Creating a React JS Application Part 1

# Install Cmder: Console Emulators on Windows

# Create a New Project and Install Dependencies

1. Go to cmder, create a new directory called FirstApp. Use the following commands:

mkdir FirstApp

cd FirstApp

npm init

1. Install the following tooling dependencies:

npm install webpack babel-loader babel-core npm-run-all babel-preset-es2015 babel-preset-react serve --save-dev

Dependencies installed:

* Webpack: Module loader/bundler for JavaScript.
* Babel-loader: Webpack loader for Babel. Babel transpiles ES6/ES7 to ES5.
* Babel-preset-react: React preset for Babel. Just for Babel to understand React terms like JSX
* Serve: HTTP server

# Configure Webpack Loader and Babel

1. Open Visual Studio Code with the following command in cmder:

code .

1. Go to File->New File and name it webpack.config.js and add the following code:

var webpack = require('webpack');

var path = require('path');

var BUILD\_DIR = path.resolve(\_\_dirname, 'public/');

var APP\_DIR = path.resolve(\_\_dirname, 'src/');

var config = {

entry: APP\_DIR + '/index.jsx',

output: {

path: BUILD\_DIR,

filename: 'bundle.js'

},

module : {

loaders : [

{

test : /\.jsx?/,

include : APP\_DIR,

loader : 'babel-loader'

}

]

}

};

module.exports = config;

Explanation: The most important aspect of a **Webpack config** is the exported **config** object. The minimal code above just needs an entry point, entry which is where bundling needs to begin from. It also requires an output, output which is where the bundled result is dumped and then module, which defines what loaders should be used during the bundling process. In our case, **babel** is the loader we need.

1. Create a new file named “.babelrc”. Go to File->New File and name it “.babelrc”. This file will tell Babel which presets it should make use of. Type the following in this file

{

"presets" : ["es2015", "react"]

}

1. Create a new file named “package.json”. Go to File-New File and name it “package.json”. Add the following commands to run webpack:

{

"name": "firstapp",

"version": "1.0.0",

"description": "my first app",

"scripts": {

"watch": "webpack -d --watch",

"build": "webpack",

"serve": "serve ./public",

"start": "npm-run-all --parallel watch serve"

},

"author": "Luis Quintero",

"license": "ISC",

"devDependencies": {

"babel-loader": "^7.1.2",

"babel-preset-es2015": "^6.24.1",

"babel-preset-react": "^6.24.1",

"serve": "^6.4.1",

"webpack": "^3.9.0"

},

"dependencies": {

"babel-core": "^6.26.0",

"react": "^16.2.0",

"react-dom": "^16.2.0",

"react-router": "^4.2.0"

}

}

1. Create a new folder called “public” and an entry file named “index.html”.

* In cmdr run the following:

mkdir public

cd public

* In Visual Studio Code, go to File->New File and name it “index.html” and add the following code:

<html>

<head>

<!--Stylesheet-->

<link rel="stylesheet" href="styles.css">

</head>

<body>

<!--Container for React rendering-->

<div id="container"></div>

<!--Bundled file-->

<script src="bundle.js"></script>

</body>

</html>

Explanation: Loads the bundle and defines the DOM element to mount our React app.

1. Create the following folder structure

# Folder Structure

|---public

|------index.html # App entry

|------styles.css # Custom style

|------bundle.js # Generated

|---src # Components live here **<Need of Component Directory Here???>**

|------car

|---------car.component.jsx

|---------car-detail.component.jsx

|------common

|---------about.component.jsx

|---------home.component.jsx

|---------main.component.jsx

|------index.jsx # Build entry

|---.babelrc # Babel config file

|---index.js

|---package.json

|---webpack.config.js # Webpack config gile

# Setting up React for Routing

1. Go to cmder, create a new directory called FirstApp. Go to that folder and run the following command:[Here FirstApp refers the existing Folder we created earlier]

npm install react react-dom react-router --save

1. Create a basic component named “index.jsx” in the “src” folder. Go to File->New File and name it “index.jsx”. Add the following code:

// ./src/index.jsx

import React, { Component } from 'react';

import { render } from 'react-dom';

class Home extends Component {

render(){

return (<h1>Hi</h1>);

}

}

render(<Home />, document.getElementById('container'));

1. Test the results. Go to cmder and run the following:

npm start

Then type the following in your browser:

http://localhost:5000/index.html

# Creating Final Routes and Components

1. Add more routing feature to the index.jsx
2. file: save and exit

// ./src/index.jsx

import React, { Component } from 'react';

import { render } from 'react-dom';

// Import routing components

import {Router, Route} from 'react-router';

// Import custom components

import Home from './common/home.component.jsx'

import About from './common/about.component.jsx'

import Car from './car/car.component.jsx'

render(

<Router>

<Route path="/" component={Home}/>

<Route path="/cars" component={Car}/>

<Route path="/about" component={About}/>

</Router>,

document.getElementById('container')

);

Explanation: The path attribute defines the rout URL and component attribute defines the component for this source.

1. Create the “Car” component

* Create a new file under the “/src/car” folder named car.component.jsx and add the following code:

// ./src/car/car.component.jsx

import React, { Component } from 'react';

class Car extends Component {

render(){

return (<h1>Cars page</h1>);

}

}

export default Car

1. Create the “Home” component

* Create a new file under the “/src/common” folder named home.component.jsx and add the following code:

// ./src/common/home.component.jsx

import React, { Component } from 'react';

class Home extends Component {

render(){

return (<h1>Home Page</h1>);

}

}

export default Home

1. Create the “About” component

* Create a new file under the “/src/common” folder named about.component.jsx and add the following code:

// ./src/common/about.component.jsx

import React, { Component } from 'react';

class About extends Component {

render(){

return (<h1>About Page</h1>);

}

}

export default About

1. Once Webpack is running all saved update will be exposed to the local host.
2. For Packaging and deployment into SharePoint find the - “Packaging and Deployment” Document.

Good Luck & Happy Coding..!!