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 table.

NodeJS and NPM

NodeJS is the platform needed for the ReactJS development. Check out our NodeJS Environment Setup.

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

- Using webpack and babel.
- Using the create-react-app command.

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.

```
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.

```
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.

Wrote to C:\reactApp\package.json:

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

```
{
  "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.

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

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

```
C:\Users\username\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-cli.

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

Step 4 - Install babel

Install babel, and its plugins babel-core, babel-loader, babel-preset-env, babel-preset-react and, html-webpack-plugin

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

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**.

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

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: 8001
  },
  module: {
    rules: [
      {
        test: /\.jsx?$/,
        exclude: /node_modules/,
        loader: 'babel-loader',
        query: {
          presets: ['es2015', 'react']
        }
      }
    ]
  },
  plugins:[
    new HtmlWebpackPlugin({
      template: './index.html'
    })
  ]
}

```

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.

```
<!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**.

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.

main.js

```
1    import React from 'react';
2    import ReactDOM from 'react-dom';
3    import App from './App.js';
4    ReactDOM.render(<App />, document.getElementById('app'));
5
```

Step 9 - Running the Server

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

```
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.