Start MongoDB.

Open a DOS command prompt window.

cd C:\Users\{your\_username}\sandbox

git clone <https://github.com/genma123/nodeJSBootcampReactWorkshop.git>

cd nodeJSBootcampReactWorkshop

npm install

node server (your REST API is now up and running)

Open a 2nd DOS command prompt window.

cd C:\Users\{your\_username}\sandbox\nodeJSBootcampReactWorkshop

npm install -g create-react-app

set PATH=%PATH%;%APPDATA%\npm

create-react-app client\_react

Be patient!

cd client\_react

*All the following can be found in the Step 1 folder, if you want to avoid/minimize typing!*

In public\index.html, change line 16:

<title>My TaskList</title>

npm install bootstrap

In src\index.js, add the following lines after line 2:

import 'bootstrap/dist/css/bootstrap.css';

import 'bootstrap/dist/css/bootstrap-theme.css';

In src\App.css, add the following lines after line 20:

h1 {

color: #369;

font-family: Arial, Helvetica, sans-serif;

font-size: 250%;

}

Modify src\App.js to look like this:

import React, { Component } from 'react';

// import logo from './logo.svg';

import './App.css';

class App extends Component {

render() {

  return (

    <div className="container">

      <h1>MyTaskList</h1>

      <hr/>

      <form className="well">

       <div className="form-group">

         <input type="text" name="title" className="form-control" placeholder="Add Task..."/>

       </div>

      </form>

    </div> );

}

}

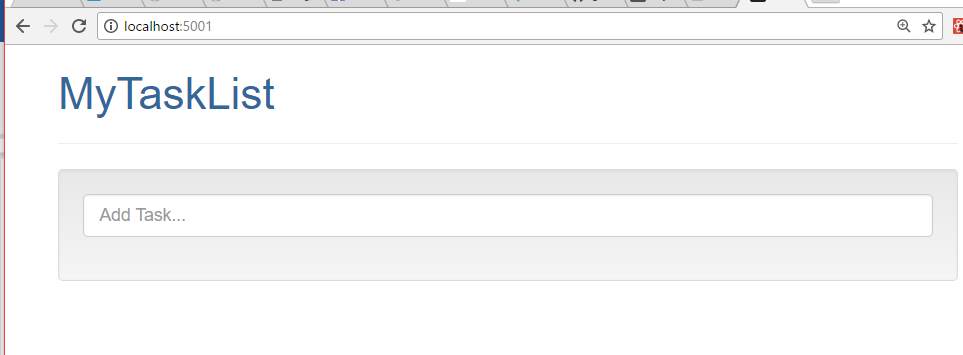
export default App;

Create .env in root of client\_react:

PORT=5001

npm run start

Launch Chrome (not IE!) Navigate to <http://localhost:5001/>. You should see:



*Note: this browser window may launch automatically. If IE is the default browser, it will come up in IE. If it comes up in IE, close IE and open it in Chrome, because the following steps won’t work in IE.*

*Step 1 complete!*

*All the following can be found in the Step 2 folder, if you want to avoid/minimize typing!*

Open a DOS command prompt window and navigate to C:\sandbox\final. Launch the Angular version of the Task List using node server. Open a new Chrome (or IE11?) tab or window and navigate to <http://localhost:3000/>. If there are not already to-do items, add some (or add some new ones if so inspired). In Step 2, we will add the ability to view tasks, but not yet to add them.

Modify package.json in the root of the client\_react folder as shown (add the “proxy” at line 5):

{

"name": "client\_react",

"version": "0.1.0",

"private": true,

**"proxy": "http://localhost:5000/",**

"devDependencies": {

"react-scripts": "0.9.5"

},

"dependencies": {

"react": "^15.5.4",

"react-dom": "^15.5.4"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test --env=jsdom",

"eject": "react-scripts eject"

}

}

May need to restart the Webpack Dev Server with “^C-npm run start”.

Copy Client.js from the Step 2 folder to the src folder in your workspace. There is nothing specific to React in this script, but please feel free to take a look! NOTE: this will NOT work with IE11!

Create (or copy from the Step 2 folder) src\Task.css in in your workspace (VSC should still be open):

.Task {

text-align: left;

}

.Task-title {

    display: inline;

}

Create src\Task.js in your workspace:

import React, { Component } from 'react';

import './Task.css';

class Task extends Component {

    constructor() {

        super();

    }

    render() {

        return (<span className="Task"><div className="col-md-1">

<input name="select" type="checkbox" checked={this.props.selected} />

</div>

<div className="col-md-7">

{this.props.title}

</div>

<div className="col-md-4">

<input type="button" value="Delete" className="btn btn-danger"/>

</div>

<br/><br/></span>);

    }

}

export default Task;

Create src\TaskList.css in your workspace:

.TaskList {

text-align: center;

}

.TaskList-image {

    width: 100px;

    height: 75px;

}

Create src\TaskList.js in your workspace:

import React, { Component } from 'react'; // React must be in scope when using JSX

import Task from './Task';

import './TaskList.css';

class TaskList extends Component {

    render() {

        // NOTE task.isDone can be undefined, so forcing the prop to be either true or false prevents

        // a "switch from uncontrolled to controlled" warning

        const tasks = this.props.tasks.map((task) =>

            <Task key={task.\_id} id={task.\_id} title={task.title} selected={task.isDone ? true : false} />);

        return (

            <div className="TaskList">

            {tasks}

            </div>

        );

    }

}

export default TaskList;

Modify src\App.js:

import React, { Component } from 'react';

// import logo from './logo.svg';

import './App.css';

**import TaskList from './TaskList';**

**import Client from './Client';**

class App extends Component {

**// NOTE this is the only "stateful" component.**

**constructor() {**

**super();**

**this.state = {**

**tasks: []**

**};**

**};**

**//Use of arrow functions keeps "this" in scope.**

**retrieveTasks() {**

**Client.retrieve((data) => this.setState({ tasks: data.sort() }));**

**}**

**componentWillMount() {**

**console.log("In componentWillMount");**

**this.retrieveTasks();**

**}**

render() {

return (

<div className="container">

<h1>MyTaskList</h1>

<hr/>

<form className="well">

<div className="form-group">

<input type="text" name="title" className="form-control" placeholder="Add Task..."/>

</div>

</form>

**<TaskList tasks={this.state.tasks} />**

</div> );

}

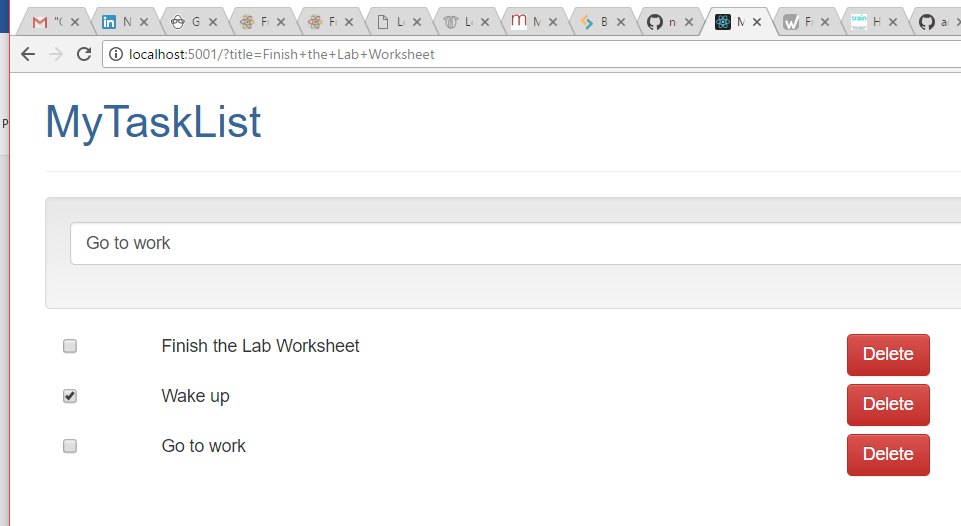
}

export default App;

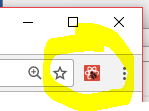
At this point, if Webpack is not started, start it with npm run start (from root of client\_react folder).

In Chrome, navigate to <http://localhost:5001/>.

If everything is working, and if there are tasks in Mongo, you should see something like this:



If React Developer Tools is not installed,



install by going to:

<https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en>

and following the instructions found there. Feel free to explore some of the functionality!

Also try intentionally introducing a syntax error on one of the files you created (assuming there aren’t some already) to see what happens (may need to reload the web page for full effect). What you are seeing is due to Webpack functionality.

*Step 2 complete!*

*All the following can be found in the Step 3 folder, if you want to avoid/minimize typing!*

Note: you may intermittently see errors when saving after each code change, please press forward!

Modify src\App.js as shown:

import React, { Component } from 'react';

// import logo from './logo.svg';

import './App.css';

import TaskList from './TaskList';

import Client from './Client';

**import \_ from 'lodash';**

class App extends Component {

  // NOTE this is the only "stateful" component.

constructor() {

super();

this.state = {

    tasks: []

};

**this.addTask = this.addTask.bind(this);**

**this.deleteTask = this.deleteTask.bind(this);**

**this.updateTask = this.updateTask.bind(this);**

};

//Use of arrow functions keeps "this" in scope.

retrieveTasks() {

   Client.retrieve((data) => this.setState({ tasks: data.sort() }));

}

**addTask(event) {**

**Client.add(event, (data) => this.setState({ tasks: this.state.tasks.concat([data]) }));**

**event.preventDefault();**

**}**

**updateTask(id, title, selected) {**

**var newSelected = !selected;**

**Client.update(id, title, newSelected, (data) => {**

**var tasks = this.state.tasks;**

**// console.log("found: " + JSON.stringify(\_(tasks).find({ "\_id": id })));**

**\_(tasks).find({ "\_id": id }).isDone = newSelected;**

**this.setState({ tasks: tasks });**

**});**

**}**

**deleteTask(id) {**

**Client.remove(id, (data) => {**

**this.setState({ tasks: \_.filter(this.state.tasks, function(t) { return t.\_id !== id; }) });**

**});**

**}**

componentWillMount() {

   console.log("In componentWillMount");

   this.retrieveTasks();

}

render() {

return (

<div className="container">

<h1>MyTaskList</h1>

<hr/>

<form className="well**" onSubmit={this.addTask}**

>

<div className="form-group">

<input type="text" name="title" className="form-control" placeholder="Add Task..."/>

</div>

</form>

<TaskList tasks={this.state.tasks**} deleteTask={this.deleteTask} updateTask={this.updateTask}** />

</div> );

}

}

export default App;

Modify src\TaskList.js as shown:

import React, { Component } from 'react'; // React must be in scope when using JSX

import Task from './Task';

import './TaskList.css';

class TaskList extends Component {

    render() {

        // NOTE task.isDone can be undefined, so forcing the prop to be either true or false prevents

        // a "switch from uncontrolled to controlled" warning

        const tasks = this.props.tasks.map((task) =>

            <Task key={task.\_id} id={task.\_id} title={task.title} selected={task.isDone ? true : false**} deleteTask={this.props.deleteTask} updateTask={this.props.updateTask}** />);

        return (

            <div className="TaskList">

            {tasks}

            </div>

        );

    }

}

export default TaskList;

Modify src\Task.js as shown:

import React, { Component } from 'react';

import './Task.css';

class Task extends Component {

    constructor() {

        super();

**this.handleClickDelete = this.handleClickDelete.bind(this);**

**this.handleSelect = this.handleSelect.bind(this);**

    }

**shouldComponentUpdate(nextProps, nextState) {**

**return this.props.selected !== nextProps.selected;**

**}**

**handleSelect() {**

**this.props.updateTask(this.props.id, this.props.title, this.props.selected);**

**}**

**handleClickDelete() {**

**event.preventDefault();**

**this.props.deleteTask(this.props.id);**

**}**

    render() {

        return (<span className="Task"><div className="col-md-1">

<input name="select" type="checkbox" checked={this.props.selected} **onChange={this.handleSelect}** />

</div>

<div className="col-md-7">

{this.props.title}

</div>

<div className="col-md-4">

<input type="button" **onClick={this.handleClickDelete}** value="Delete" className="btn btn-danger"/>

</div>

<br/><br/></span>);

    }

}

export default Task;

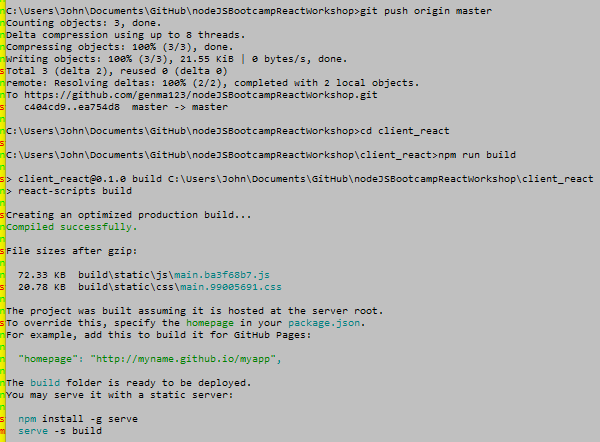
You should now be able to add and delete tasks, and mark them as done.

*Step 3 complete!*

In the Command Prompt in which the Webpack Dev Server is running, kill the server with Control-C. Then perform the following command to build the Production “bundle”:

npm run build

You should see messages such as the following:



In the first Command Prompt window you opened (running the REST API), kill the server with Control-C.

Then set the Node environment to production

set NODE\_ENV=production

which will affect the following logic in Node:

// Express only serves static assets in production

if (process.env.NODE\_ENV === 'production') {

app.use(express.static('client\_react/build'));

}

Start the NodeJS/Express server again using:

node server

Then open a new Chrome browser tab or window, and navigate to <http://localhost:5000/>. You should see the complete Todo list application running in Production mode.

*Step 4 and workshop complete!*