|  |  |
| --- | --- |
| **ch4.ReactJS create-react-app** | **Date: 22-02-2022** |

**Topics**

React create-react-app tool, React Requirements, React App Directory Structure With create-react-app tool, Lets Make Changes in React App

# React create-react-app tool

1. Starting a new React project is very complicated, with so many build tools. It uses many dependencies, configuration files, and other requirements such as Babel, Webpack, ESLint before writing a single line of React code.
2. Create React App CLI tool removes all that complexities and makes React app simple. For this, you need to install the package using NPM, and then run a few simple commands to get a new React project.
3. The **create-react-app** is an excellent tool for beginners, which allows you to create and run React project very quickly. It does not take any configuration manually.
4. This tool is wrapping all of the required dependencies like **Webpack**, **Babel** for React project itself and then you need to focus on writing React code only. This tool sets up the development environment, provides an excellent developer experience, and optimizes the app for production.
5. The Create React App is maintained by **Facebook** and can works on any **platform**, for example, macOS, Windows, Linux, etc.

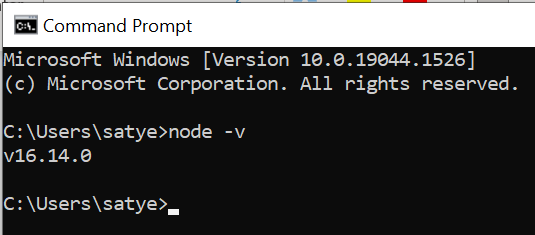
# React Requirements

* Step-1: To create a React Project using create-react-app, you need to have installed the following things in your system.

1. Node version >= 8.10
2. NPM version >= 5.6

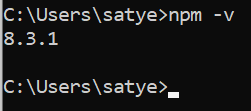
* Step-2: Let us check the current version of **Node**, Run the following command to check the Node version in the command prompt.

1. $ node -v



* Step-3: Let us check the current version of **NPM**, Run the following command to check the NPM version in the command prompt.

1. $ npm -v



Step-4: Now Install create-react-app package globally or locally.

C:\Users\javatpoint> npm install -g create-react-app

Step-5: Create a new React project

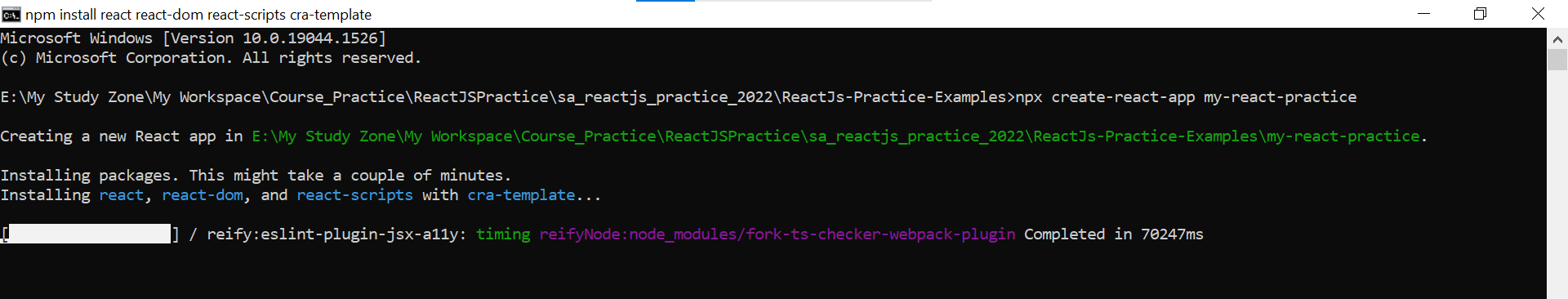
* Firstly create a new project Folder here I created as “**ReactJs-Practice-Examples**”.
* Now move inside in this folder and open your terminal or cmd and use this command to create react project **create-react-app my-react-practice** Here, I choose " **my-react-practice**" name for my project.

C:\Users\javatpoint> create-react-app reactproject

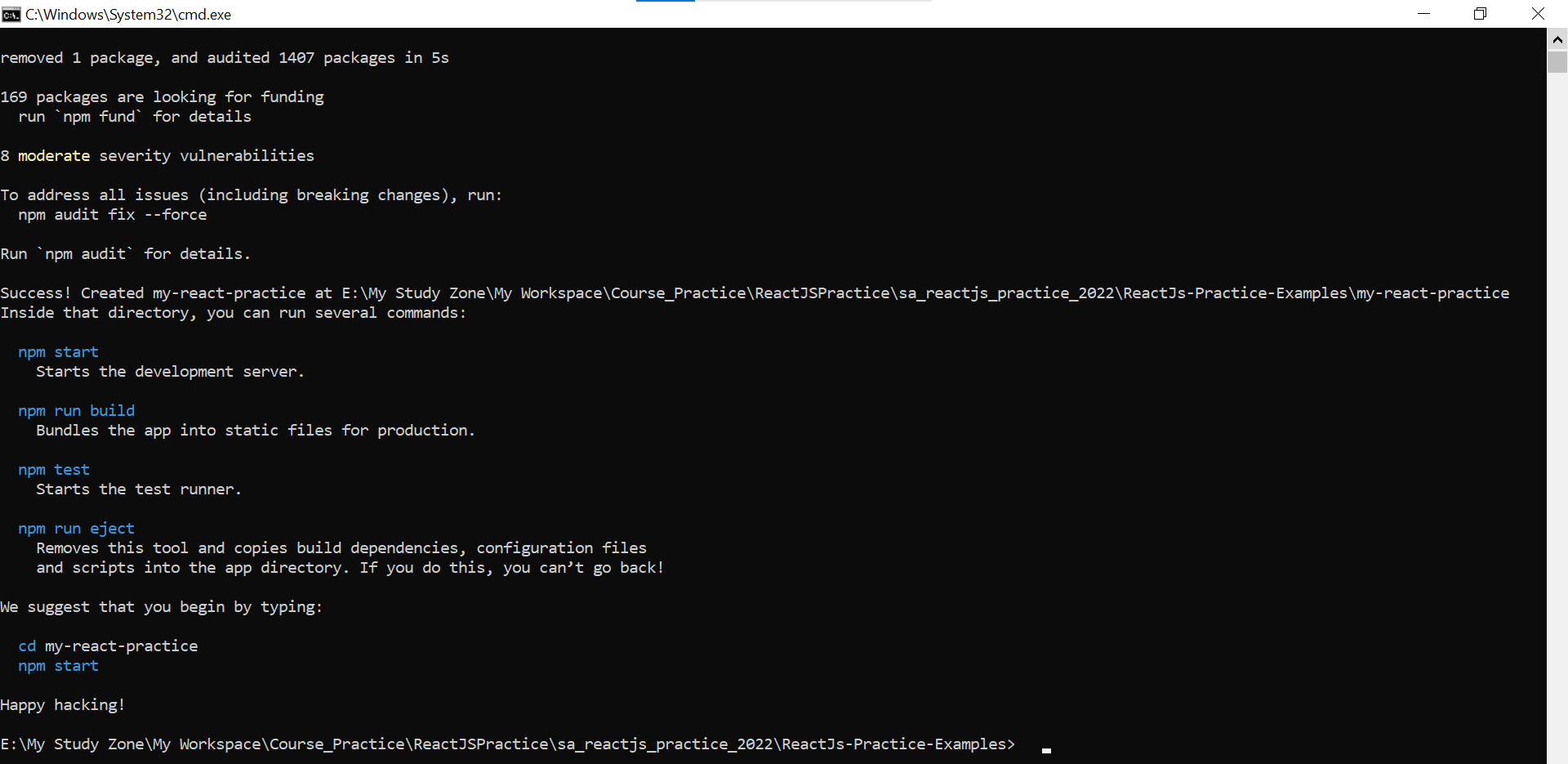
Step-6: We can skip step 4 and step 5

#### NOTE: We can combine the above two steps in a single command using npx. The npx is a package runner tool which comes with npm 5.2 and above version.

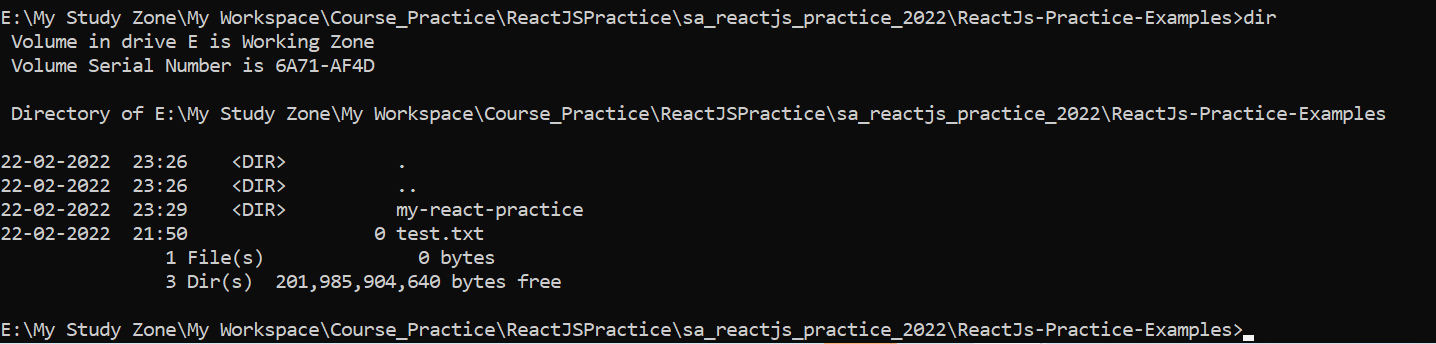
C:\Users\javatpoint> npx create-react-app **my-react-practice**



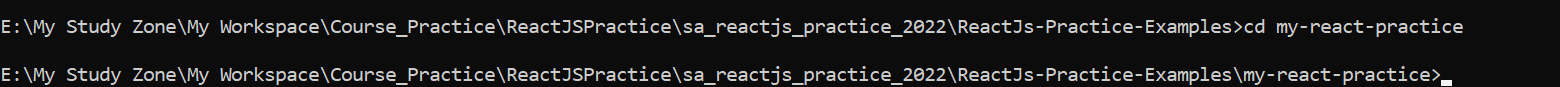
The above command will take some time to install the React and create a new project with the name " **my-react-practice**." Now, we can see the terminal as like below.



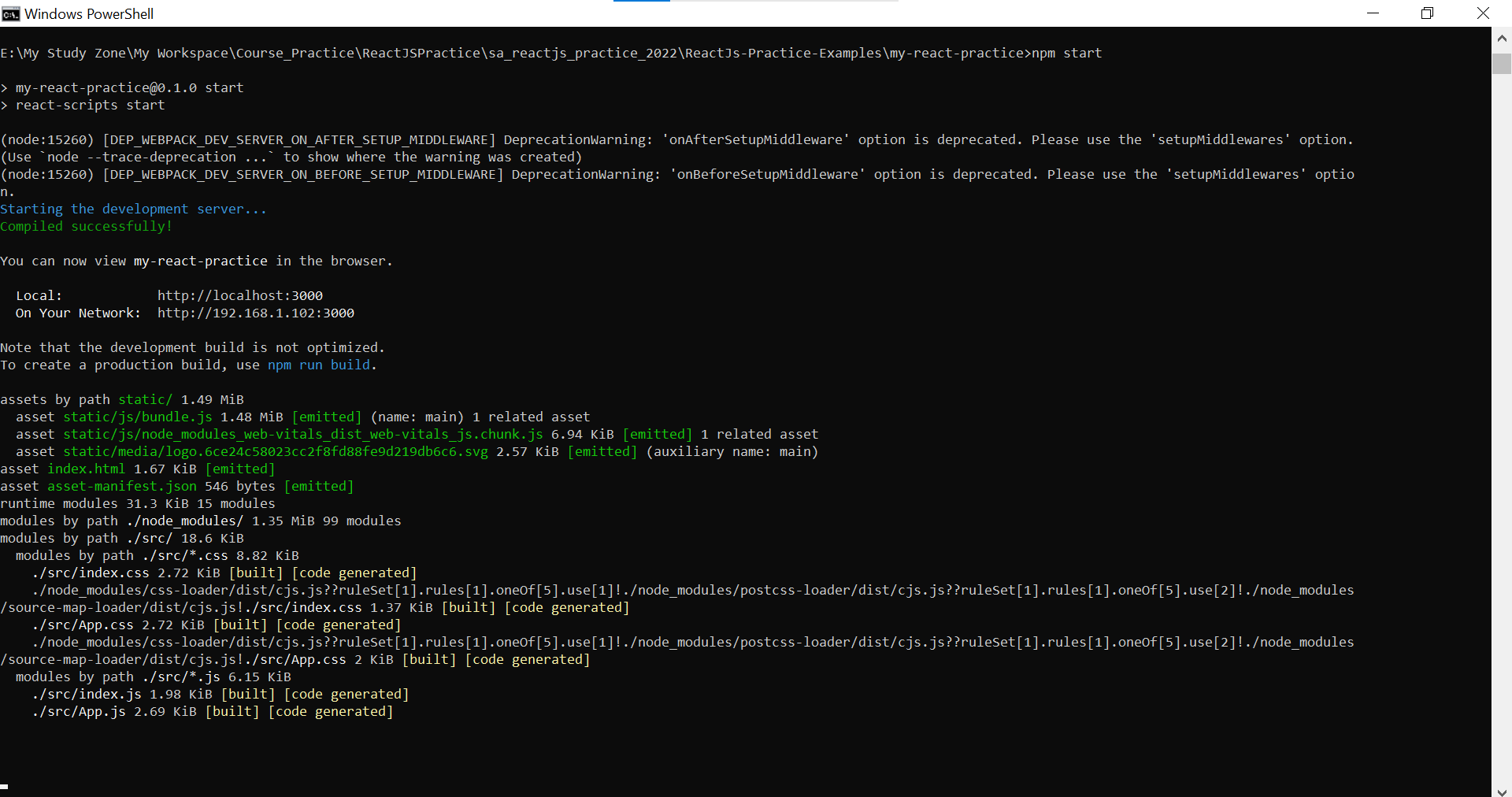
The above screen tells that the React project is created successfully on our system. Now, we need to start the server so that we can access the application on the browser. Type the following command in the terminal window.



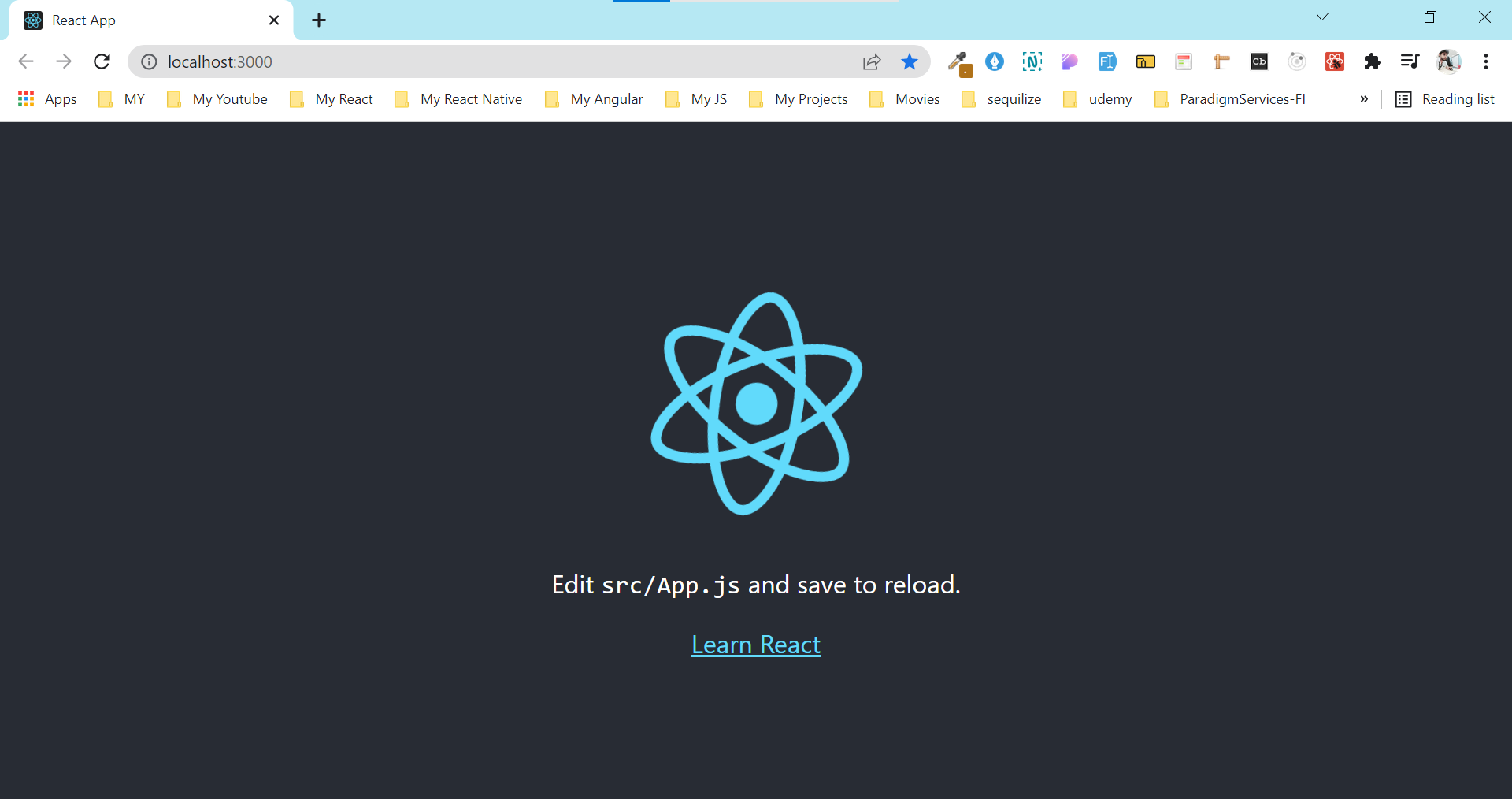
Step-7: $ cd Desktop



Step-8: $ npm start



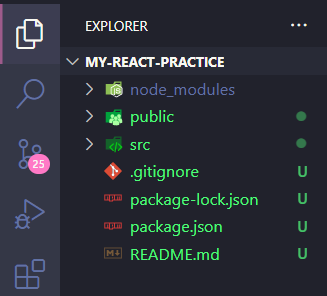
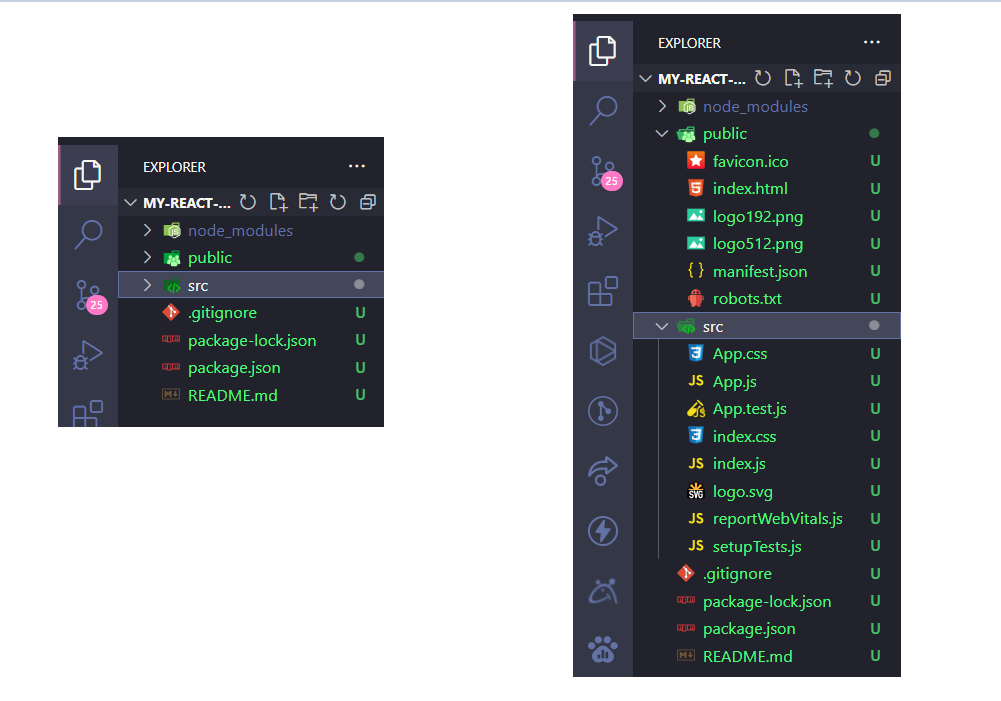
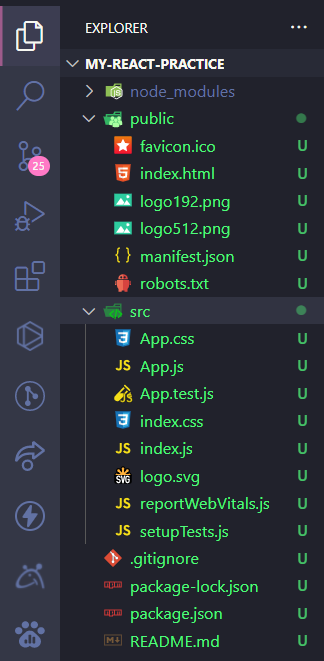
NPM is a package manager which starts the server and access the application at default server [http://localhost:3000](http://localhost:3000/). Now, we will get the following screen.



# React App Directory Structure With create-react-app tool

Next, open the project on Code editor. Here, I am using Visual Studio Code.

Our project's default structure looks like as below image.

In React application, there are several files and folders in the root directory. Some of them are as follows:

|  |  |  |
| --- | --- | --- |
| **SN** | **Folder And Files** | **Description** |
| **1** | **node\_modules** | It contains the React library and any other third party libraries needed. |
| **2** | **public** | It holds the public assets of the application. It contains the index.html where React will mount the application by default on the <div id="root"></div> element. |
| **3** | **src** | It contains the App.css, App.js, App.test.js, index.css, index.js, and serviceWorker.js files. Here, the App.js file always responsible for displaying the output screen in React. |
| **4** | **package-lock.json** | It is generated automatically for any operations where npm package modifies either the node\_modules tree or package.json. It cannot be published. It will be ignored if it finds any other place rather than the top-level package. |
| **5** | **package.json** | It holds various metadata required for the project. It gives information to npm, which allows to identify the project as well as handle the project?s dependencies. |
| **6** | **README.md** | It provides the documentation to read about React topics. |
| **7** |  |  |

# React – Lets Make Changes in React App

1. Now, open the **src >> App.js** file and make changes which you want to display on the screen.
2. After making desired changes, **save** the file.

# my-react-practice\src\App.js

import "./App.css";

function App() {

  return (

    <div>

      <h1>Hello React</h1>

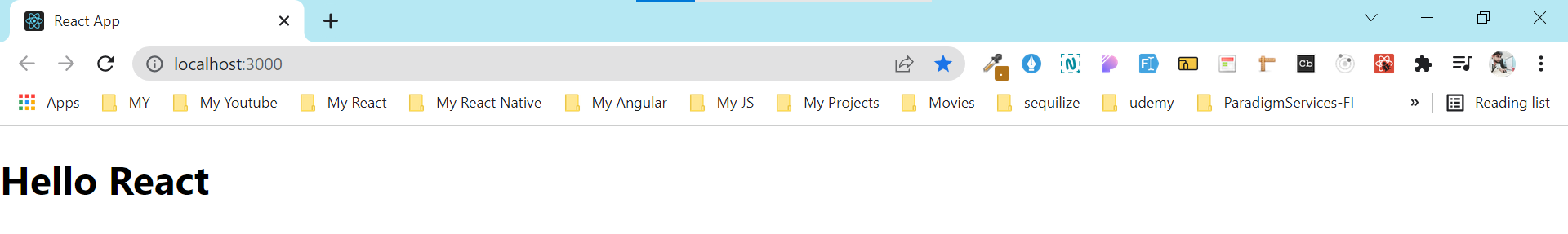
    </div>

  );

}

export default App;

**Output**



1. As soon as we save the file, Webpack recompiles the code, and the page will refresh automatically, and changes are reflected on the browser screen.
2. Now, we can create as many components as we want, import the newly created component inside the **App.js** file and that file will be included in our main **index.html** file after compiling by Webpack.
3. Next, if we want to make the project for the production mode, type the following command. This command will generate the production build, which is best optimized.

$ npm build