React JS

Introduction:

ReactJS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

Our ReactJS tutorial includes all the topics which help to learn ReactJS. These are ReactJS Introduction, ReactJS Features, ReactJS Installation, Pros and Cons of ReactJS, ReactJS JSX, ReactJS Components, ReactJS State, ReactJS Props, ReactJS Forms, ReactJS Events, ReactJS Animation and many more.

Why we use ReactJS?

The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM. We can use ReactJS on the client and server-side as well as with other frameworks. It uses component and data patterns that improve readability and helps to maintain larger apps.

Steps to insatll :

Create Folder : mkdir foldername

Path to created folder

Check nodeJS and Npm versions

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.

Install react-dom:

npm install react react-dom --save

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

npm install react --save

npm install react-dom --save

Install Webpack

npm install webpack webpack-dev-server webpack-cli --save

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

npm install webpack --save

npm install webpack-dev-server --save

npm install webpack-cli --save

Install Babel

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.

npm install babel-core --save-dev

npm install babel-loader --save-dev

npm install babel-preset-env --save-dev

npm install babel-preset-react --save-dev

npm install babel-webpack-plugin --save-dev

Create Files

touch index.html

touch App.js

touch main.js

touch webpack.config.js

touch .babelrc

Configure webpack:

```
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/,
       use: {
          loader: "babel-loader",
        }
      }
    ]
  },
  plugins:[
    new HtmlWebpackPlugin({
      template: './index.html'
    })
  ]
}
```

package .json file :
Update scripts in place of test:

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

Install webpack plugin:
npm i --save-dev html-webpack-plugin

Any server issues :
`npm audit fix --force`

Step-by-Step Process to Set Up ReactJS Development Environment

React JS environment setup can be done by following the below-mentioned steps.

1. Install Node.js and npm

It is imperative to have NodeJS installed on your PC for setting up a React development environment.

What are Node.js and npm? They are the runtime and command-line tools required to build React applications. While Node.js is a JS runtime environment that allows you to run js code outside of the web browser, npm is a package manager used to download javascript packages built to run on Node. You can get both of them together when you install Node.js.

You can visit the official link of NodeJS and download and install the latest version.

2. Test

Next, you need to test that Node.js and npm have been installed successfully. For this, you can run the commands: `node -v` and `npm -v`.

3. Install create-react-app using npm

The next important step in the ReactJS installation is to download the *create-node-app* package using npm (node package manager).

In the command prompt, write this command and hit Enter:

```
npm install -g create-react-app
```

This will start the installation process.

To check whether it is installed properly, write the following command and hit Enter:

```
create-react-app --version
```

It will show you the version of the create-react-app.

3. Create a project using create-react-app

Create a new folder on your desktop and rename it to React. Open command prompt through this folder and run the following command:

```
create-react-app myapp
```

It will initiate the installation of the app (*myapp*) in the folder you created on the desktop (React).

You can open this folder and see all the subfolders and files created for creating the ReactJS app using the *create-react-app* command.

4. Test the React app

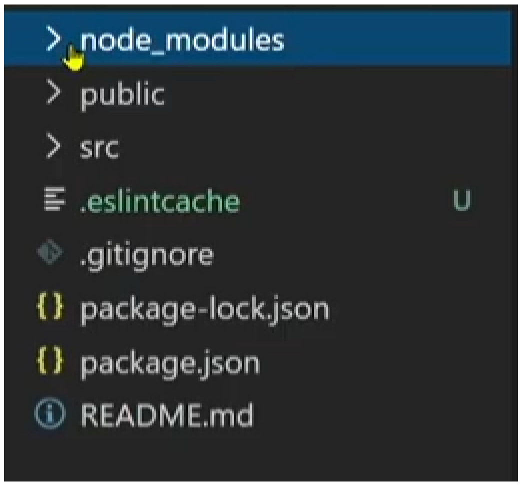
Run the given command in the Command Prompt:

```
npm start
```

It will open your app in the browser, which shows that you created the app successfully.

ReactJS Directory Structure

Once you have created the ReactJS app, open its folders in the Sublime Text Editor. Here is how the directory structure or fold structure of the React app looks in the editor.



```
> node_modules
> public
> src
.eslintcache            U
.gitignore
package-lock.json
package.json
README.md
```

- ### package.json file

This file holds the dependencies. Whatever you install within the app project, it is stored in this file. You can explore all the dependencies in the package.json file.

- ### node_modules

This folder holds the packages and libraries to make the development process easier. For instance, you can make use of some predefined functions from here instead of writing code from scratch. It is an auto-generated folder, and you are not supposed to make changes to it.

- **public**

This folder holds a number of different files, including the index.html file. The index.html file is the only file used in the ReactJS project. This works as a unified page for all sorts of other HTML files.

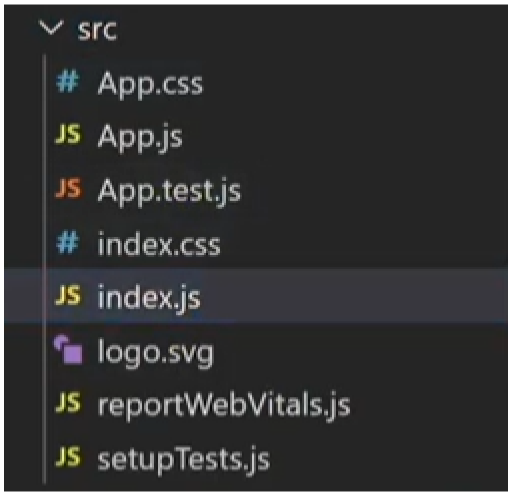
manifest.json

This file is used for the development of PWAs or Progressive Web Applications. If you want to create a PWA app, then this file plays a significant role.

logo192.png

When you begin a new project in ReactJS using the create-react-app, this logo file gets loaded in your project.

- **src folder in React**



```
src
  App.css
  App.js
  App.test.js
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  setupTests.js
```

index.js file

The index.js files are there to provide easy entry points for components. Even though they add noise, they simplify imports

app.css file

The role of the app.css file in the ReactJS project is to style the app and components.

- ### Other important files and folders

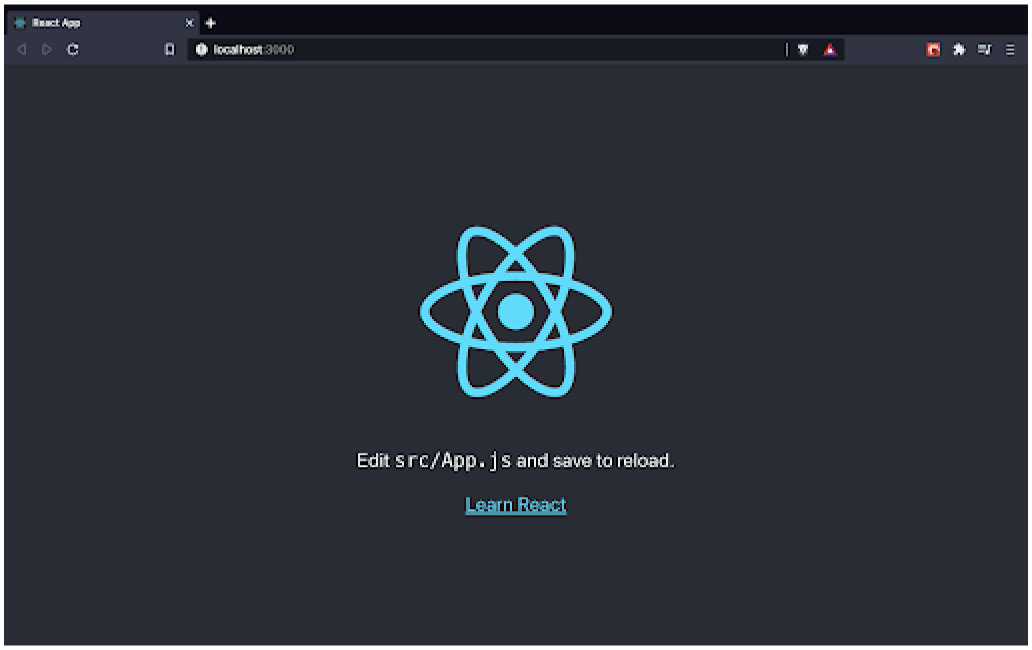
The other files and folders in the ReactJS directory include some assets, components, and containers.

1. The assets contain media files like images, videos, JSON files, and more.
2. Components contain the presentational and stateless components.
3. Containers include the stateful components.

Workflow of React App

Once you have created an app or project using create-react-app, you can view it in the browser. The development server shows up your app on localhost:3000.

Here is how the app looks on the browser:

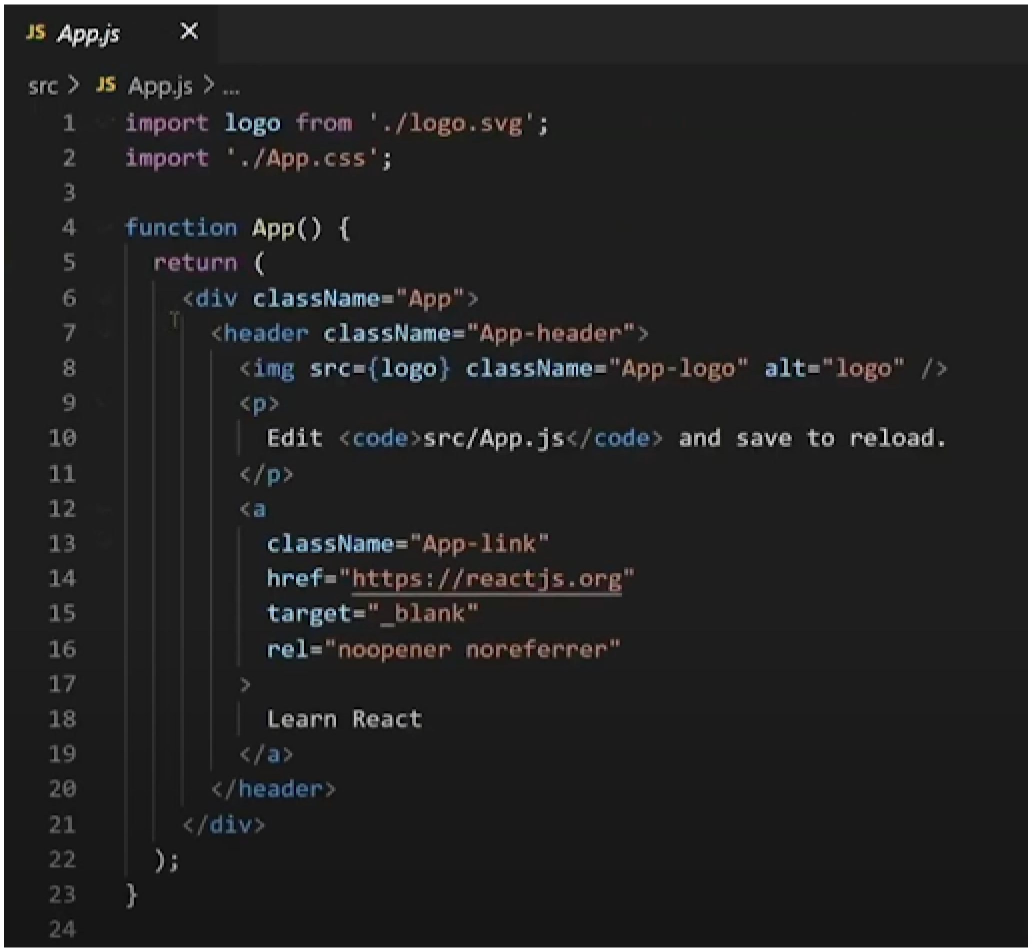


Edit src/App.js and save to reload.

Now, let's understand how a React app works.

- ## Content Fetching

The content in the app is fetched from the App.js file which is located in the src folder. You can make changes to the app from the App.js file.



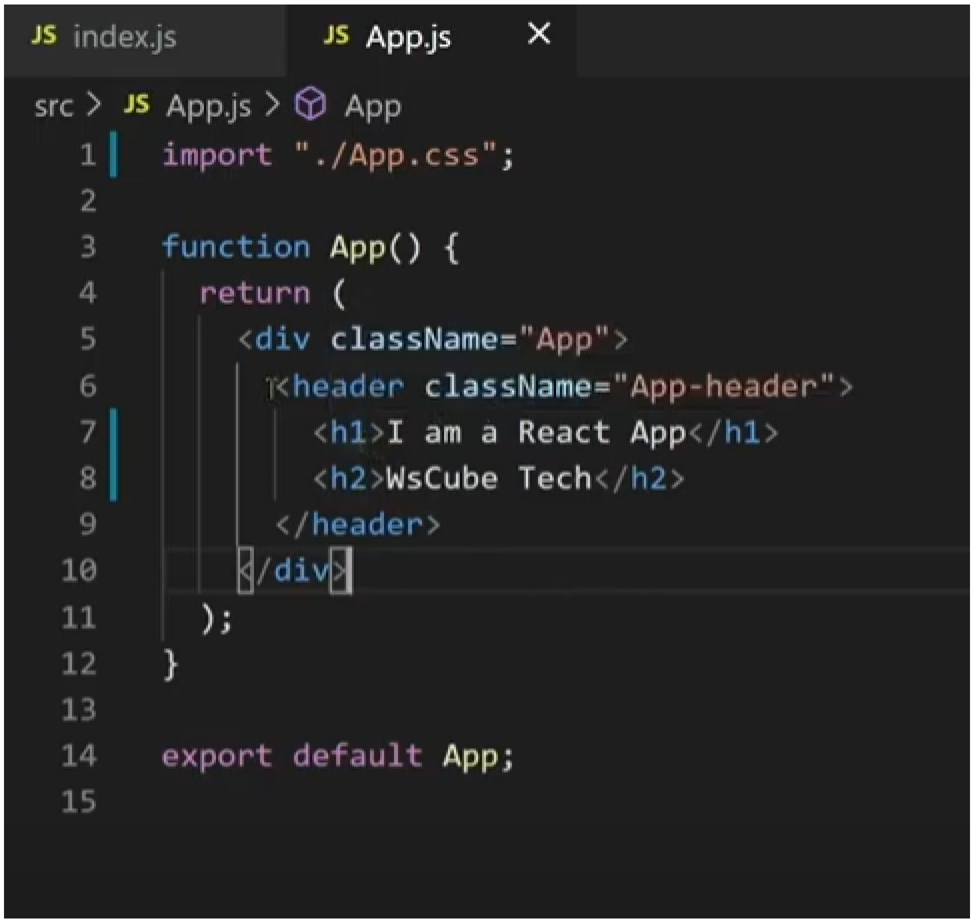
```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

```

Let's try to modify the code in the App.js file and see how it works.

For example, remove the paragraph (p) and hyperlink (a) html tags from the above file and use h1 and h2 heading tags.

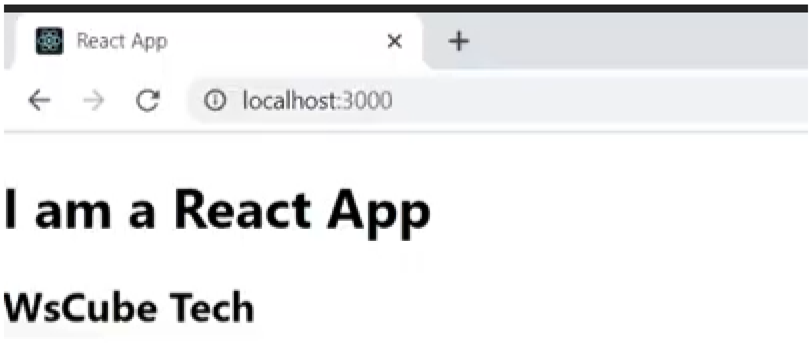
H1: I am a React App

H2: WsCube Tech



```
import "./App.css";

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>I am a React App</h1>
        <h2>WsCube Tech</h2>
      </header>
    </div>
  );
}

export default App;
```

After making the changes, save the updated file using Command/Ctrl + S. The page in the browser will get updated. It will now look like this:



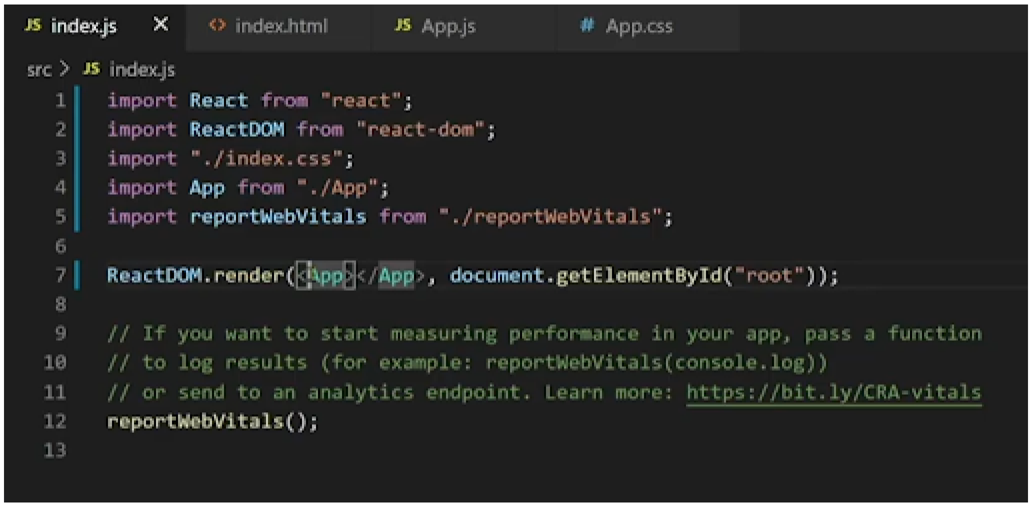
I am a React App

WsCube Tech

You can further make changes to the other classes for changing font size, colors, alignment, and more. On saving, these changes will be replicated automatically on the browser without needing refresh operation from your end.

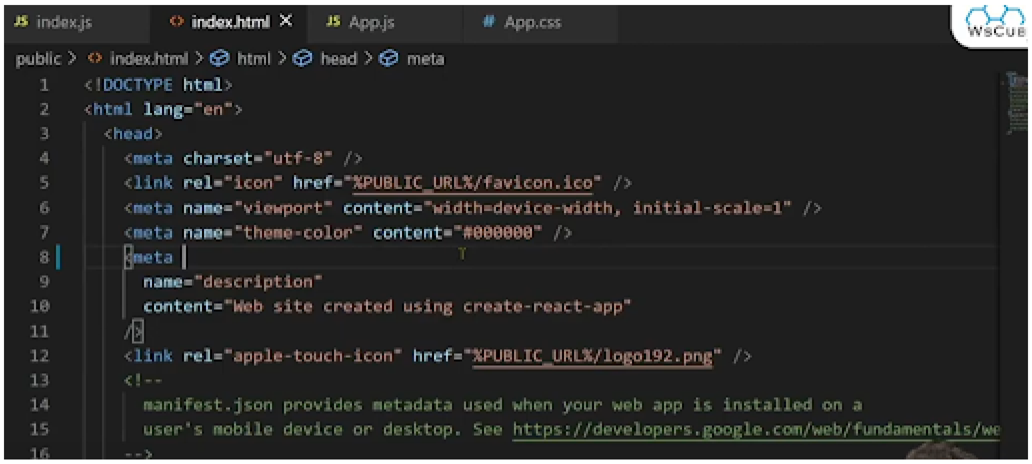
- ## Working of React App

You can understand the workflow of your React application by visiting the index.js file. The ReactDOM package renders the components of the app. It attaches the components to HTML elements with an id value of root, as shown in the screenshot below.



```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";

ReactDOM.render(<App></App>, document.getElementById("root"));

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

This element can be found in the index.html file located in the public folder. Your app is completely attached to this file. The index.html uses div along with the id of the root.



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/we
    -->
```

You don’t need to modify the body tags. If needed, you can make changes to the heading or meta tags for search engine purposes. The meta tags include the tags for title, description, favicon, headings, etc.

These changes will become applicable to your app once you save the updates.

create-react-app is a command to create a React.js project with default configuration. Create-react-app will help in running react applications. Command will be executed on npm or yarn

If npm and node.js is installed on computer, install create-react-app globally with command –

```
npm install –g create-react-app
```

Creation of project – to create a project once above commands are executed, run below command –

```
npx create-react-app hello-world-example
```

npx comes with npm version 5.2+ , npm version can be checked on terminal using npm –version

If npm version is 5.2+, then react.js project can be directly created with command –

```
npx create-react-app hello-world-example
```

If npm version is 6+, npm init react-app hello-world is also an option to create React.js project.

With yarn, we have command – yarn create react-app hello-world-example

Once above commands are done , change directory to hello-world-example

With create-react-app, webpack or babel required for using advanced JavaScript features are preconfigured and we can only concentrate on writing code.

```
cd hello-world-example
```

To execute the application, run the below command on terminal –

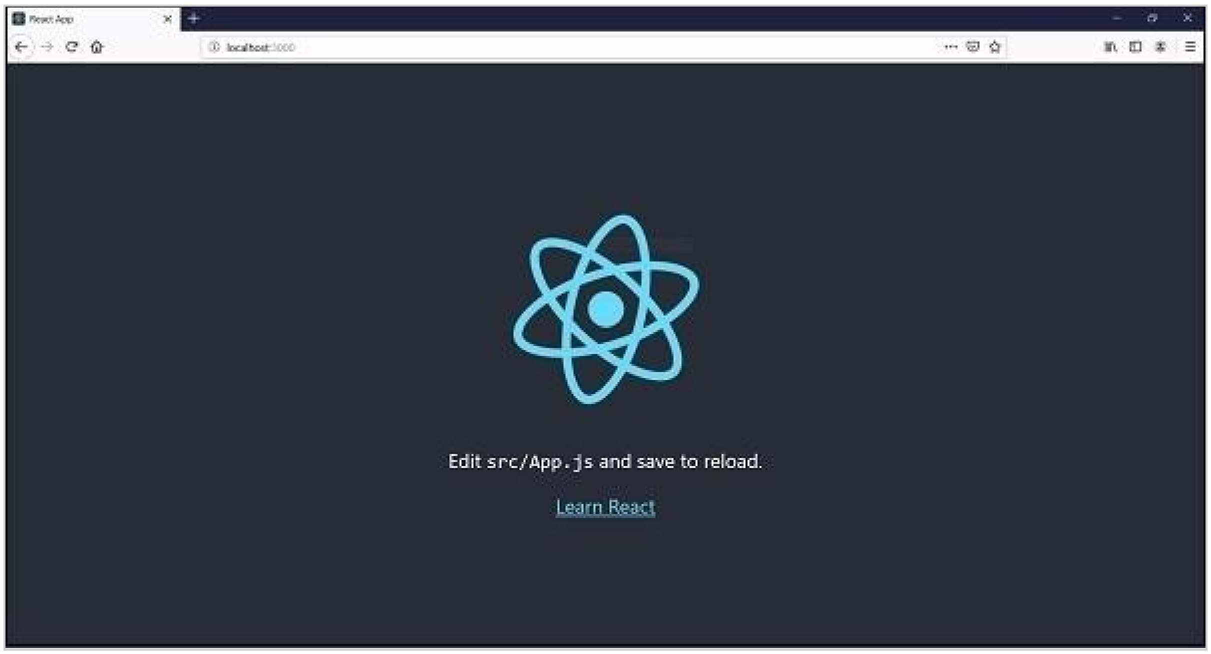
```
npm start
```

npm start runs a live development server and the code changes will automatically refresh the browser and reflect the changes.

A browser window will be opened, if not opened automatically open a browser manually and type url – localhost:3000 in the address bar. 3000 is the default port used in React application. Port number can be changed if any issue with port number.

The default text on application run is shown below –

Display



Edit src/App.js and save to reload.

To update open the project using any code editor tools e.g. visual studio code

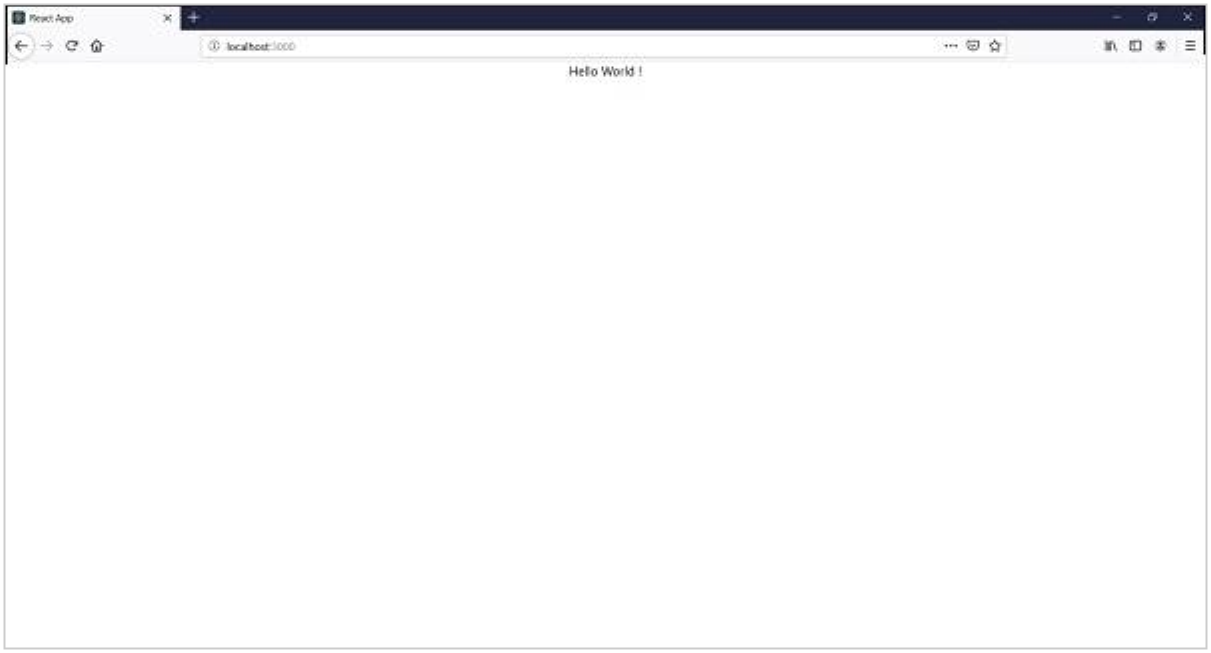
Open the file App.js

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>Edit <code>src/App.js</code> and save to reload.</p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer">
          Learn React
        </a>
      </header>
    </div>
  );
}
export default App;
```

Change the content of return statement to just hello World text –

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
function App() {
  return (
    <div className="App">
      Hello World !
    </div>
  );
}
export default App;
```

The text on the browser will be changed immediately



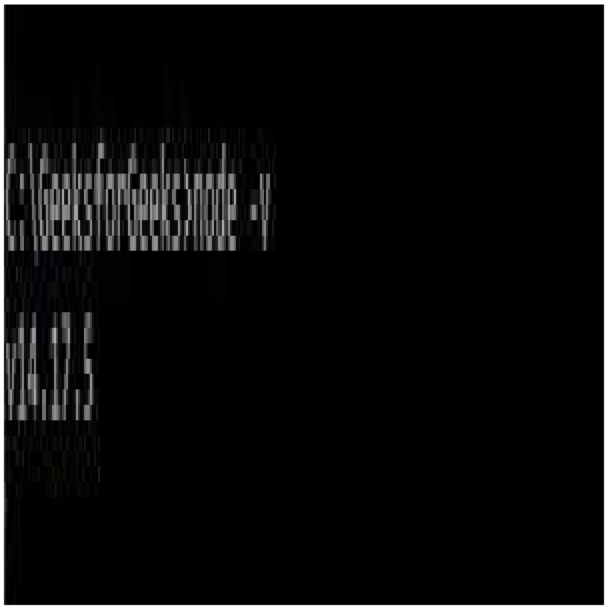
Hello World !

[React](#) is a Javascript Library that was created by Facebook for building better User Interface(UI) web applications and mobile applications. It is an open source library for creating interactive and dynamic applications.

In this article, we will see how to build a basic react app that shows hello world.

- To create a react application, [Node.js](#) version of at least 10 or higher need to be installed on your system. If it is installed you can check by using the following command in your command line.

```
node -v
```



```
C:\GeeksforGeeks>node -v
v14.17.5
```

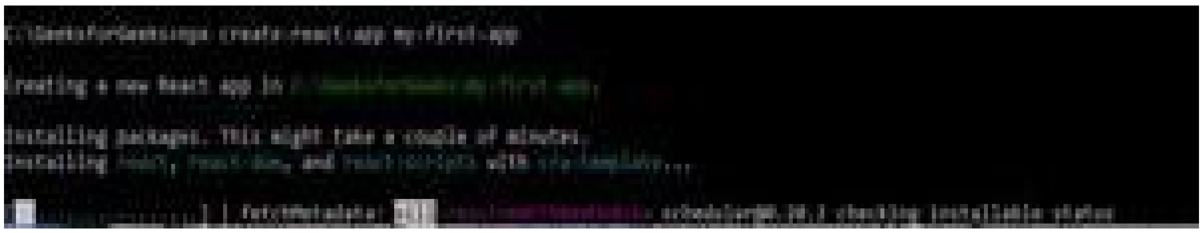
- Make sure you have a code editor for working on your project files.

Let’s create our first react application:

To build a react application follow the below steps:

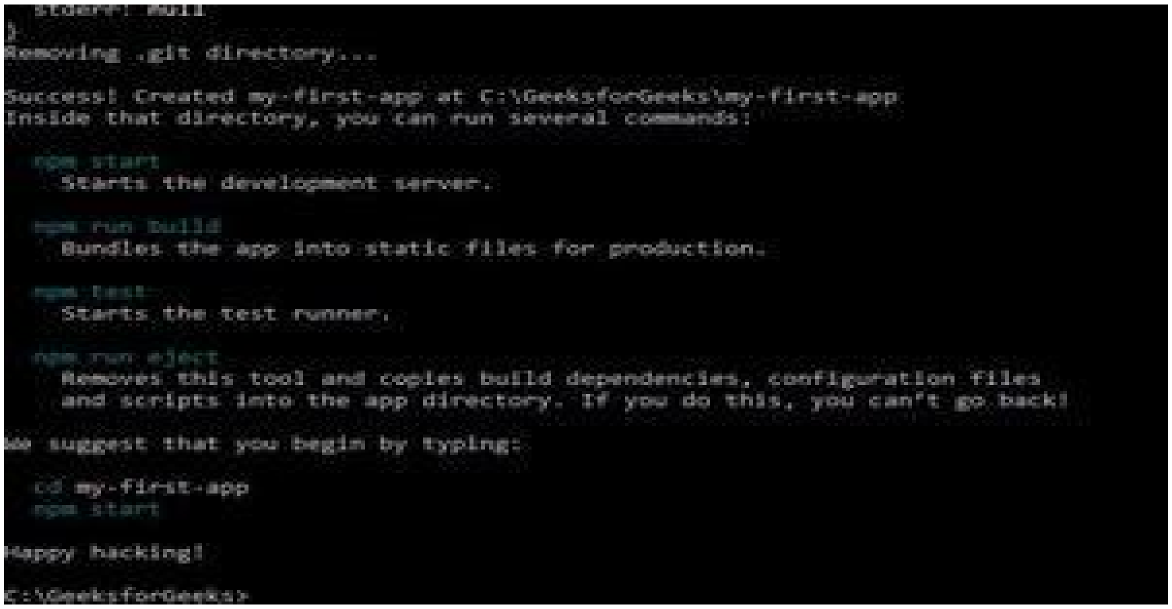
Step 1: Create a react application using the following command

```
npx create-react-app foldername
```



```
C:\GeeksforGeeks>npx create-react-app my-first-app

Creating a new React app in C:\GeeksforGeeks\my-first-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

[illegible] | | fetchMetadata: [illegible] scheduler@0.20.2 checking installable status
```

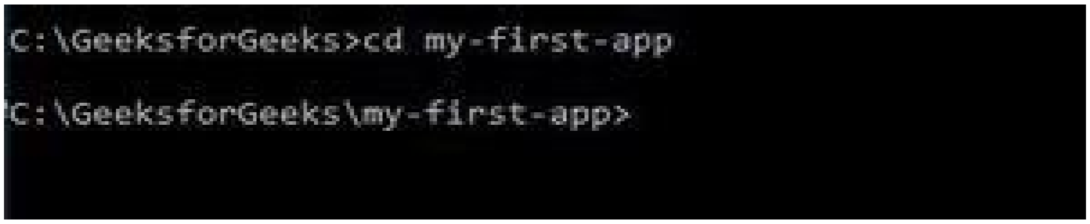
It takes a couple of minutes to install the packages.



```
  stderr: null
}
Removing .git directory...

Success! Created my-first-app at C:\GeeksforGeeks\my-first-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd my-first-app
  npm start

Happy hacking!

C:\GeeksforGeeks>
```

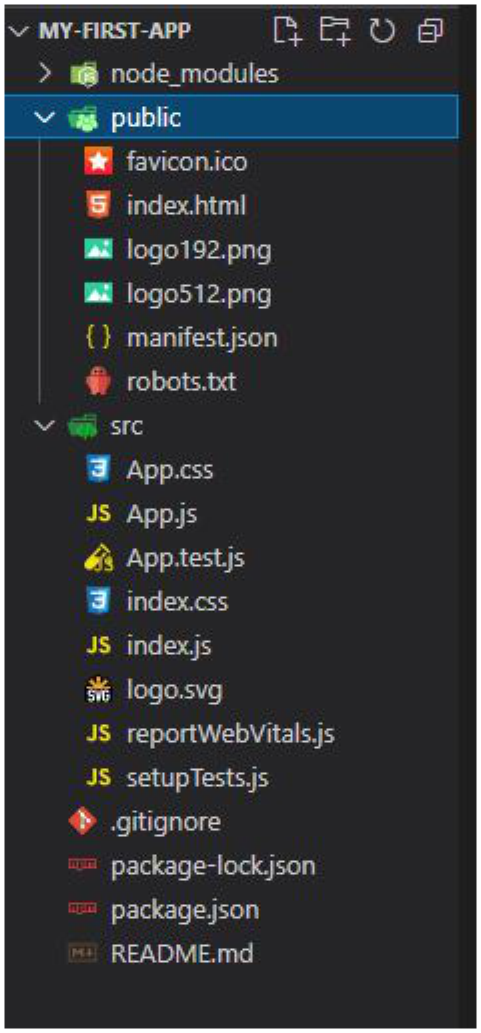
Step 2: Once it is done change your directory to the newly created application using the following command

```
cd foldername
```



```
C:\GeeksforGeeks>cd my-first-app

C:\GeeksforGeeks\my-first-app>
```

Project Structure: It is created as shown below.



```
MY-FIRST-APP
  node_modules
  public
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Project file structure

Step 3: Now inside App.js and write down the following code as shown below:

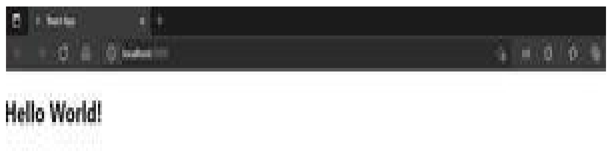
Javascript

```
import React from 'react';

import './App.css';

function App() {

    return (

        <h1> Hello World! </h1>

);

}export default App;
```

Step to run the application: Enter the following command to run the application.

```
npm start
```

Output: You will see the following output in your browser.


Hello World!

Congratulations, you have created your first React app. You have learned something new today. Don’t stop learning Go ahead. Learn [React.js]