02. REACT Environment Setup.

In this chapter, we will show you how to set up an environment for successful React development. Notice that there are many steps involved but this will help speed up the development process later. We will need NodeJS, so if you don't have it installed, check the link from the following link.

- **NodeJS and NPM.**
  NodeJS is the platform needed for the ReactJS development.
  https://nodejs.org/

After successfully installing NodeJS, we can start installing React upon it using npm.
You can install NodeJS in two ways

- Using webpack and babel.

- Using the create-react-app command.

2.1. Installing ReactJS using webpack and babel

Webpack is a module bundler (manages and loads independent modules). It takes dependent modules and compiles them to a single (file) bundle. You can use this bundle while developing apps using command line or, by configuring it using **webpack.config file**.

Babel is a JavaScript compiler and transpiler. It is used to convert one source code to other. Using this you will be able to use the new ES6 features in your code where, babel converts it into plain old ES5 which can be run on all browsers.

Step 1 - Create the Root Folder

Create a folder with name reactApp on the desktop to install all the required files, using the mkdir command.

Example:

```
C:\Users\username\Desktop>mkdir reactApp
C:\Users\username\Desktop>cd reactApp
```

To create any module, it is required to generate the ***package.json*** file. Therefore, after Creating the folder, we need to create a ***package.json*** file. To do so you need to run the **npm init** command from the command prompt.

Example:

```
C:\Users\username\Desktop\reactApp>npm init
```

This command asks information about the module such as **packagename**, **description, author** etc. you can skip these using the **–y option**.

Example:

```
C:\Users\username\Desktop\reactApp>npm init -y

Wrote to C:\reactApp\package.json:
{
   "name": "reactApp",
   "version": "1.0.0",
   "description": "",
   "main": "index.js",
   "scripts": {
      "test": "echo \"Error: no test specified\" && exit 1"
   },
```

```
        "keywords": [],
        "author": "",
        "license": "ISC"
    }
```

Step 2 - install React and react dom

Since our main task is to install ReactJS, install it, and its dom packages, using install **react and react-dom** commands of npm respectively. You can add the packages we install, to *package.json* file using the **--save option**.

Example:

```
C:\Users\MyUser\Desktop\reactApp>npm install react --save
C:\Users\MyUser\Desktop\reactApp>npm install react-dom --save
```

Or, you can install all of them in single command as −

Example:

```
C:\Users\MyUser\Desktop\reactApp>npm install react react-dom --save
```

Step 3 - Install webpack

Since we are using webpack to generate bundler install ***webpack, webpack-dev-server*** and ***webpack-cl***i.

Example:

```
C:\Users\MyUser\Desktop\reactApp>npm install webpack –save
```

```
C:\Users\MyUser\Desktop\reactApp>npm install webpack-dev-server --save
C:\Users\MyUser\Desktop\reactApp>npm install webpack-cli --save
```

Or, you can install all of them in single command as −

Example:

```
C:\Users\MyUser\Desktop\reactApp>npm install webpack webpack-dev-server
webpack-cli --save
```

Step 4 - Install babel

Install babel, and its plugins ***babel-core, babel-loader, babel-preset-env, babel-preset-reac***t and, ***html-webpack-plugin***

Example:

```
C:\Users\MyUser\Desktop\reactApp>npm install babel-core --save-dev
C:\Users\MyUser\Desktop\reactApp>npm install babel-loader --save-dev
C:\Users\MyUser\Desktop\reactApp>npm install babel-preset-env --save-dev
C:\Users\MyUser\Desktop\reactApp>npm install babel-preset-react --save-dev
C:\Users\MyUser\Desktop\reactApp>npm install html-webpack-plugin --save-dev
```

Or, you can install all of them in single command as −

Example:

```
C:\Users\username\Desktop\reactApp>npm install babel-core babel-loader
babel-preset-env babel-preset-react html-webpack-plugin --save-dev
```

Step 5 - Create the Files

To complete the installation, we need to create certain files namely, ***index.html, App.js, main.js, webpack.config.js*** and***, .babelrc***. You can create these files manually or, using command prompt.

Example:

```
C:\Users\MyUser\Desktop\reactApp>type nul > index.html
C:\Users\MyUser\Desktop\reactApp>type nul > App.js
C:\Users\MyUser\Desktop\reactApp>type nul > main.js
C:\Users\MyUser\Desktop\reactApp>type nul > webpack.config.js
C:\Users\MyUser\Desktop\reactApp>type nul > .babelrc
```

Step 6 - Set Compiler, Server and Loaders

Open ***webpack-config.js*** file and add the following code. We are setting webpack entry point to be main.js. Output path is the place where bundled app will be served. We are also setting the development server to 8001 port. You can choose any port you want.

Example: *webpack.config.js*

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {
   entry: './main.js',
   output: {
      path: path.join(__dirname, '/bundle'),
      filename: 'index_bundle.js'
   },
   devServer: {
      inline: true,
      port: 8080
```

```
        },
        module: {
            rules: [
                {
                    test: /\.jsx?$/,
                    exclude: /node_modules/,
                    loader: 'babel-loader',
                    query: {
                        presets: ['es2015', 'react']
                    }
                }
            ]
        },
        plugins:[
            new HtmlWebpackPlugin({
                template: './index.html'
            })
        ]
    }
```

Open the ***package.json*** and delete "*test*" "*echo \"Error: no test specified\"* && exit 1" inside "*scripts*" object. We are deleting this line since we will not do any testing in this tutorial. Let's add the **start** and **build** commands instead.

Example:

```
    "start": "webpack-dev-server --mode development --open --hot",
    "build": "webpack --mode production"
```

Step 7 - index.html

This is just regular HTML. We are setting *div id = "app"* as a root element for our app and adding ***index_bundle.js*** script, which is our bundled app file.

Example:

```
<!DOCTYPE html>
<html lang = "en">
   <head>
      <meta charset = "UTF-8">
      <title>React App</title>
   </head>
   <body>
      <div id = "app"></div>
      <script src = 'index_bundle.js'></script>
   </body>
</html>
```

Step 8 – App.jsx and main.js

This is the first React component. We will explain React components in depth in a subsequent chapter. This component will render Hello World.

Example: *App.js*

```
import React, { Component } from 'react';
class App extends Component{
   render(){
      return(
         <div>
            <h1>Hello World</h1>
         </div>
```

```
            );
        }
    }
    export default App;
```

We need to import this component and render it to our root App element, so we can see it in the browser.

Example: *main.js*

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App.js';

ReactDOM.render(<App />, document.getElementById('app'));
```

Note − Whenever you want to use something, you need to import it first. If you want to make the component usable in other parts of the app, you need to export it after creation and import it in the file where you want to use it.

Create a file with name ***.babelrc*** and copy the following content to it.

Example:

```
{
   "presets":["env", "react"]
}
```

Step 9 - Running the Server

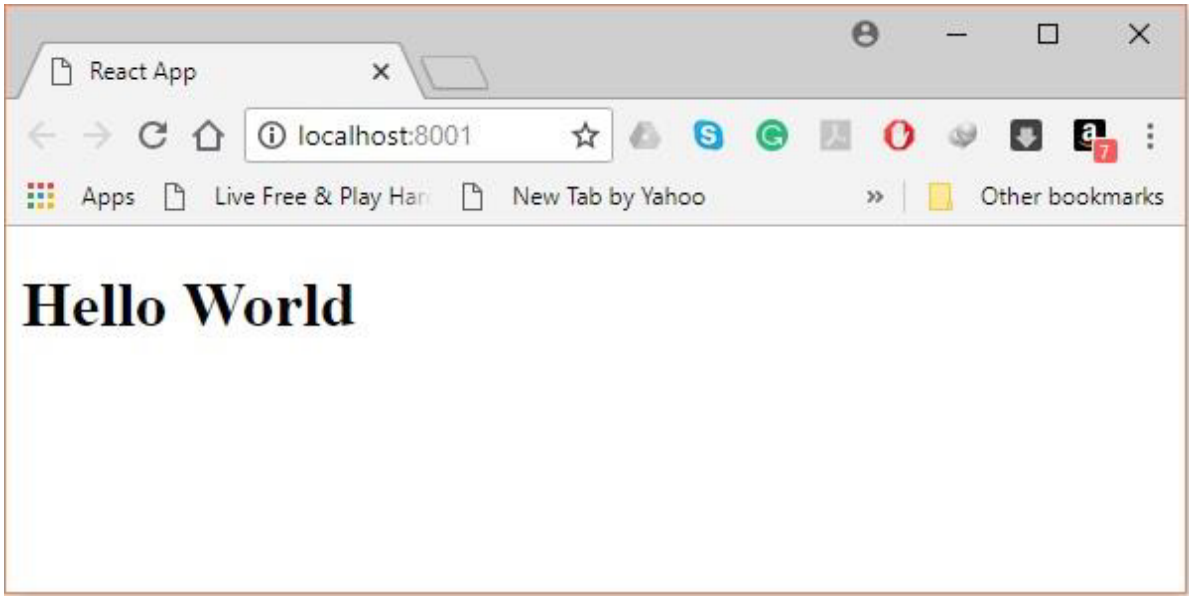
The setup is complete and we can start the server by running the following command.

Example:

```
C:\Users\username\Desktop\reactApp>npm start
```

It will show the port we need to open in the browser. In our case, it is **http://localhost:8001/**. After we open it, we will see the following output.

Output:



Step 10 - Generating the bundle

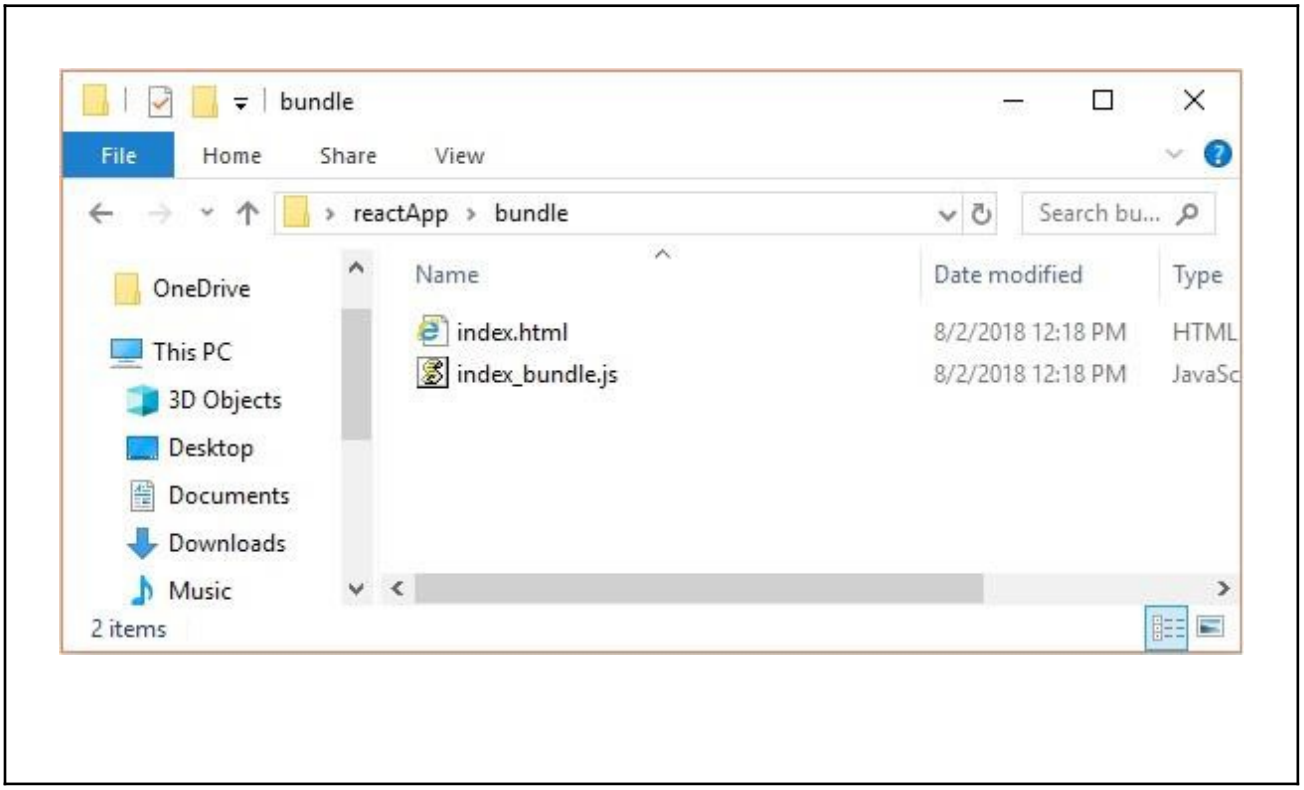
Finally, to generate the bundle you need to run the build command in the command prompt as −

Example:

```
C:\Users\MyUser\Desktop\reactApp>npm run build
```

This will generate the bundle in the current folder as shown below.

Output:



2.2. Using the create-react-app command

Instead of using **webpack** and babel you can install ReactJS more simply by installing ***create-react-app***.

Step 1 - install create-react-app

Browse through the desktop and install the ***Create React App*** using command prompt as shown below:

Example:

```
C:\Users\MyUser>cd C:\Users\MyUser\Desktop\
C:\Users\MyUser\Desktop>npx create-react-app my-app
```

This will create a folder named my-app on the desktop and installs all the required files in it.

Step 2 - Delete all the source files

Browse through the *src* folder in the generated ***my-app*** folder and remove all the files in it as shown below −

Example:

```
C:\Users\MyUser\Desktop>cd my-app/src
C:\Users\MyUser\Desktop\my-app\src>del *
C:\Users\MyUser\Desktop\my-app\src\*, Are you sure (Y/N)? y
```

Step 3 - Add files

Add files with names ***index.css*** and ***index.js*** in the src folder as −

Example:

```
C:\Users\MyUser\Desktop\my-app\src>type nul > index.css
C:\Users\MyUser\Desktop\my-app\src>type nul > index.js
```

In the ***index.js*** file add the following code

Example:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
```

Step 4 - Run the project

Finally, run the project using the start command.

Ouput: