Chapter 8: React Boilerplate [React + Babel + Webpack]

Section 8.1: react-starter project

About this Project

This is simple boilerplate project. This post will guide you to set up the environment for ReactJs + Webpack + Bable.

Lets get Started

we will need node package manager for fire up express server and manage dependencies throughout the project. if you are new to node package manager, you can check here. Note : Installing node package manager is require here.

Create a folder with suitable name and navigate into it from terminal or by GUI.Then go to terminal and type `npm init` this will create a package.json file, Nothing scary , it will ask you few question like name of your project ,version, description, entry point, git repository, author, license etc. Here entry point is important because node will initially look for it when you run the project. At the end it will ask you to verify the information you provide. You can type *yes* or modify it. Well that's it , our *package.json* file is ready.

Express server setup run *npm install express@4 --save*. This is all the dependencies we needed for this project.Here save flag is important, without it *package.js* file will not be updated. Main task of *package.json* is to store list of dependencies. It will add express version 4. Your *package.json* will look like `"dependencies": { "express": "^4.13.4", ............. },`

After complete download you can see there is *node_modules* folder and sub folder of our dependencies. Now on the root of project create new file *server.js* file. Now we are setting express server. I am going to past all the code and explain it later.

```
var express = require('express');
// Create our app
var app = express();

app.use(express.static('public'));

app.listen(3000, function () {
  console.log('Express server is using port:3000');
});
```

var express = require('express'); this will gave you the access of entire express api.

var app = express(); will call express library as function. *app.use();* let the add the functionality to your express application. *app.use(express.static('public'));* will specify the folder name that will be expose in our web server. *app.listen(port, function(){})* will here our port will be *3000* and function we are calling will verify that out web server is running properly. That's it express server is set up.

Now go to our project and create a new folder public and create *index.html* file. *index.html* is the default file for you application and Express server will look for this file. The *index.html* is simple html file which looks like

```
<!DOCTYPE html>
<html>

<head>
```

```
  <meta charset="UTF-8"/>
</head>

<body>
  <h1>hello World</h1>
</body>

</html>
```

And go to the project path through the terminal and type *node server.js*. Then you will see * console.log('Express server is using port:3000');*.

Go to the browser and type *http://localhost:3000* in nav bar you will see *hello World*.

Now go inside the public folder and create a new file *app.jsx*. JSX is a preprocessor step that adds XML syntax to your JavaScript.You can definitely use React without JSX but JSX makes React a lot more elegant. Here is the sample code for *app.jsx*

```
ReactDOM.render(
  <h1>Hello World!!!</h1>,
  document.getElementById('app')
);
```

Now go to *index.html* and modify the code , it should looks like this

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8"/>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.8.23
                 /browser.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/react/0.14.7/react.js">
    </script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/react/0.14.7/react-dom.js">    </script>
</head>

<body>
  <div id="app"></div>

  <script type="text/babel" src="app.jsx"></script>
</body>

</html>
```

With this in place you are all done, I hope you find it simple.

Section 8.2: Setting up the project

You need Node Package Manager to install the project dependencies. Download node for your operating system from Nodejs.org. Node Package Manager comes with node.

You can also use Node Version Manager to better manage your node and npm versions. It is great for testing your project on different node versions. However, it is not recommended for production environment.

Once you have installed node on your system, go ahead and install some essential packages to blast off your first React project using Babel and Webpack.

Before we actually start hitting commands in the terminal. Take a look at what Babel and Webpack are used for.

You can start your project by running `npm init` in your terminal. Follow the initial setup. After that, run following commands in your terminal-

Dependencies:

`npm install react react-dom --save`

Dev Dependecies:

```
npm install babel-core babel-loader babel-preset-es2015 babel-preset-react babel-preset-stage-0
webpack webpack-dev-server react-hot-loader --save-dev
```

Optional Dev Dependencies:

`npm install eslint eslint-plugin-react babel-eslint --save-dev`

You may refer to this sample package.json

Create `.babelrc` in your project root with following contents:

```
{
  "presets": ["es2015", "stage-0", "react"]
}
```

Optionally create `.eslintrc` in your project root with following contents:

```
{
  "ecmaFeatures": {
    "jsx": true,
    "modules": true
  },
  "env": {
    "browser": true,
    "node": true
  },
  "parser": "babel-eslint",
  "rules": {
    "quotes": [2, "single"],
    "strict": [2, "never"],
  },
  "plugins": [
    "react"
  ]
}
```

Create a `.gitignore` file to prevent uploading generated files to your git repo.

```
node_modules
npm-debug.log
.DS_Store
dist
```

Create `webpack.config.js` file with following minimum contents.

```
var path = require('path');
var webpack = require('webpack');
```

```
module.exports = {
  devtool: 'eval',
  entry: [
    'webpack-dev-server/client?http://localhost:3000',
    'webpack/hot/only-dev-server',
    './src/index'
  ],
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'bundle.js',
    publicPath: '/static/'
  },
  plugins: [
    new webpack.HotModuleReplacementPlugin()
  ],
  module: {
    loaders: [{
      test: /\.js$/,
      loaders: ['react-hot', 'babel'],
      include: path.join(__dirname, 'src')
    }]
  }
};
```

And finally, create a `sever.js` file to be able to run `npm start`, with following contents:

```
var webpack = require('webpack');
var WebpackDevServer = require('webpack-dev-server');
var config = require('./webpack.config');

new WebpackDevServer(webpack(config), {
  publicPath: config.output.publicPath,
  hot: true,
  historyApiFallback: true
}).listen(3000, 'localhost', function (err, result) {
  if (err) {
    return console.log(err);
  }

  console.log('Serving your awesome project at http://localhost:3000/');
});
```

Create `src/app.js` file to see your React project do something.

```
import React, { Component } from 'react';

export default class App extends Component {
  render() {
    return (
      <h1>Hello, world.</h1>
    );
  }
}
```

Run `node server.js` or `npm start` in the terminal, if you have defined what `start` stands for in your `package.json`