**[next →](https://www.javatpoint.com/react-create-react-app)**[**← prev**](https://www.javatpoint.com/react-version)

React Environment Setup

In this section, we will learn how to set up an environment for the successful development of ReactJS application.

Pre-requisite for ReactJS

1. NodeJS and NPM
2. React and React DOM
3. Webpack
4. Babel

Ways to install ReactJS

There are two ways to set up an environment for successful ReactJS application. They are given below.

1. Using the npm command
2. Using the create-react-app command

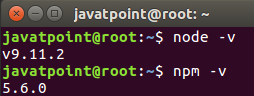
1. Using the npm command

**Install NodeJS and NPM**

NodeJS and NPM are the platforms need to develop any ReactJS application. You can install NodeJS and NPM package manager by the link given below.

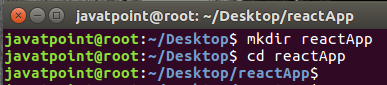
<https://www.javatpoint.com/install-nodejs-on-linux-ubuntu-centos>

To verify NodeJS and NPM, use the command shown in the below image.



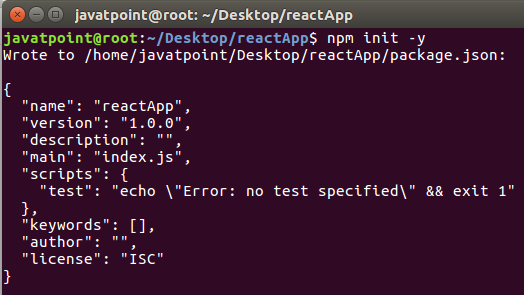
**Install React and React DOM**

Create a **root** folder with the name **reactApp** on the desktop or where you want. Here, we create it on the desktop. You can create the folder directly or using the command given below.



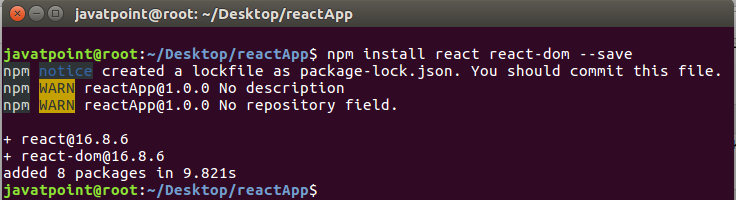
Now, you need to create a **package.json** file. To create any module, it is required to generate a package.json file in the project folder. To do this, you need to run the following command as shown in the below image.

1. javatpoint@root:~/Desktop/reactApp> npm init -y



After creating a package.json file, you need to install **react** and its DOM **packages** using the following npm command in the terminal window as shown in the below image.

1. javatpoint@root:~/Desktop/reactApp>npm install react react-dom --save



You can also use the above command separately which can be shown as below.

1. javatpoint@root:~/Desktop/reactApp>npm install react --save
2. javatpoint@root:~/Desktop/reactApp>npm install react-dom --save

**Install Webpack**

Webpack is used for module packaging, development, and production pipeline automation. We will use **webpack-dev-server** during development, **webpack** to create production builds, and **webpack CLI** provides a set of commands. Webpack compiles these into a single file(bundle). To install webpack use the command shown in the below image.

1. javatpoint@root:~/Desktop/reactApp>npm install webpack webpack-dev-server webpack-cli --save



You can also use the above command separately which can be shown as below.

1. javatpoint@root:~/Desktop/reactApp>npm install webpack --save
2. javatpoint@root:~/Desktop/reactApp>npm install webpack-dev-server --save
3. javatpoint@root:~/Desktop/reactApp>npm install webpack-cli --save

**Install Babel**

Babel is a JavaScript compiler and transpiler used to convert one source code to others. It compiles React JSX and ES6 to ES5 JavaScript which can be run on all browsers. We need **babel-loader** for JSX file types, **babel-preset-react** makes your browser update automatically when any changes occur to your code without losing the current state of the app. ES6 support requires **babel-preset-env** Babel preset. To install webpack use the following command shown in the below image.

1. javatpoint@root:~/Desktop/reactApp>npm install babel-core babel-loader babel-preset-env babel-preset-react babel-webpack-plugin --save-dev



You can also use the above command separately which can be shown as below.

1. javatpoint@root:~/Desktop/reactApp>npm install babel-core --save-dev
2. javatpoint@root:~/Desktop/reactApp>npm install babel-loader --save-dev
3. javatpoint@root:~/Desktop/reactApp>npm install babel-preset-env --save-dev
4. javatpoint@root:~/Desktop/reactApp>npm install babel-preset-react --save-dev
5. javatpoint@root:~/Desktop/reactApp>npm install babel-webpack-plugin --save-dev

**Create Files**

To complete the installation process, you need to add the following files in your project folder. These files are **index.html, App.js, main.js, webpack.config.js** and, **.babelrc.** You can create these files by manually, or by using the command prompt.

1. javatpoint@root:~/Desktop/reactApp>touch index.html
2. javatpoint@root:~/Desktop/reactApp>touch App.js
3. javatpoint@root:~/Desktop/reactApp>touch main.js
4. javatpoint@root:~/Desktop/reactApp>touch webpack.config.js
5. javatpoint@root:~/Desktop/reactApp>touch .babelrc

Set Compiler, Loader, and Server for React Application

**Configure webpack**

You can configure webpack in the **webpack.config.js** file by adding the following code. It defines your app entry point, build output and the extension which will resolve automatically. It also set the development server to **8080** port. It defines the loaders for processing various file types used within your app and wrap up by adding plugins needed during our development.

***webpack.config.json***

1. **const** path = require('path');
2. **const** HtmlWebpackPlugin = require('html-webpack-plugin');
4. module.exports = {
5. entry: './main.js',
6. output: {
7. path: path.join(\_\_dirname, '/bundle'),
8. filename: 'index\_bundle.js'
9. },
10. devServer: {
11. inline: **true**,
12. port: 8080
13. },
14. module: {
15. rules: [
16. {
17. test: /\.jsx?$/,
18. exclude: /node\_modules/,
19. use: {
20. loader: "babel-loader",
21. }
22. }
23. ]
24. },
25. plugins:[
26. **new** HtmlWebpackPlugin({
27. template: './index.html'
28. })
29. ]
30. }

Now, open the **package.json** file and delete ***"test" "echo \" Error: no test specified\" && exit 1"*** inside ***"scripts"*** object, then add the start and build commands instead. It is because we will not perform any testing in this app.

1. {
2. "name": "reactApp",
3. "version": "1.0.0",
4. "description": "",
5. "main": "index.js",
6. "scripts": {
7. "start": "webpack-dev-server --mode development --open --hot",
8. "build": "webpack --mode production"
9. },
10. "keywords": [],
11. "author": "",
12. "license": "ISC",
13. "dependencies": {
14. "react": "^16.8.6",
15. "react-dom": "^16.8.6",
16. "webpack-cli": "^3.3.1",
17. "webpack-dev-server": "^3.3.1"
18. },
19. "devDependencies": {
20. "@babel/core": "^7.4.3",
21. "@babel/preset-env": "^7.4.3",
22. "@babel/preset-react": "^7.0.0",
23. "babel-core": "^6.26.3",
24. "babel-loader": "^8.0.5",
25. "babel-preset-env": "^1.7.0",
26. "babel-preset-react": "^6.24.1",
27. "html-webpack-plugin": "^3.2.0",
28. "webpack": "^4.30.0"
29. }
30. }

**HTML webpack template for index.html**

We can add a custom template to generate **index.html** using the **HtmlWeb-packPlugin** plugin. This enables us to add a viewport tag to support mobile responsive scaling of our app. It also set the **div id = "app"** as a root element for your app and adding the **index\_bundle.js** script, which is our bundled app file.

1. <!DOCTYPE html>
2. <html lang = "en">
3. <head>
4. <meta charset = "UTF-8">
5. <title>React App</title>
6. </head>
7. <body>
8. <div id = "app"></div>
9. <script src = 'index\_bundle.js'></script>
10. </body>
11. </html>

**App.jsx and main.js**

This is the first React component, i.e. app entry point. It will render Hello World.

***App.js***

1. **import** React, { Component } from 'react';
2. **class** App **extends** Component{
3. render(){
4. **return**(
5. <div>
6. <h1>Hello World</h1>
7. </div>
8. );
9. }
10. }
11. export **default** App;

Now, import this component and render it to your root App element so that you 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';
5. ReactDOM.render(<App />, document.getElementById('app'));

Note: If you want to use something, you need to import it first. 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 .babelrc file**

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

***.babelrc***

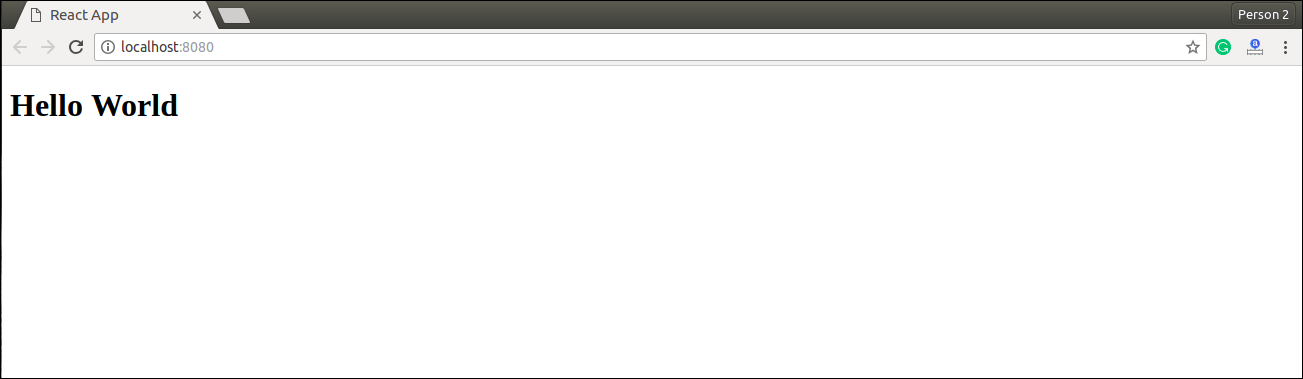
1. {
2. "presets":[
3. "@babel/preset-env", "@babel/preset-react"]
4. }

**Running the Server**

After completing the installation process and setting up the app, you can start the server by running the following command.

1. javatpoint@root:~/Desktop/reactApp>npm start

It will show the port number which we need to open in the browser. After we open it, you will see the following output.



**Generate the Bundle**

Now, generate the bundle for your app. Bundling is the process of following imported files and merging them into a single file: a **"bundle."** This bundle can then be included on a webpage to load an entire app at once. To generate this, you need to run the build command in command prompt which can be shown below.

1. javatpoint@root:~/Desktop/reactApp> npm run build

This command will generate the bundle in the current folder(in which your app belongs) and will be shown as like below image.

