice auto-reload!) to make this



Hello World!

The current time is: Sat Nov 21 2015 17:30:31 GMT-0500 (EST)

Lab 1: Discussion

@prpatel

Um, how does this all work?

@prpatel

- NPM -> Webpack
- webpack-dev-server
- compiles + bundles

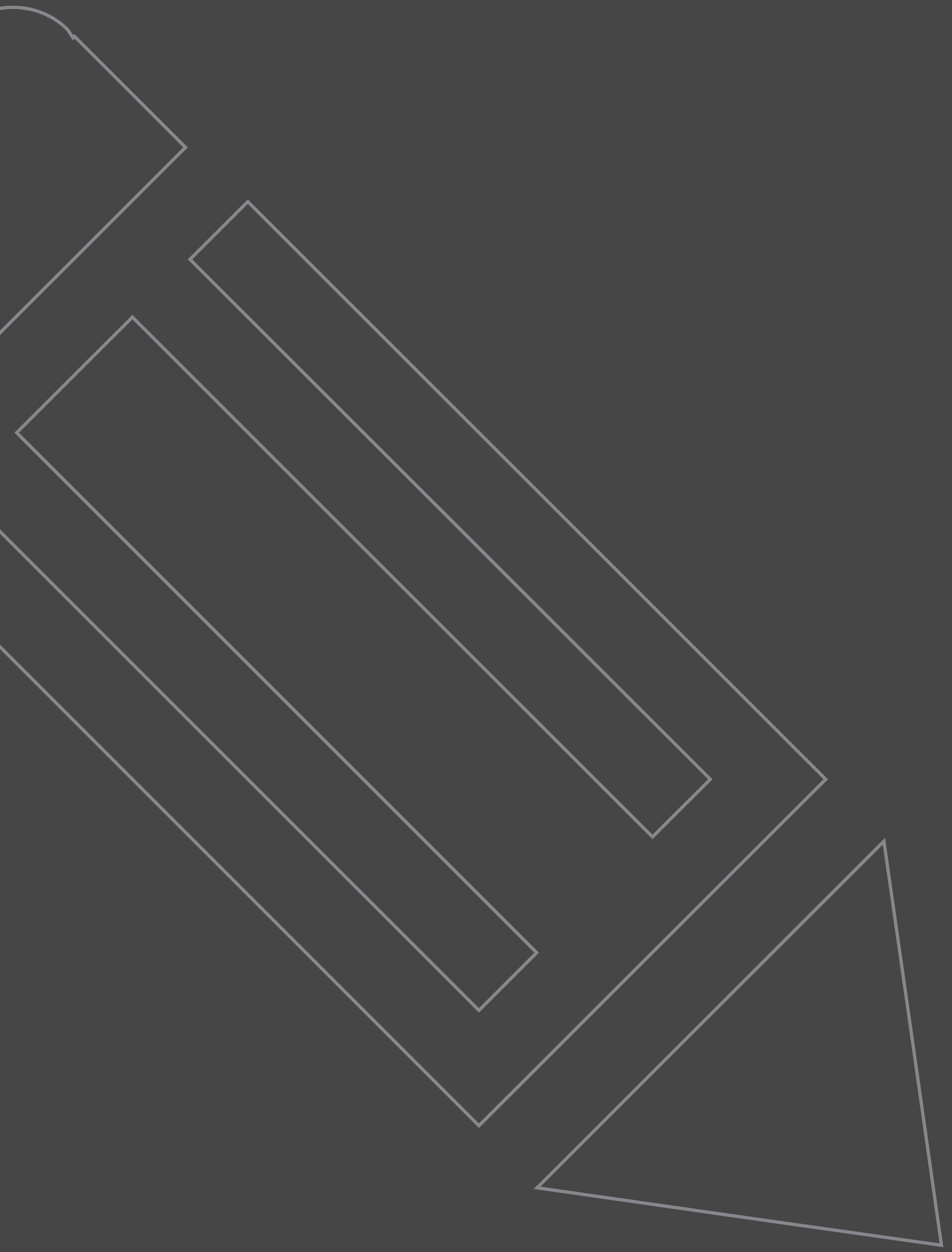
@prpatel

```
{
/// package.json
  "scripts": {
    "start": "webpack-dev-server",
    "solution": "webpack-dev-server —config webpack.solution.co
  }
}
```

@prpatel

```
module.exports = {
  entry: {
    app: ['./app.js']
  },
  output: {
    path: path.resolve(__dirname, 'build'),
    publicPath: '/',
    filename: 'bundle.js'
  },
  devServer: {
    historyApiFallback: true,
    hot: false,
    inline: true,
    progress: true,
  },
  module: {
    loaders: [
      {
        test: /\.js/,
        exclude: /node_modules/,
        loaders: [
          'babel-loader?optional[]=runtime&stage=0'
        ]
  …
```

@prpatel



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Lab1</title>
</head>
<body>
  <div id="root"></div>
  <script src="bundle.js"></script>
</body>
</html>
```

@prpatel

- ES6
- HTML file
- NPM Modules for browser deps!

@prpatel

ALERT!

TAGS IN RENDER MUST BE WRAPPED WITHIN A SINGLE ROOT TAG

Lab 2: Using NPM Libs

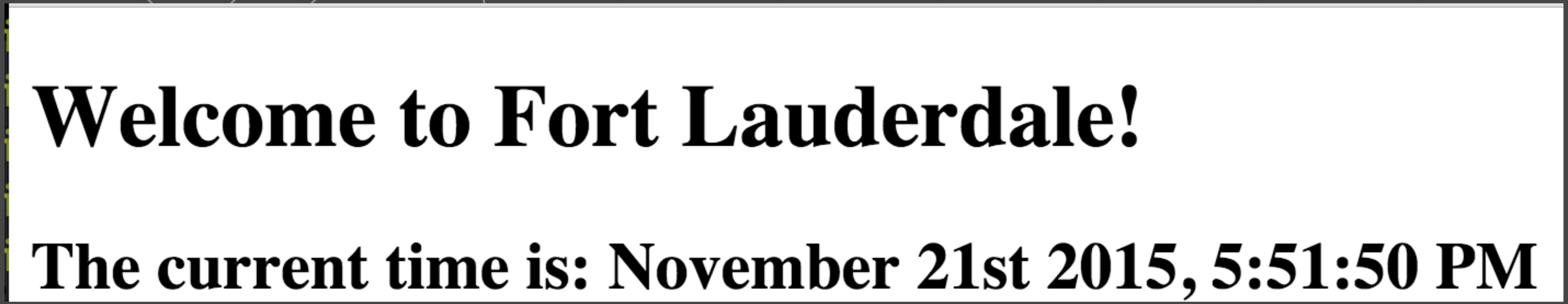
ALERT!

FOR OUR WORKSHOP PROJECT, REMEMBER NEW LIBS MUST BE ADDED AT BASE LEVEL!

@prpatel

Lab 2: NPM Libs

- Add moment.js library!
  - (at root of project) npm install moment -save
- cd Lab2; npm start
- open http://localhost:8080/
- http://momentjs.com/
- Edit app.js to make nicely formatted date, using moment.js like this:



Welcome to Fort Lauderdale!

The current time is: November 21st 2015, 5:51:50 PM

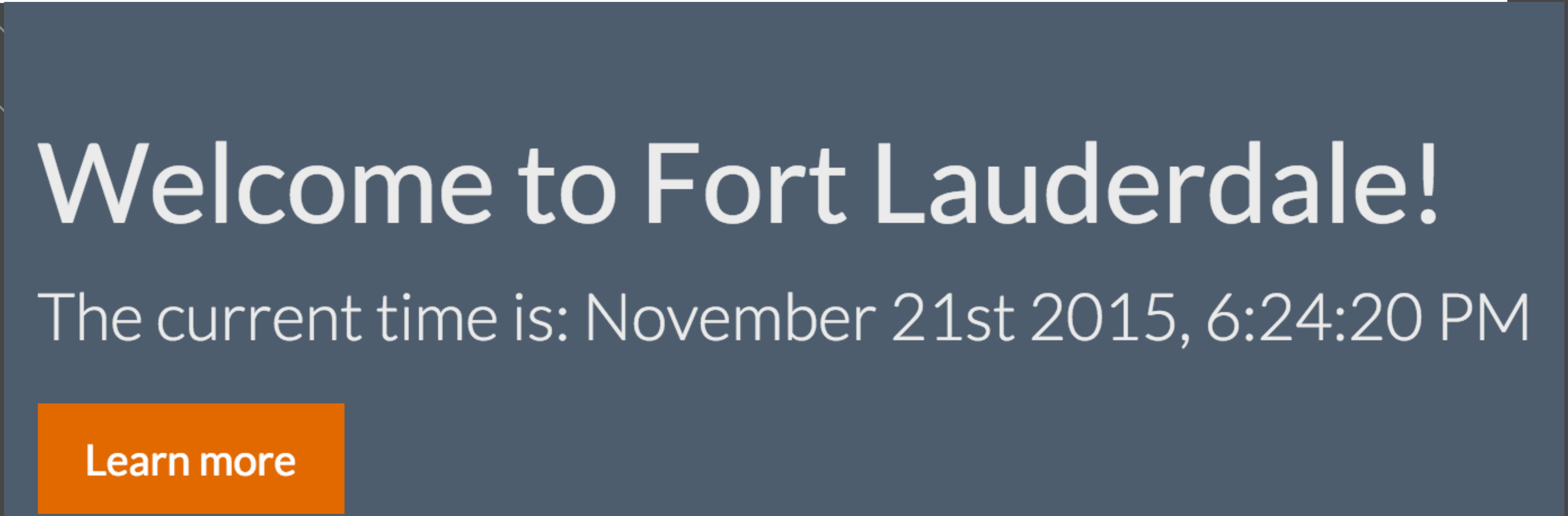
@prpatel

Lab 3: Bling with pre-made components

@prpatel

Lab 3: react-bootstrap

* react-bootstrap is already added as an npm dep
* Start up Lab3
* Edit app.js and use react-bootstrap components Panel and Jumbotron to make below
* import Jumbotron from 'react-bootstrap/lib/Jumbotron';
* http://react-bootstrap.github.io/components.html



Welcome to Fort Lauderdale!

The current time is: November 21st 2015, 6:24:20 PM

@prpatel

PROPERTIES

- PROPS USED TO PASS DATA 'DOWNSTREAM'
- ACCESS USING THIS.PROPS
- 2: CAN ALSO PASS FUNCTIONS

@prpatel

DYNAMIC COMPONENTS: attributes

@prpatel

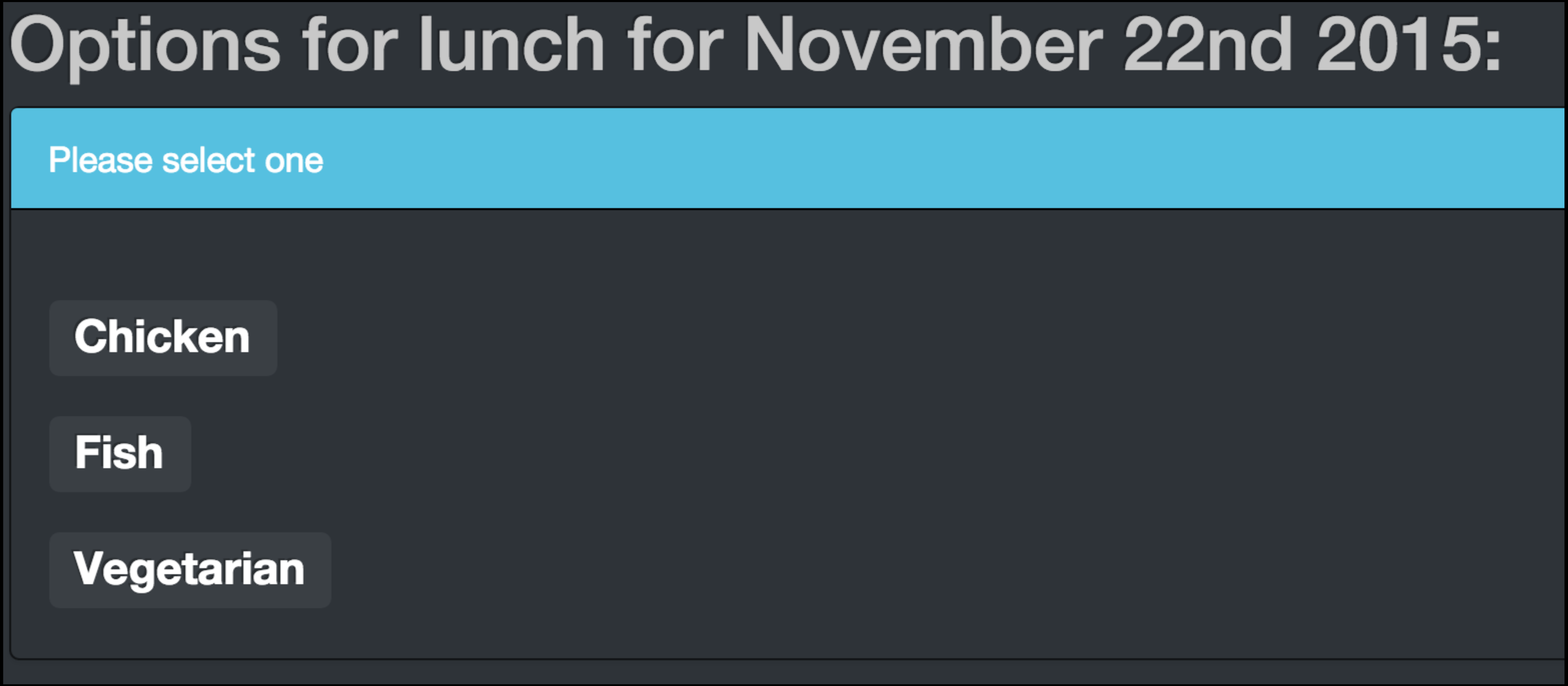
```
var MyComponent = React.createClass({
    render: function(){
        return (<h1>Hello, {this.props.name}!</h1>);
    }
});

React.renderComponent(<MyComponent
name="Pratik" />,
document.getElementById('myDiv'));
```

@prpatel

Lab 4: Static to Dynamic

@prpatel


Options for lunch for November 22nd 2015:

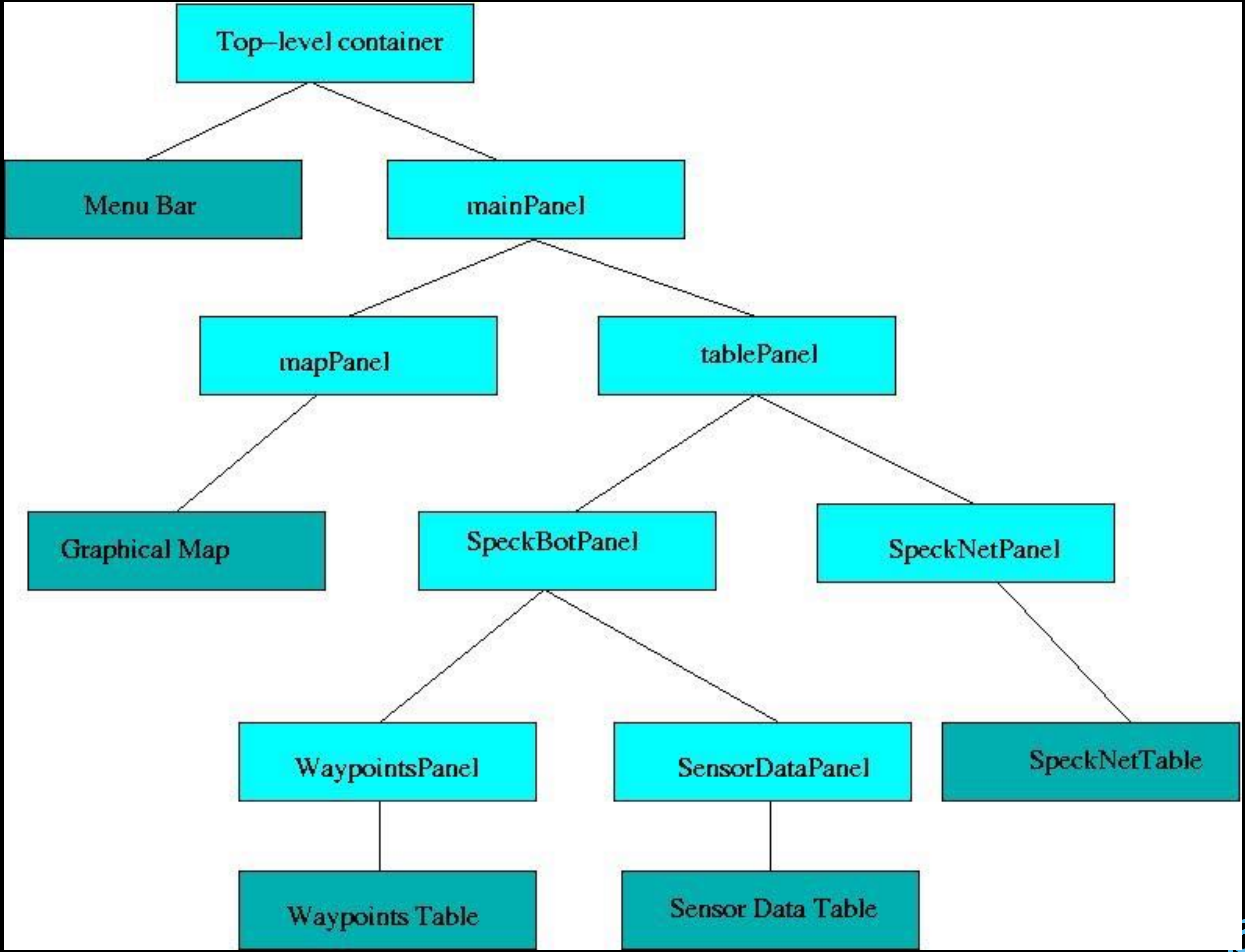
Please select one

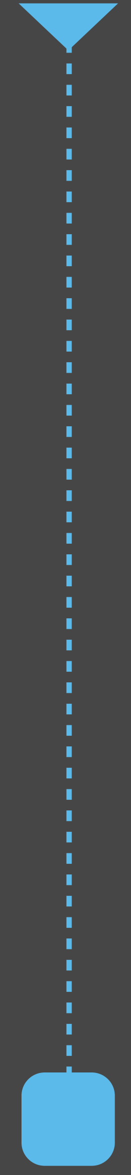
@prpatel

Lab 4: Data with props

- Pass in an array to render function:
- *var lunchChoices = ['Chicken', 'Fish', ‘Vegetarian’];*
- *<HelloWorld lunchChoices={lunchChoices}/>*
- Access data using this.props and create Labels by iterating over the array. Hint:
- let lunchOptions = this.props.lunchChoices.map(function(c) {
-         return <h2><Label>{c}</Label></h2> });
- Hint 2: Inside of return block, use {lunchOptions} to show this!
- Create a NEW component called LunchOptionsPanel, include inside LunchApp - pass data down and show lunch options inside of it

@prpatel


Top–level container
Menu Bar
mainPanel
mapPanel
tablePanel
Graphical Map
SpeckBotPanel
SpeckNetPanel
WaypointsPanel
SensorDataPanel
SpeckNetTable
Waypoints Table
Sensor Data Table



COMPONENTS
HAVE STATE

@prpatel

```
var MyComponent = React.createClass({
    getInitialState: function(){
        return {            count: 5        }
    },
    render: function(){
        return (
            <h1>{this.state.count}</h1>
        )
    }
});
```

@prpatel

Changing the state causes a refresh: this.setState(…)

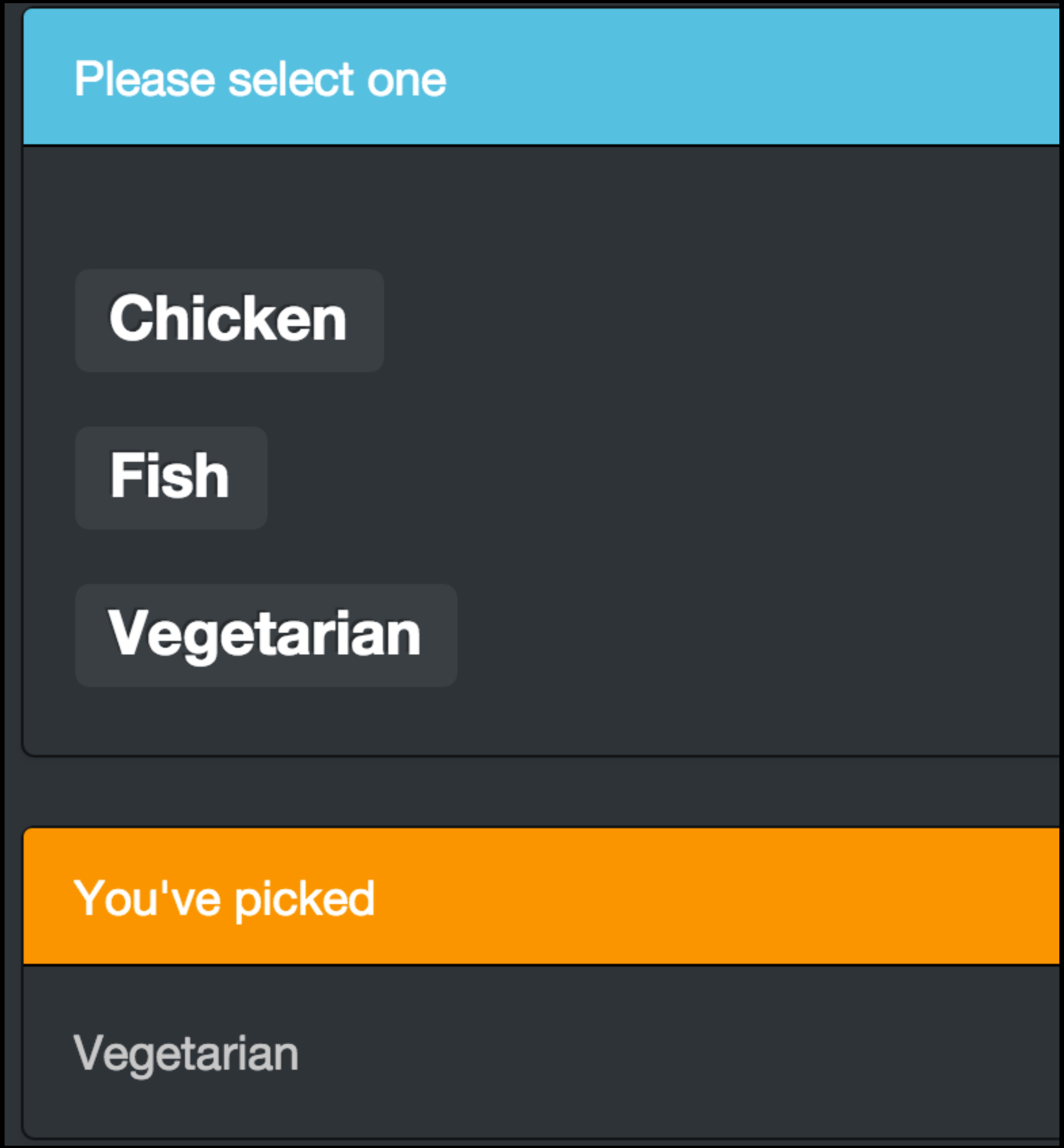
@prpatel

setState(...)
invokes render for
component and all
sub components!

@prpatel

Lab 5: STATE

@prpatel


Please select one

Chicken

Fish

Vegetarian

You've picked

Vegetarian

@prpatel

ALERT!

AS WE'RE USING ES6 / V0.14, GETINITIALSTATE IS DEPRECATED, USE A SIMPLE INSTANCE PROPERTY

@prpatel

```
class ExampleComponent extends React.Component {
 getInitialState() {
  return Store.getState();
 }
 constructor() {
  super();
  this. _handleClick = this. _handleClick.bind(this);
 }
}
```

After with ES6:

```
class ExampleComponent extends React.Component {
 constructor() {
  super();
  this. _handleClick = this. _handleClick.bind(this);
  this.state = {something: ‘hello’}
 }
}
```

@prpatel

ALERT!

REACT'S CREATECLASS FUNCTIONALITY AUTOMATICALLY BOUND YOUR METHODS TO A COMPONENT INSTANCE. THIS MEANT THAT WITHIN A CLICK CALLBACK `THIS` WOULD BE BOUND TO THE COMPONENT. WITH ES6 CLASSES, WE MUST HANDLE THIS BINDING. WE PREBIND IN THE CONSTRUCTOR.

```
class ExampleComponent extends React.Component {
 constructor() {
  super();
  this. _handleClick = this. _handleClick.bind(this);
 }
 render() {
  return <div onClick={this._handleClick}>Hello, world.</div>;
 }
 _handleClick() {
  console.log(this); // this is an ExampleComponent
 }
}
```

@prpatel

```
var Counter = React.createClass({
  incrementCount: function(){
    this.setState({
      count: this.state.count + 1
    });
  },
  getInitialState: function(){
     return {     count: 0     }
  },
```

@prpatel

```
  render: function(){
    return (
      <div class="my-component">
        <h1>Count: {this.state.count}</h1>
        <button type="button"
onClick={this.incrementCount}>
Increment</button></div>    );  }
});
```

@prpatel

Lab 5: STATE

* When user clicks an option, change the display of SelectedLunchPanel (new component) to show it!
* **In LunchOptionsPanel add:**
* constructor(props) {
*   super(props);
*   this.state = {selectedLunch: 'Nothing selected'};
*   this.handleClick = this.handleClick.bind(this);
*   }
* **Then add a handleClick function that does this.setState({}) with new selectedLunch value**
* **getting the clicked item in handleclick(event) -> event.target.textContent**
* **Inside of render, add:**
* let clickHandler = this.handleClick;
* onClick={clickHandler}
* **Pass the state down to child:**
* <SelectedLunchPanel selectedLunch={this.state.selectedLunch}>
* atom package for linting: linter-eslint

@prpatel

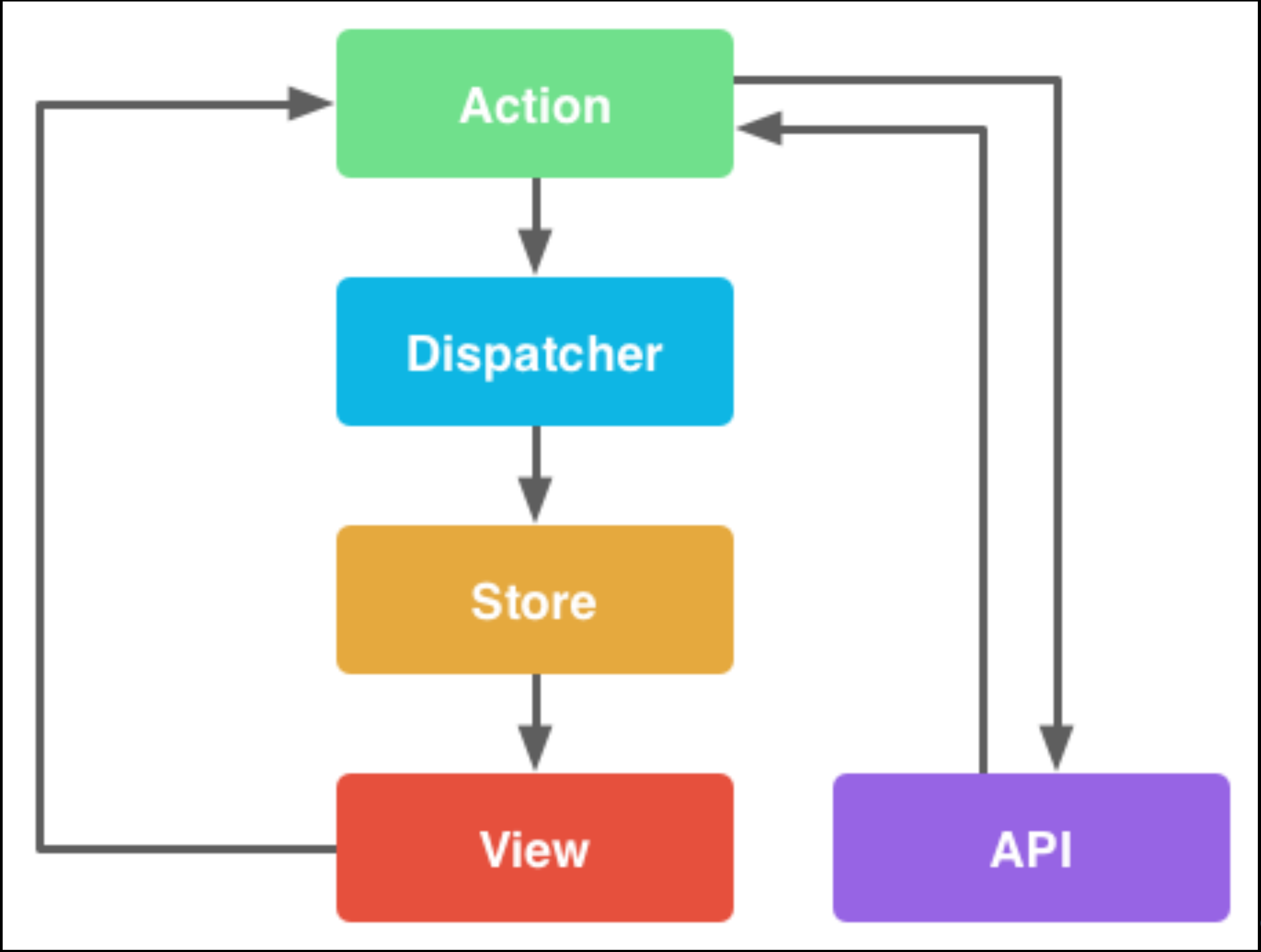
Lab 5: Discussion

@prpatel

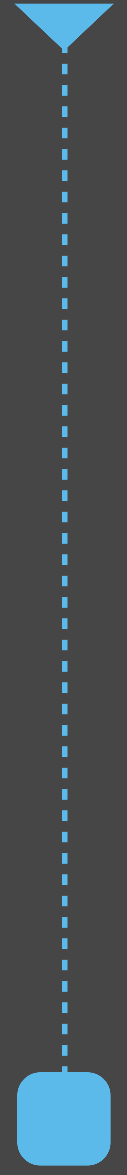
- Within component state is mutable, props aren’t
- setState == refresh
- The child comp get updated or destroyed?

@prpatel

ONE-WAY DATA FLOW



@prpatel



EVENTS ARE WRAPPED!

- SyntheticEvent
- cross-browser wrapper
- wrap native event

@prpatel

Lab 6: Component Lifecycle

@prpatel

Lab 6: Lifecyle

* Add these lifecycle functions to the SelectedLunchPanel & LunchOptionsPanel components; insert a console.log. Observe what happens as the app starts (use browser reload button if needed), and also what happens when you click the lunch options.
* Quickly read “Lifecycle Methods": https://facebook.github.io/react/docs/component-specs.html
  * componentWillMount()
  * componentDidMount()
  * componentWillReceiveProps(nextProps)
  * componentWillUpdate(nextProps, nextState)
  * componentDidUpdate(prevProps, prevState)
  * componentWillUnmount()
  * Hint:

```
*   componentWillMount() {
*       console.log('componentName -> componentWillMount')
*     }
```

@prpatel

COMPONENT SPECIFICATIONS

@prpatel

- getInitalState
- getDefaultProps
- mixins

componentWillRender

componentDidMount

componentWillUnmount

@prpatel

ALERT!

AS YOU SEE, SINCE WE JUST UPDATED THE CONTENTS OF THE SUB-COMP, IT DID NOT GET DESTROYED!

@prpatel

ANIMATIONS

@prpatel

- Never directly update the DOM
- Including CSS
- Remember Virtual DOM is our “buffer”

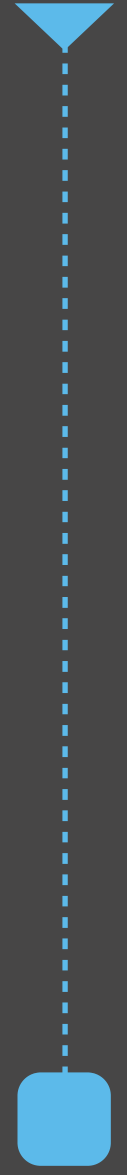
@prpatel

Lab 7: Animations

@prpatel

Lab 7: Animations

* We’ve already added a new css file: assets/styles.css
* Going to use a transition group to “slide in” the selected lunch
* go into styles.css and figure out and play with code to do fix animation
* Apply a 3D CSS3 animation, reference: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Transforms/Using_CSS_transforms
* https://facebook.github.io/react/docs/animation.html



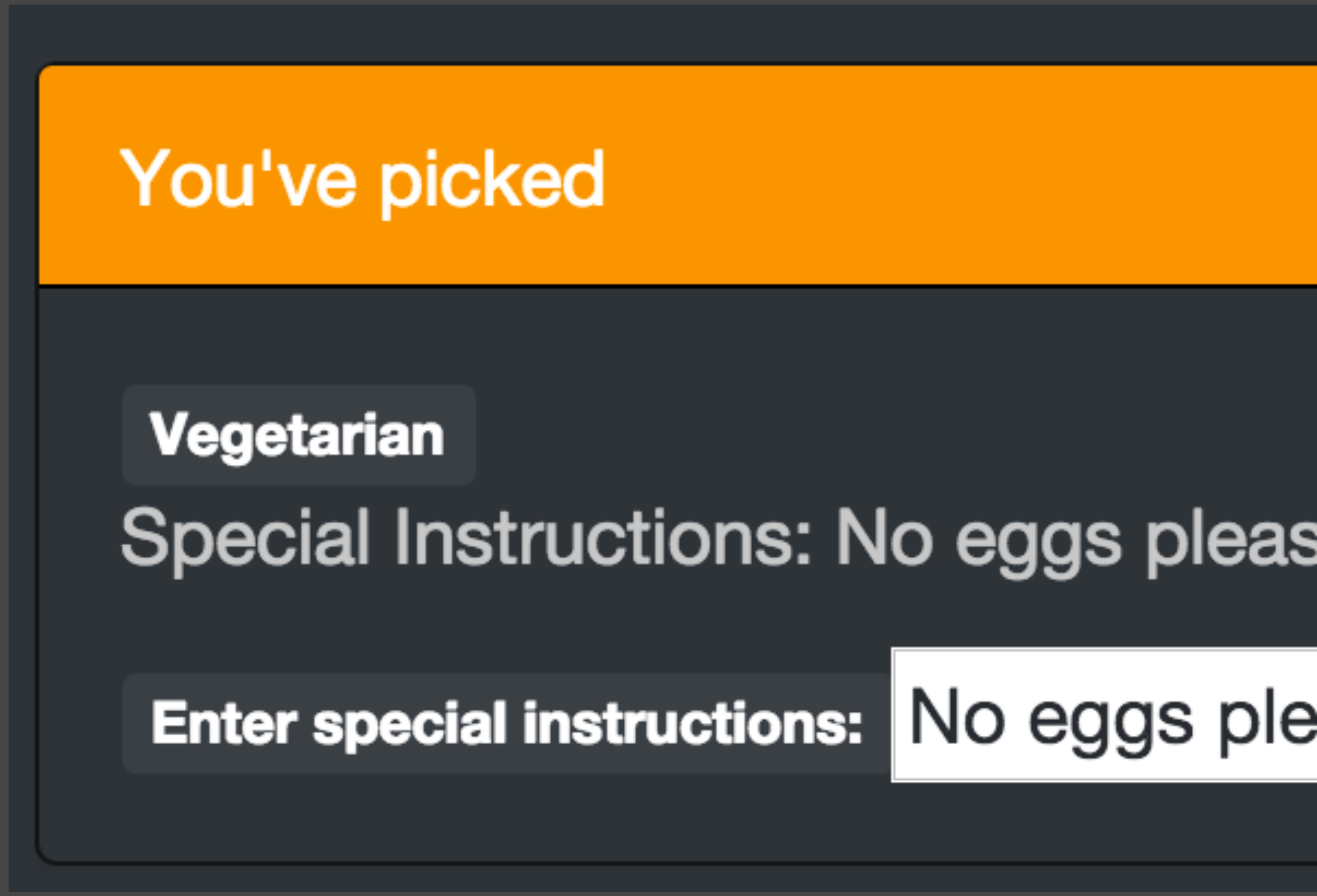
What happens

@prpatel

Lab 8-10: Component Communication

Lab 8: Child-Parent Comms

* **Create a new component SpecialInstructionsInput with an Input element:**
* <input ref='specialInstructionsInput' type='text' value={this.props.value} onChange={this.handleChange} />
* **Add state into SelectedLunchPanel; update it with a function, pass this state AND updateInstructions FUNCTION down to the SpecialInstructionsInput component:**
* updateInstructions(instructions) {
*   this.setState({instructions: instructions});
*   }
* **Show this state, this.state.instructions, in SelectedLunchPanel:**
* <p>Special Instructions: {this.state.instructions}</p>
* **In SpecialInstructionsInput, have this.handleChange call the passed in updateInstructions to update the PARENT**
*   handleChange(e) { e.target.value ....


You've picked

Vegetarian

Special Instructions: No eggs pleas

Enter special instructions: No eggs ple

@prpatel

Lab 9: Refs

* **Create a submit button in SpecialInstructionsInput to “submit” the instructions**
* **The handleChange should use “ref” to get the data out of the input element!**
* **Don’t forget to import the Bootstrap button!**
* import Button from ‘react-bootstrap/lib/Button’;

Hints:

<input ref=‘specialInstructionsInput'

this.refs.specialInstructionsInput.value

@prpatel

Refs should be considered PRIVATE to the component

Use only within a component! (dont' reach into other comps)

@prpatel

Lab 10: Dispatcher

* **Create a simple dispatcher, as an ES6 module, and wire it up into the example! Put in its own file, call it: my-dispatcher.js**

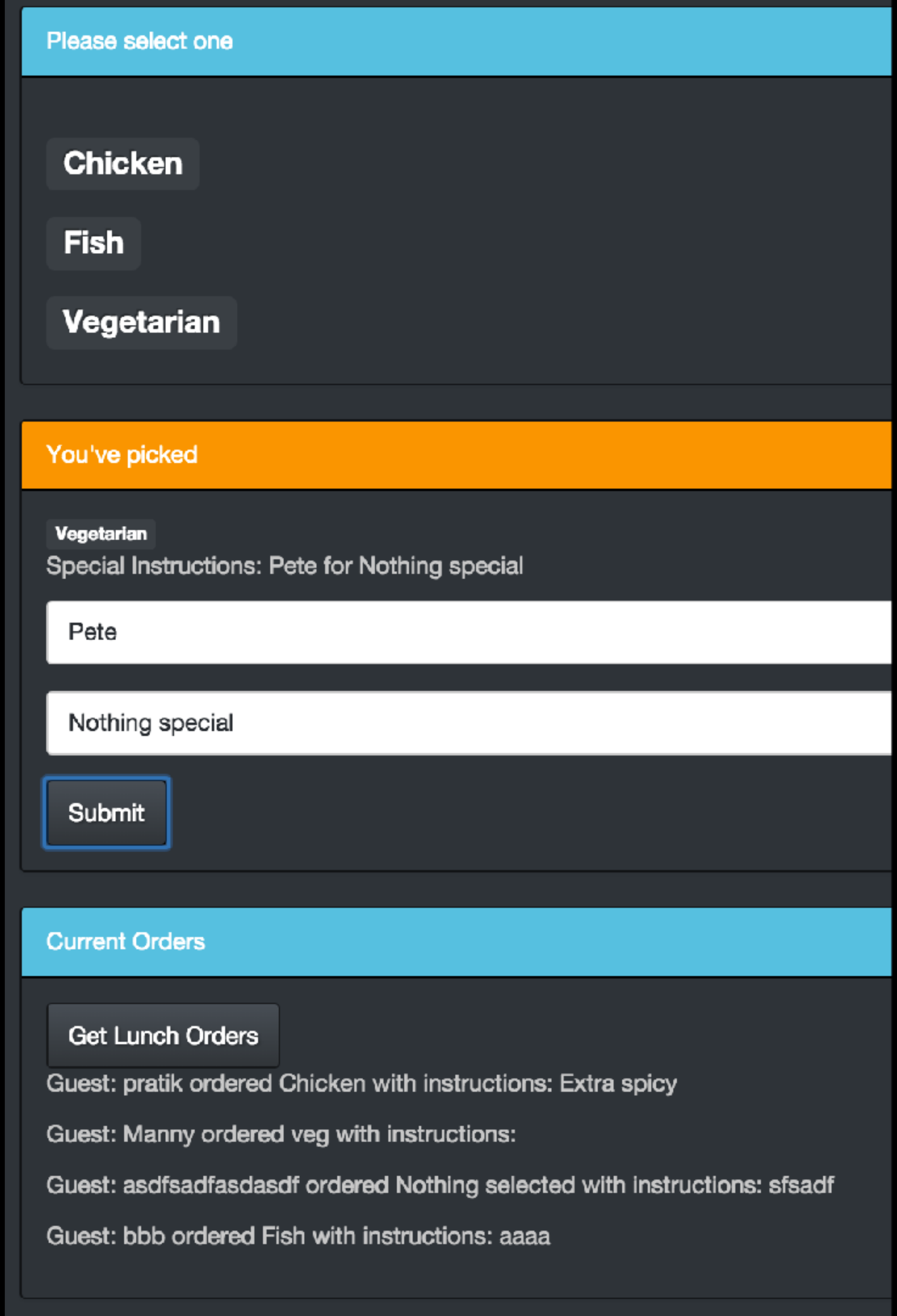
```
var registeredCallback;
exports.on = function(channel, callback) {
*          registeredCallback = callback; };
*  exports.trigger = function(channel, data) {
*          registeredCallback(channel, data) };
*  exports.removeCallback = function() {
*          registeredCallback = null; };

*  Hints:
*  Register with the dispatcher using:
   *  componentDidMount() { …
*  Registering with dispatcher:
   *  dispatcher.on('updateInstructionsDispatch', this.updateInstructions);
*  Firing dispatcher:
   *  dispatcher.trigger(‘updateInstructionsDispatch', somedata)
```

@prpatel

Lab 11: Network + REST + App Composition

@prpatel


Please select one

Chicken

Fish

Vegetarian

You've picked

Vegetarian

Special Instructions: Pete for Nothing special

Pete

Nothing special

Submit

Current Orders

Get Lunch Orders

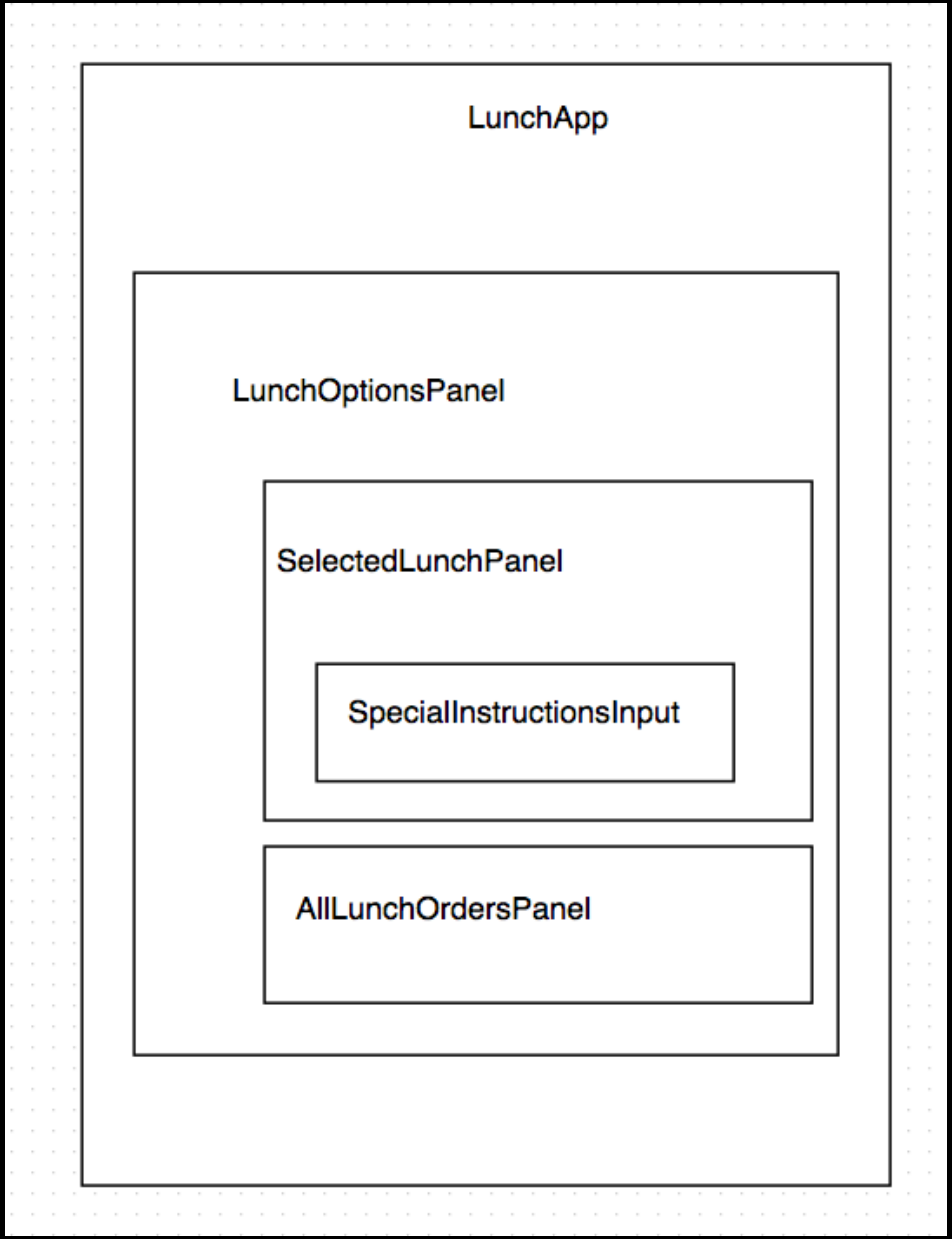
Guest: pratik ordered Chicken with instructions: Extra spicy

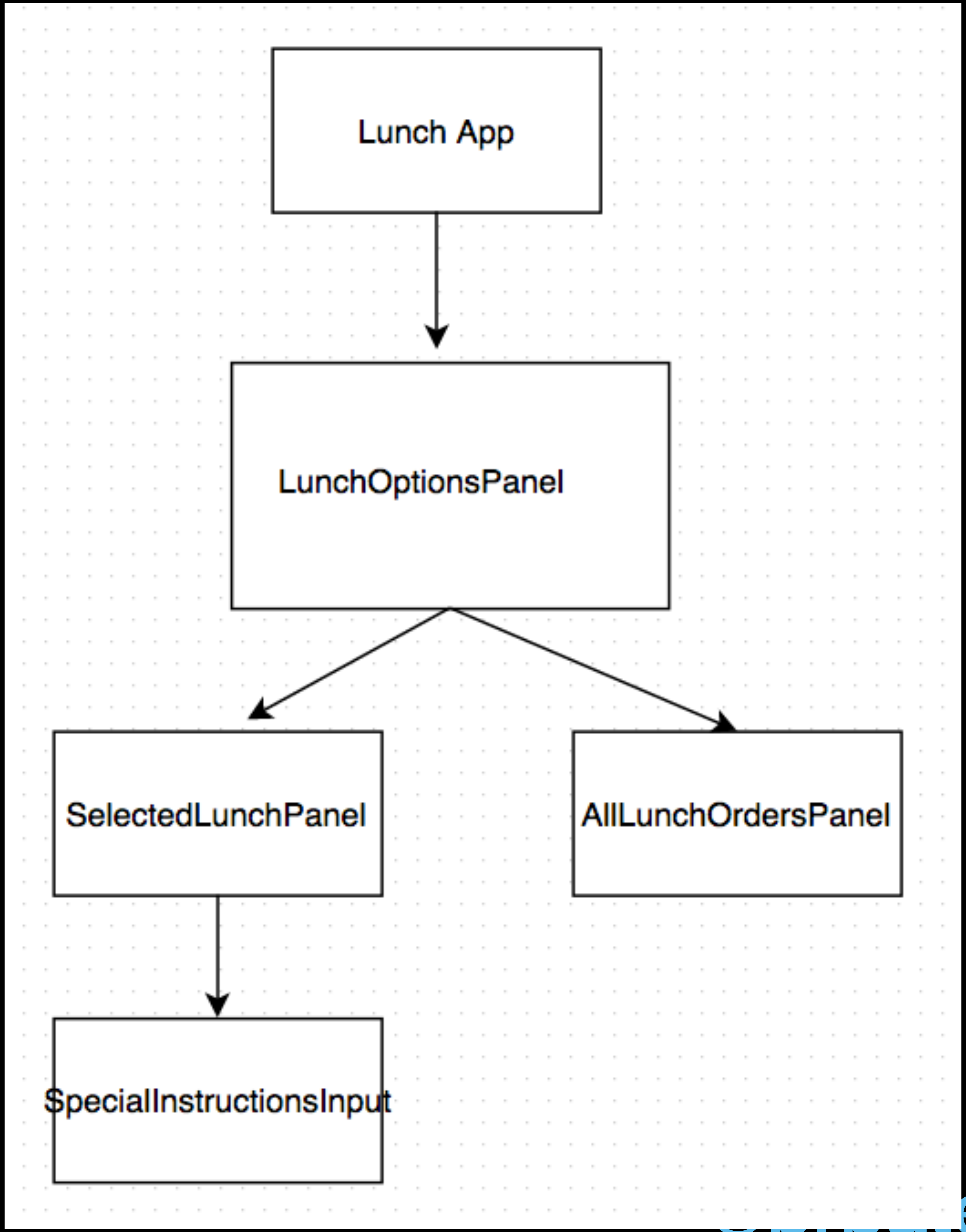
Guest: Manny ordered veg with instructions:

Guest: asdfsadfasdasdf ordered Nothing selected with instructions: sfsadf

Guest: bbb ordered Fish with instructions: aaaa

@prpatel


LunchApp
LunchOptionsPanel
SelectedLunchPanel
SpecialInstructionsInput
AllLunchOrdersPanel


Lunch App
LunchOptionsPanel
SelectedLunchPanel
AllLunchOrdersPanel
SpecialInstructionsInput

Lab 11: Network

* Use the Axios library to save and pull lunch data
* Create a new input called 'guestName' in SpecialInstructionsInput component
* Send name, selection, instruction data to REST server; var endpoint = '/lunches';
* Create a button called Get Lunch Orders, show fetched lunch data in our React app in new component named AllLunchOrdersPanel
    * Where do we put saveData function if we need the selection, name, and instructions?
    * Passing data to AllLunchOrdersPanel from LunchOptionsPanel.
    * Use the property passing style to pass function down to LunchOptionsPanel

```
axios.get('/lunches')
 .then(function (response) {
  console.log(response);
 })
 .catch(function (response) {
  console.log(response);
 });
```

```
axios.post(endpoint, {
      name: name,
    lunch: selection,
 instructions: instructions
          })
         .then(
```

WARNING: WHEN POSTING OR READING PLEASE USE THIS STRUCT

```
{    name: name,
     lunch: selection,
     instructions: instructions  }
```

@prpatel

WON'T THAT BE SUPER SLOW?

@prpatel

NO,
NOT IN
REACT

REMEMBER THE VIRTUAL DOM AND THE COOL STUFF IT DOES!

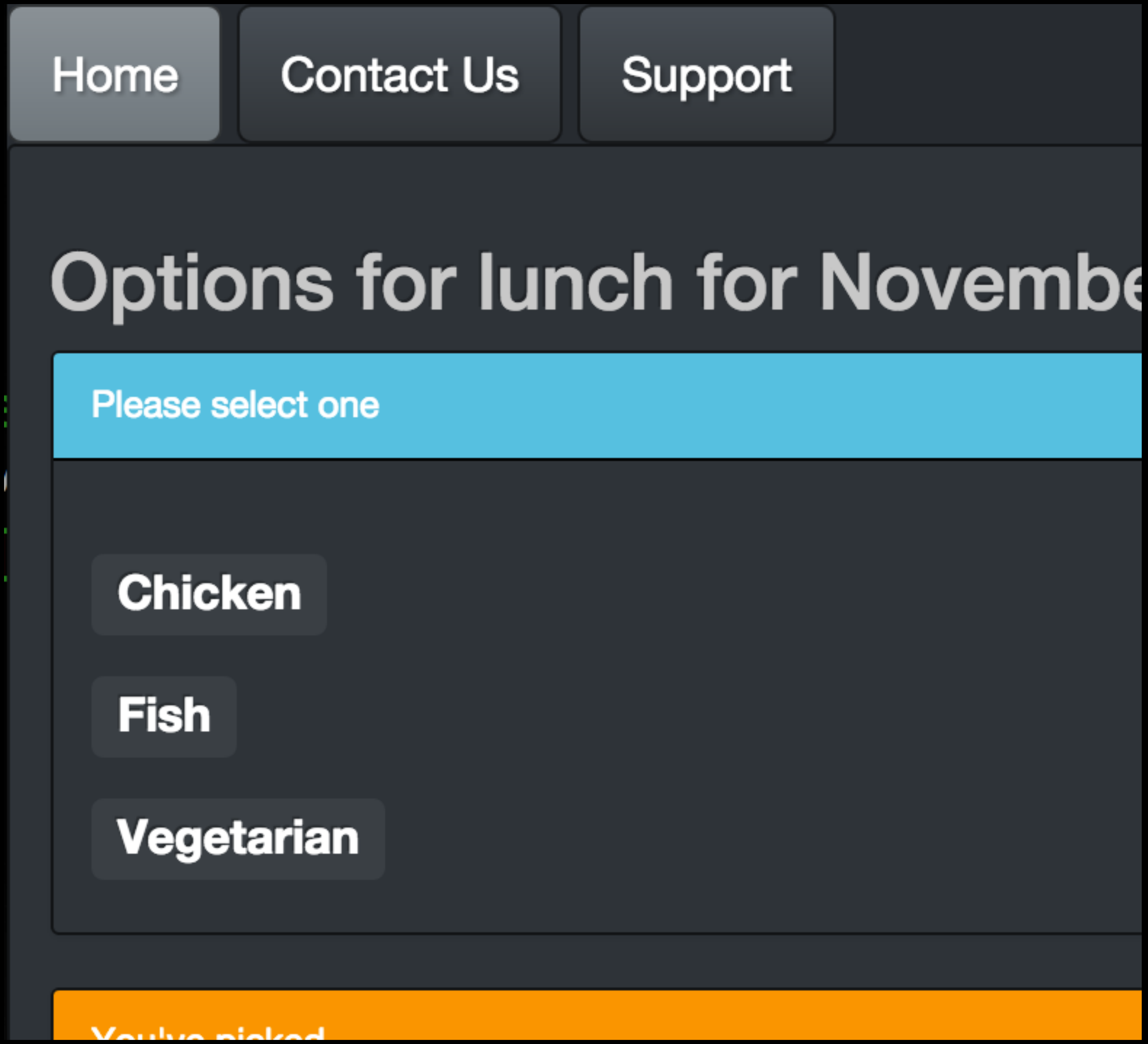
@prpatel

- VIRTUAL DOM
- SMART DIFF'ING OF DOM
- BATCHED DOM UPDATES

@prpatel

Lab 12: React Router

@prpatel


Home
Contact Us
Support

Options for lunch for Novembe

Please select one

Chicken

Fish

Vegetarian

@prpatel

Lab 12: Router

* import ButtonToolbar from 'react-bootstrap/lib/ButtonToolbar';
* import { Router, Route, Link, IndexRoute } from ‘react-router';
* Will need to create a new component: LunchAppPanel, that has contents of current LunchApp component. Also create stub Support and Contact React components
* <Router>
*   <Route path="/" component={LunchApp}>
*   <IndexRoute component={LunchAppPanel} lunchChoices={lunchChoices} />
*   <Route path="support" component={Support}/> …
* In LunchApp, use React-Bootstrap ButtonToolbar and Buttons, then create Links, and LAST LINE VERY IMPORTANT:

```
<Link to={‘/'}>Home</Link>
<Link to={'contact'}>Contact Us</Link>
<Link to={'support'}>Support</Link>
{this.props.children}
```

@prpatel

TESTING REACT APPS

@prpatel

MYRID OF CHOICES

- Wing it / no tests
- Shallow Renderer
- Render and test HTML

@prpatel

- Discrete Components easy to test
- Need tooling for JSX / ES6
- Mocks

@prpatel

Lab 13: Tests

* go to ROOT/test
* run:
* npm test
* Explore the test case under test/, and the source file under Lab13/
* Add a CSS class to the input element in the source (remember in React you have to use "className" - then write a test case to verify it made it into the rendered DOM

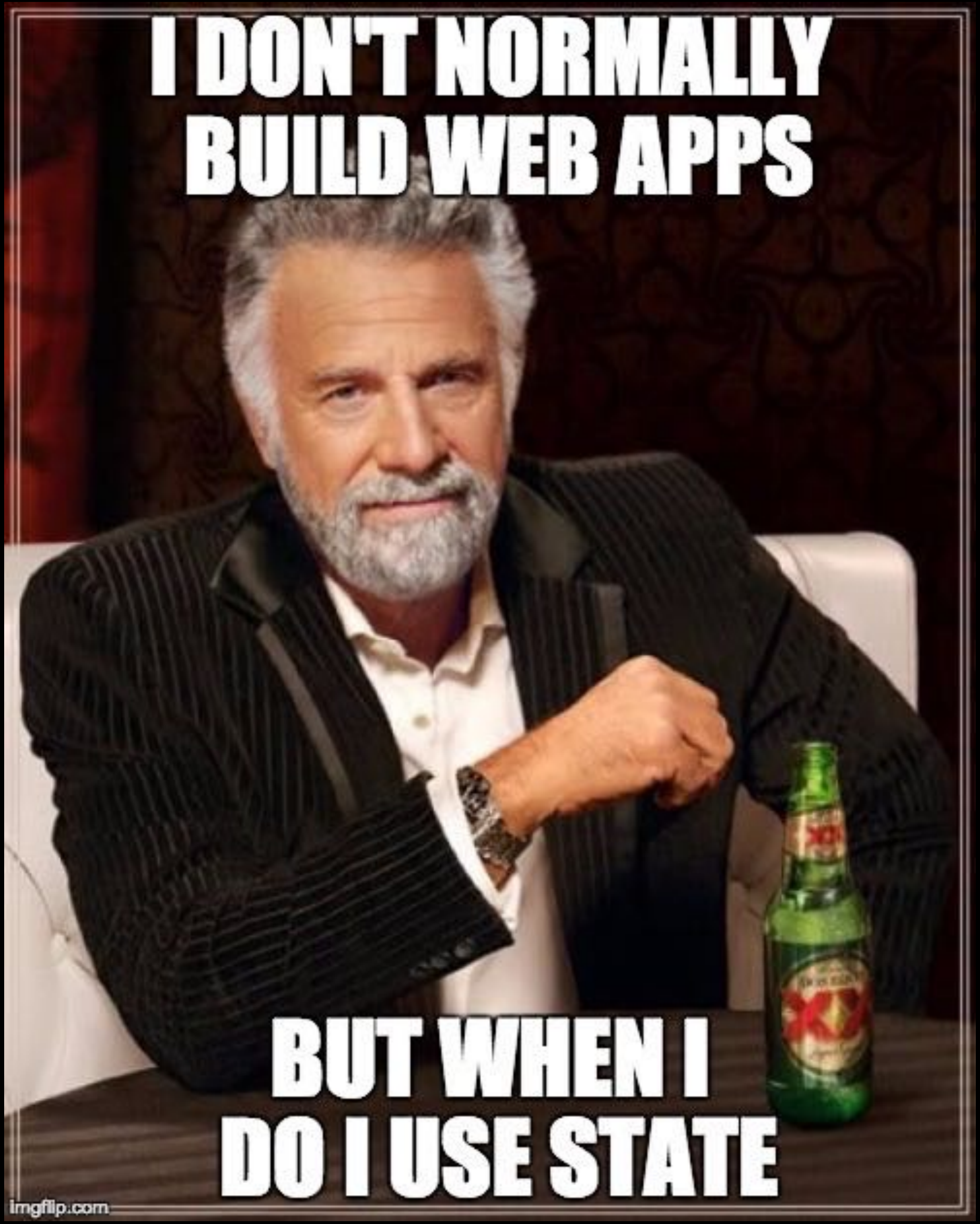
Lab 14: THE WHOLE BURRITO

@prpatel

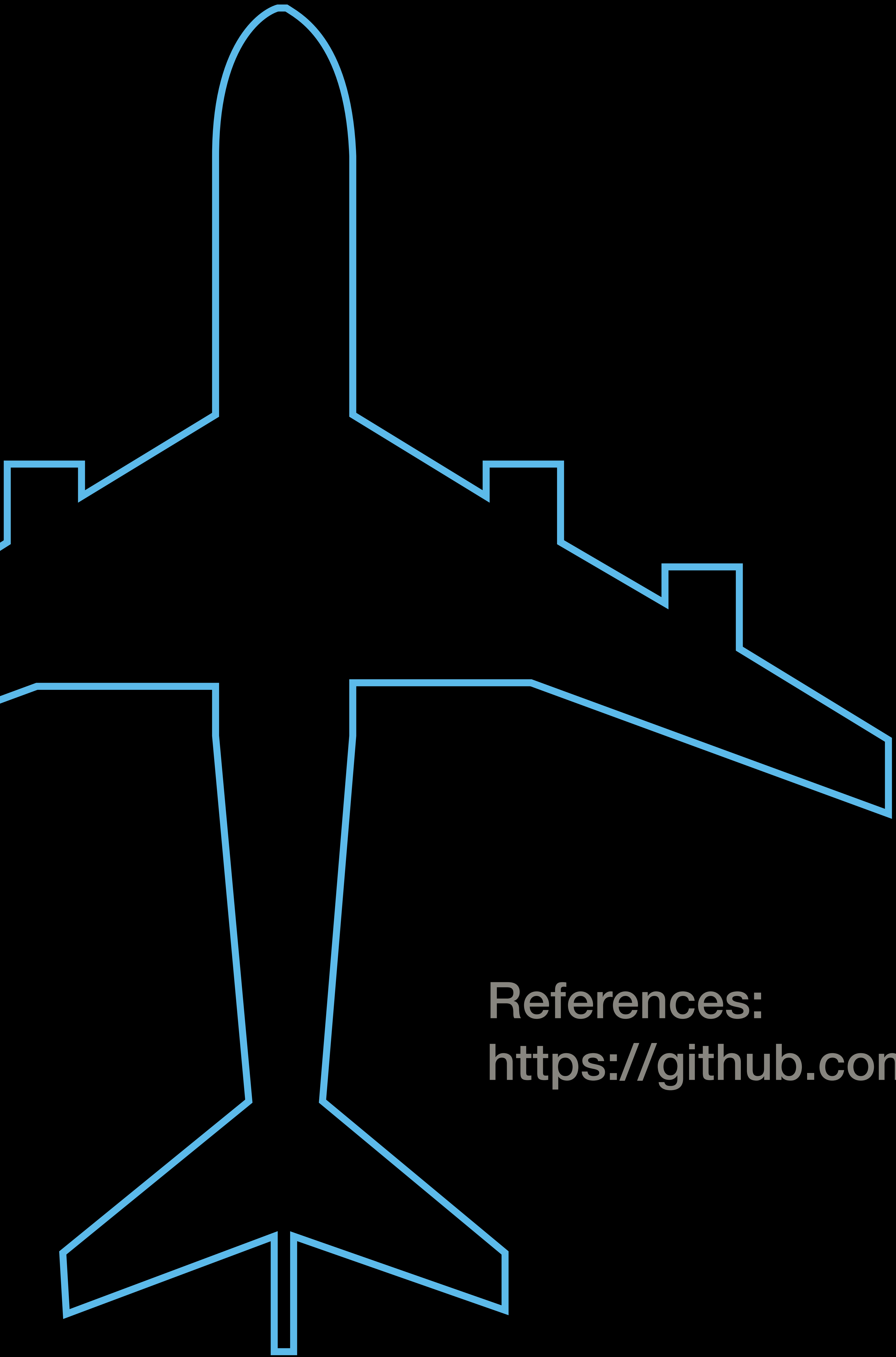
Lab 14: Full Project Structure

* Pull down this repo:
* https://github.com/davezuko/react-redux-starter-kit
* Run and experiment with the source:
* npm start

@prpatel


I DON'T NORMALLY BUILD WEB APPS
BUT WHEN I DO I USE STATE
imgflip.com

@prpatel



THANK YOU

References:
https://github.com/davezuko/react-redux-starter-kit

@prpatel