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.

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.

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

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

C:\Users\username\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-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 --save-dev

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

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**.

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'

})

]

}

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.

"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.

<!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**

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.

{

"presets":["env", "react"]

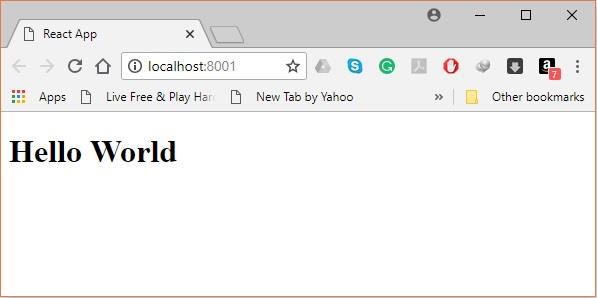
}

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.



Step 10 - Generating the bundle

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

C:\Users\Tutorialspoint\Desktop\reactApp>npm run build

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

