**Installation Setup**

* Installation Setup React JS

For setting up React in our local system, the first step is to Install NodeJs and npm.

Install Nodejs

Node.js provides a runtime environment to execute JavaScript code from outside a browser. NPM, the Node package manager is used for managing and sharing the packages for either React or Angular.

NPM will be installed along with Nodejs. Node.js can be downloaded and installed from the official NodeJs website.

https://nodejs.org

Once the Installation of Node is complete. Open Node.Js Command Prompt and we can check the version as well.

Install Create-React-App Tool

The next step is to install a tool called create-react-app using NPM. This tool is used to create react applications easily from our system. You can install this at the system level or temporarily at a folder level. We will install it globally by using the following command.

npm install -g create-react-app

Creating a new react project

After create-react-app is installed, we can create our first react application. Let's say I want to create the project or application in D:\React\_Programs. I will create this folder and let our command prompt point to it by using the change directory command.

Let's create a new Project now using the command.

create-react-app test-project

Remember not to create the project with an upper case character In it.

Running the React Application

Let's do CD to the Project we have created and run it locally on our system using npm start. Launch the browser and visit [http://localhost:3000](http://localhost:3000/). We can then see our first React Application response in the browser.

cd test-project

npm start

We have created a New Project using React and executed the Project.

But as a developer, we would be more interested to know about the Project which is created, its structure and we would like to play around with it. So it is time for us to get an Editor. When we think of IDE, we have a variety of choices like Visual Studio Code, React IDE, Sublime Editor, Atom Editor, Webstorm and a few others. We will use the VS Code as our Editor.

Visual Studio Code is a free IDE from Microsoft built for developing and debugging web applications. It has integrated Git control & terminal.  VS code’s IntelliSense allows Visual Studio Code to provide you with useful hints and auto-completion features while you code. So the next step is to install the Visual Studio Code.

Install Visual Studio Code

Download and install Visual Studio Code from the following URL

<https://code.visualstudio.com/download>

After the installation, open the Project we have created earlier using the VS Code. The Project has the following 3 folders

* Node\_modules
* Public
* src

The output we have seen when the Project is executed comes from a file called Index.html which resides inside the public folder.

In index.html we have one div tag with id as root.

<div id="root"></div>

To understand the relation between the output we see and this index.html, Open src/app.js file. The image and the text we see in the browser are coming from here. Let's make a small change in the text, save it and let's have a look at the browser. We can see the changes and it happens very fast.

How the index.html is linked to App.js will be discussed in our upcoming videos. With this we have the react environment setup on our local machine and we are ready to explore React.

React online editors

Let's say we are in office, we have some free time and we’re interested in playing around with React, then you can use an online code playground like  CodePen, CodeSandbox, or Glitch.

For example, let's say we want to create react project using CodePen. In the browser, navigate to https://codepen.io/ and click on Start Coding.

Create a simple div in HTML section.

<div id="root"></div>

Followed by writing some JavaScript Code :

ReactDOM.render(

<h1>Welcome to React World</h1>,

document.getElementById('root')

);

This code will throw an error as we are missing the references to two Javascript files.

Go to Pen Settings section of Js and add,

https://unpkg.com/react/umd/react.development.js

https://unpkg.com/react-dom/umd/react-dom.development.js

One script file refers to React and the other refers to ReactDOM which is the Virtual DOM introduced by React. Set the Javascript Preprocessor to Babel.

With the above settings, you should have the output produced as expected.

Babel is a free and open-source JavaScript transcompiler that is mainly used to convert ECMAScript 2015+ code into a backwards-compatible version of JavaScript that can be run by older JavaScript engines. Babel is a popular tool for using the newest features of the JavaScript programming language. More about Babel will be discussed in our upcoming videos.

I hope we are clear on doing the React setup and creating our first Project using React.

* React JS environment Setup Using the Create-react-app Command

Instead of using webpack and babel you can install ReactJS more simply by installing create-react-app.

**Step 1 - install create-react-app**

Browse through the desktop and install the Create React App using command prompt as shown below −

C:\Users\Tutorialspoint>cd C:\Users\Tutorialspoint\Desktop\

C:\Users\Tutorialspoint\Desktop>npx create-react-app my-app

This will create a folder named my-app on the desktop and installs all the required files in it.

**Step 2 - Delete all the source files**

Browse through the src folder in the generated my-app folder and remove all the files in it as shown below −

C:\Users\Tutorialspoint\Desktop>cd my-app/src

C:\Users\Tutorialspoint\Desktop\my-app\src>del \*

C:\Users\Tutorialspoint\Desktop\my-app\src\\*, Are you sure (Y/N)? y

**Step 3 - Add files**

Add files with names index.css and index.js in the src folder as −

C:\Users\Tutorialspoint\Desktop\my-app\src>type nul > index.css

C:\Users\Tutorialspoint\Desktop\my-app\src>type nul > index.js

In the index.js file add the following code

import React from 'react';

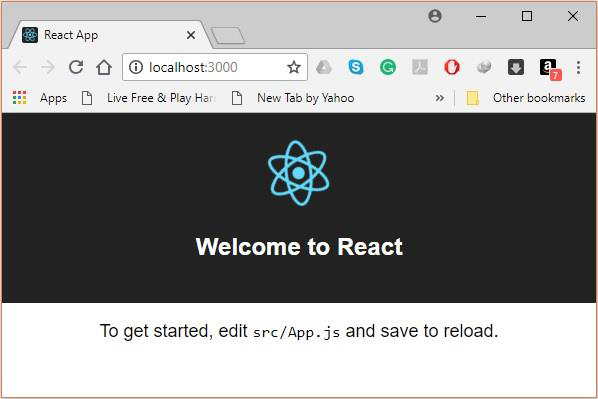
import ReactDOM from 'react-dom';

import './index.css';

**Step 4 - Run the project**

Finally, run the project using the start command.

npm start



* Node Package Manager

**NPM (Node Package Manager)** is the default package manager for Node.js and is written entirely in [Javascript](https://www.geeksforgeeks.org/javascript-tutorial/). Developed by Isaac Z. Schlueter, it was initially released in January 12, 2010. NPM manages all the packages and modules for Node.js and consists of command line client **npm**. It gets installed into the system with installation of Node.js. The required packages and modules in Node project are installed using NPM.  
A package contains all the files needed for a module and modules are the JavaScript libraries that can be included in Node project according to the requirement of the project.  
NPM can install all the dependencies of a project through the [package.json](https://www.geeksforgeeks.org/node-js-package-json/" \t "https://www.geeksforgeeks.org/node-js-package-json/) file. It can also update and uninstall packages. In the [package.json](https://www.geeksforgeeks.org/node-js-package-json/" \t "https://www.geeksforgeeks.org/node-js-package-json/) file, each dependency can specify a range of valid versions using the semantic versioning scheme, allowing developers to auto-update their packages while at the same time avoiding unwanted breaking changes.

**Some facts about NPM:**

* At the time of writing this article, NPM has 580096 registered packages. The average rate of growth of this number is 291/day which outraces every other package registry.
* npm is open source
* The top npm packages in the decreasing order are: lodash, async, react, request, express.

**Installing NPM:**  
To install NPM, it is required to install Node.js as NPM gets installed with Node.js automatically.  
[Install Node.js](https://nodejs.org/en/).

**Checking and updating npm version:**  
Version of **npm** installed on system can be checked using following syntax:  
**Syntax:**

npm -v

* Default vs Named Exports

The export statement is used when creating JavaScript modules to export objects, functions, variables from the module so they can be used by other programs with the help of the import statements. There are two types of exports. One is Named Exports and other is Default Exports.

|  |  |
| --- | --- |
| Named Exports | Default Exports |
| Named exports are useful to export several values. During the import, it is mandatory to use the same name of the corresponding object.  Example:  //file math.js  function square(x) {  return x \* x;  }  function cube(x) {  return x \* x \* x;  }  export { square, cube };  //while importing square function in test.js  import { square, cube } from './math;  console.log(square(8)) //64  console.log(cube(8)) //512  Output:  64  512 | Default exports are useful to export only a single object, function, variable. During the import, we can use any name to import.  Example:  // file module.js  var x=4;  export default x;  // test.js  // while importing x in test.js  import y from './module';  // note that y is used import x instead of  // import x, because x was default export  console.log(y);  // output will be 4  Output:  4 |

* Lets Start Coding Hello World Program

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>

   );

}

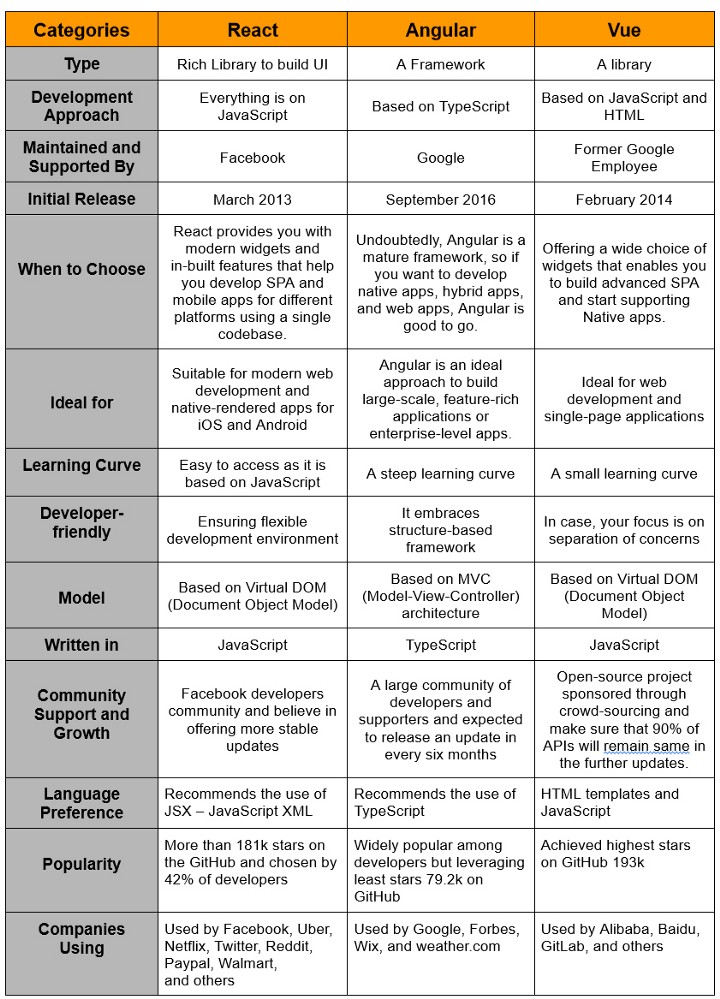
export default App;

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

npm start

**Output:** Hello World !

* React vs Angular vs Vue



* How to Connect React JS with NodeJS

[ReactJS](https://www.geeksforgeeks.org/react-js-introduction-working/) is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It is widely used for making SPA(Single Page Application) and it has a large developer community.

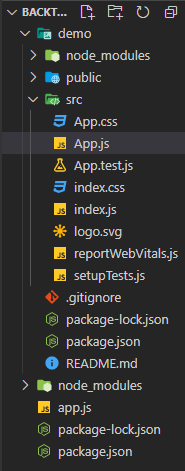
[NodeJS](https://www.geeksforgeeks.org/introduction-to-nodejs/) is primarily used for non-blocking, event-driven servers, due to its single-threaded nature. It’s used for traditional websites and back-end API services but was designed with real-time, push-based architectures in mind.

In this article, we will learn how to connect NodeJS as a backend with ReactJS as a frontend.

**Prerequisites:**

* Basic knowledge of React and Node.
* Node.js installed (version 12+).
* npm installed (version 6+).

**Project Structure:**This is the structure when all the modules and required files are ready.



**Backend setup:** Firstly we will work on our backend(NodeJS) portion. In your working folder make a file named app.js for NodeJS and package.json file to run all the modules we required.

**Installing Module:**

* Installing ExpressJS is a nodeJS framework:

npm install express

* Installing nodemon:

npm install nodemon

**Configuration of package .json file:**Add the start and dev script, which are important for running and dynamically running the code after changes made in your Node.js app respectively in**package.json** file as shown below.

{

"name": "demoapp",

"version": "1.0.0",

"description": "",

"main": "app.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node app.js",

"dev": "nodemon app.js"

},

"author": "",

"license": "ISC",

"dependencies": {

"express": "^4.17.1"

}

}

**Filename- app.js:**Here is the simple JavaScript code that should be written in app.js which is for NodeJS.

const express = require("express");

const app = express();

app.get("/", (req, res) => {

res.send("Hello World!");

});

const PORT = process.env.PORT || 8080;

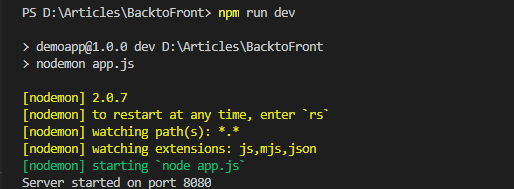
app.listen(PORT, console.log(`Server started on port ${PORT}`));

Run the application using the following command:

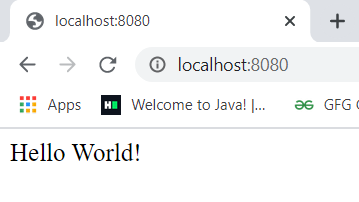
npm run dev

**Output:**

* This will be a console output.



Now go to **http://localhost:8080/** in your browser, you will see the following output.



**Filename- app.js:**Now for connecting the React part we have to make some changes in the app.js of NodeJS. We have completed the backend part.

const express = require("express");

const app = express();

app.post("/post", (req, res) => {

console.log("Connected to React");

res.redirect("/");

});

const PORT = process.env.PORT || 8080;

app.listen(PORT, console.log(`Server started on port ${PORT}`));

**Frontend setup:**First, we have to create React app and run your app by writing the below command.

npx create-react-app demo

cd demo

npm start

**Output:**Now go to**http://localhost:3000/** in your browser, you will see the following output.



**Connecting:** We have completed both the frontend and backend parts, now we have to connect both. Now for connecting Reactjs with Nodejs we have added this line in package.json of react app folder:

"proxy": "http://localhost:8080

**Filename- package.json:** The package.json file is in your React app folder. This tells React to proxy API requests to the Node.js server built with Express in our project.

{

"name": "demo",

"version": "0.1.0",

"private": true,

"proxy": "http://localhost:8080",

"dependencies": {

"@testing-library/jest-dom": "^5.11.4",

"@testing-library/react": "^11.1.0",

"@testing-library/user-event": "^12.1.10",

"react": "^17.0.1",

"react-dom": "^17.0.1",

"react-scripts": "4.0.2",

"web-vitals": "^1.0.1"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

},

"eslintConfig": {

"extends": [

"react-app",

"react-app/jest"

]

},

"browserslist": {

"production": [

">0.2%",

"not dead",

"not op\_mini all"

],

"development": [

"last 1 chrome version",

"last 1 firefox version",

"last 1 safari version"

]}}

**Filename- App.js:**Made some changes in app.js of React to show that they are connected.

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>A simple React app.....</p>

<a

className="App-link"

href="https://reactjs.org"

target="\_blank"

rel="noopener noreferrer"

>

Learn React

</a>

<form action="../../post" method="post"

className="form">

<button type="submit">Connected?</button>

</form>

</header>

</div>

);}

export default App;

Now run the Nodejs process **npm run** **dev**in one terminal and in another terminal start Reactjs using **npm start** simultaneously**.**

**Output:**We see react output we see a button “Connect” we have to click it. Now when we see the console server-side we see that the ReactJS is connected with NodeJS.

