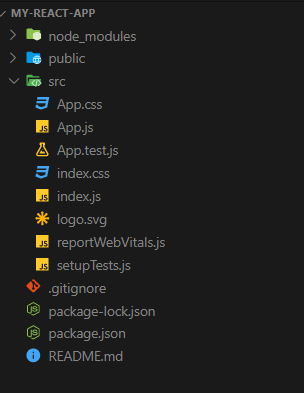
**REACT.JS**

**ReactJS is an open-source, component based front end library responsible only for the view layer of the application. It is maintained by Facebook.**

**ReactJS uses virtual DOM based mechanism to ﬁll in data (views) in HTML DOM. The virtual DOM works fast owning to the fact that it only changes individual DOM elements instead of reloading complete DOM every time**



**What is node modules folder in React?**

In simple word node\_modules folder is **the repository of modules/library which you are using inside your project**. What ever you are importing in your project that module or library should present inside the mode\_module folder.

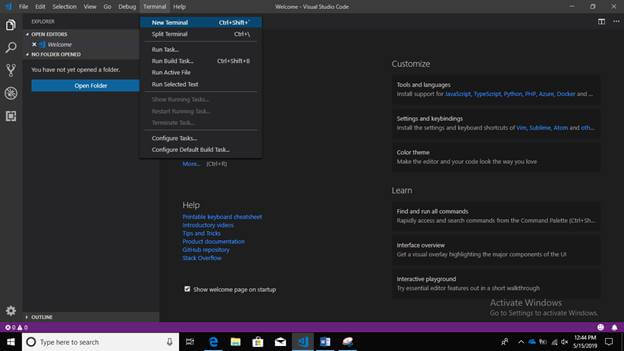
**What is the Public folder in React?**

The public folder contains the HTML file so you can tweak it, for example, **to set the page title**. The <script> tag with the compiled code will be added to it automatically during the build process.

Creating React applications

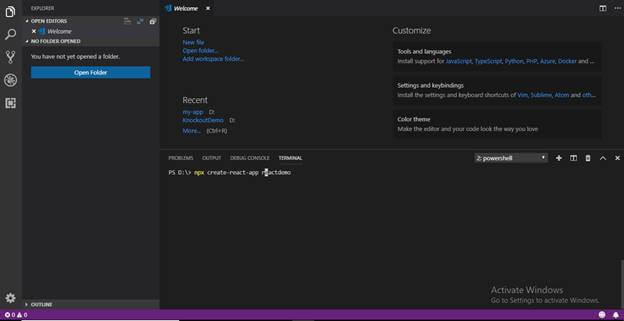
In order to create a React application, we will be using the *create-react-app node\_module* provided by React. This helps us by creating a consistent folder structure automatically so, that we don’t need to care about proper and valid folder structure. For this demo, I will be using Visual Studio Code.

Open Visual Studio Code > go to Terminal > open new terminal.

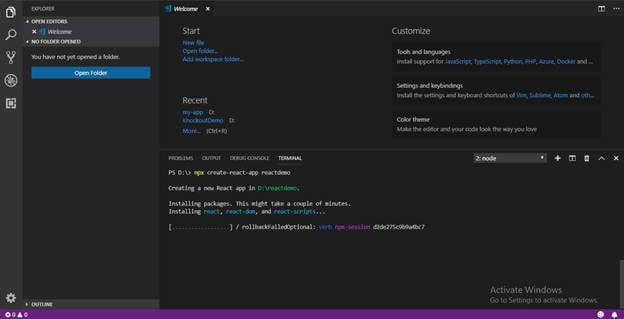


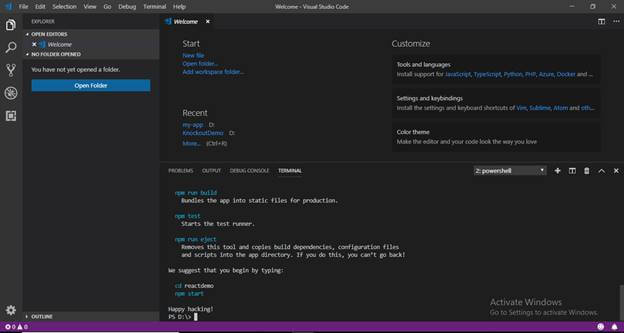
Write the following command in Terminal and press Enter.

*npx create-react-app ReactDemo*



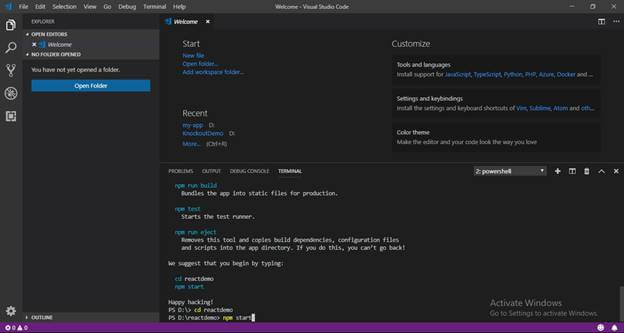
After executing the command, it will take some time to create a React application and its related dependencies.





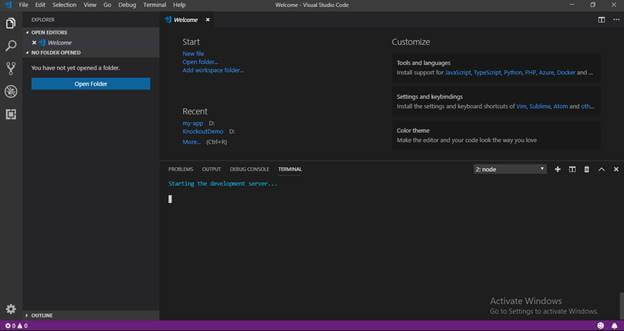
After successfully creating a React application, you can use the *"cd reactdemo"* command to navigate to the React application. To start running the React application, use the following command.

*npm start*

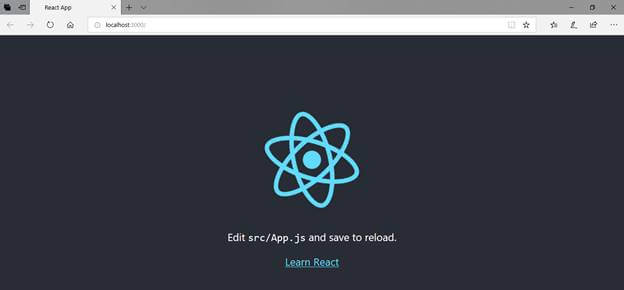


After creating a React application, we will notice that in Terminal, it provides quick tips or commands that will help in executing the React application. For example -

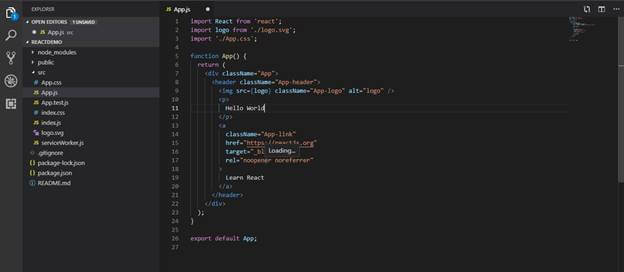
* npm start - This command will start the development server.
* npm run build - Bundles the app into static files for production.
* npm test - starts the test runner.
* npm run object - Removes this tool and copies build dependencies, configuration files, and scripts into the app directory. If you do this, you can’t go back!

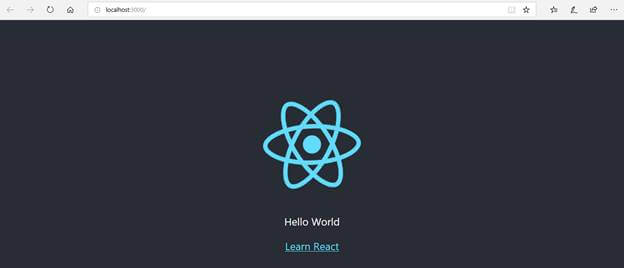


After "npm start" command, it will open the application in a browser on port number 3000.



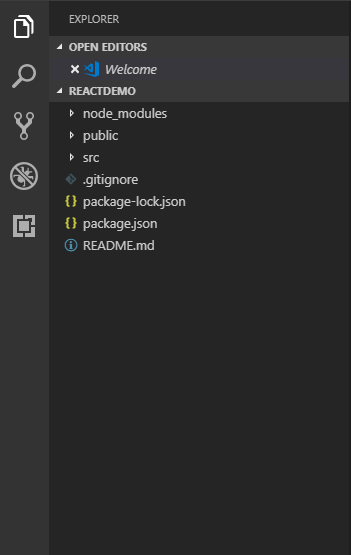
Now, open App.js and change the code as "Hello world" and refresh the browser. It will display "Hello world" on the screen.



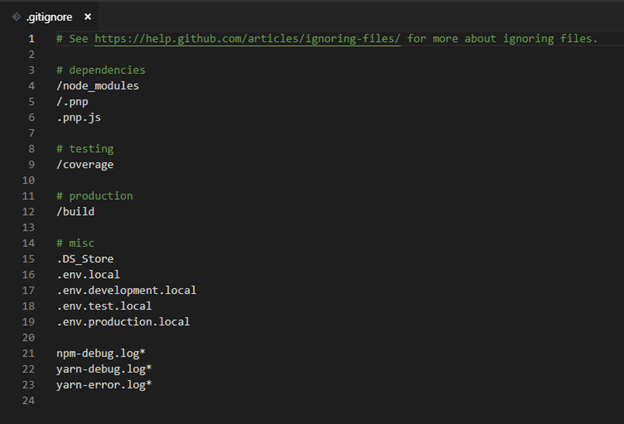


Folder Structure

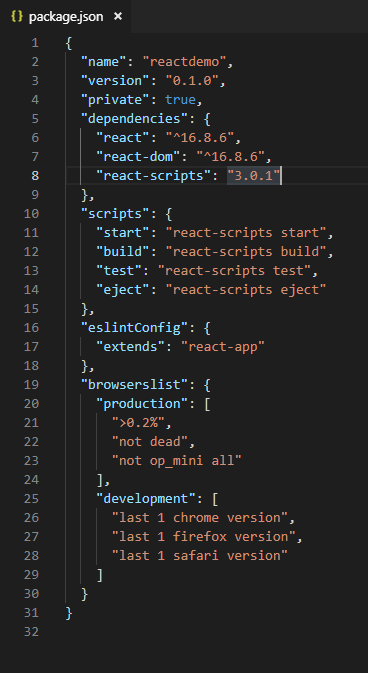
The React application automatically creates required folders, as shown below.



.gitignore - This file is the standard file which is used by source control tool git to identify which files and folders are need to be ignored while committing the code. Until and unless this file exists, the create-react-app command will not create a git repo in this folder.



package.json – This file contains dependencies and scripts required for the project.



This file contains all settings for React applications including:

1. name - points to name of your app.
2. version - refers to the current version that the application is using.
3. “private”: true - is a foolproof setting which avoids accidentally publishing of your react app as a public package in npm ecosystem.
4. Dependencies will contain all required node modules and versions required for the application. By default, 2 dependencies are added which include React and React-Dom that allow using JavaScript. In our demo, we are using React version 16.8.
5. Scripts specify aliases that can be used to access some React command in a more efficient manner.
   * package-lock.json contain exact dependency tree to be installed in /node\_modules. It helps while a team is working on private apps to ensure that they are working on the same version of dependencies and sub-dependencies. It also maintains a history of changes done in package.json so, that at any point of time, when required previous changes can be looked back in the package-lock.json file.
   * node\_modules - This folder contains all dependencies and sub-dependencies specified in package.json used by React app. It contains more than 800 subfolders, this folder is automatically added in the .gitignore file.
   * public - This folder contains files which don’t require additional processing by webpack. The index.html file is considered as an entry point for the web application. Here, the favicon is a header icon and manifest.xml file contains configuration when your application is used for Android app.
   * src - This folder is the heart of React application as it contains JavaScript which needs to be processed by webpack. In this folder, there is a main component App.js, its related styles (App.css), test suite (App.test.js). index.js, and its style (index.css); which provide an entry point into the App. Lastly, it contains registerServiceWorker.js which takes care of caching and updating files for the end user. It helps in offline capability and faster page loading after the first visit.

After all this, we add /Component folder in src to add our custom component and its related files and /Views folder for adding React views and its related files.

Summary

In this article, we have learned to create a React application using *create-react-app* and its folder structure was created. In the next article, we are going to cover the basics of components and their types and usage in React applications. If you have any questions or feedback, please feel free to mention it in the comment section.

Component

In React, a component is referred to as an isolated piece of code which can be reused in one or the other module. The React application contains a root component in which other subcomponents are included; for example - in a single page, the application is subdivided into 3 parts - Header, Main Content, and Footer. So, there is a single App Component having 3 subcomponents - Header Component, MainContent Component, and Footer Component.

There are 2 types of components in React.js

* Stateless Functional Component
* Stateful Class Component

Stateless Functional Component

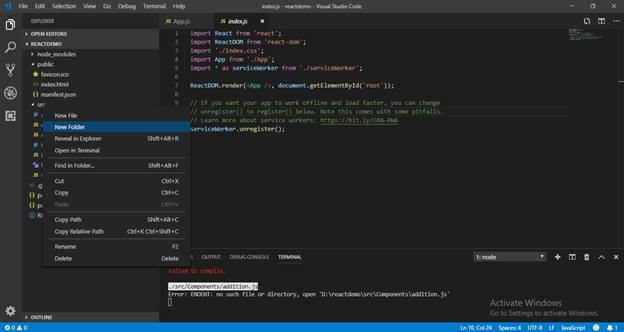
This type of component includes simple JavaScript functions. These components include immutable properties, i.e., the value for properties cannot be changed. We need to use Hooks (will be discussed in the next article) to achieve functionality for making changes in properties using JS. A functional component is used mainly for UI.

**Example**

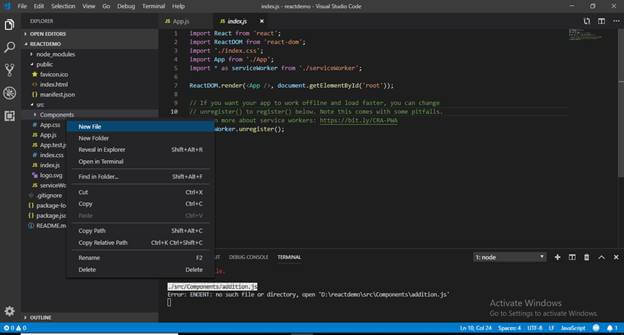
1. **function** Demo(props) {
2. **return** <h1> Hello, {props.Name} </h1>;
3. }

**Demo**

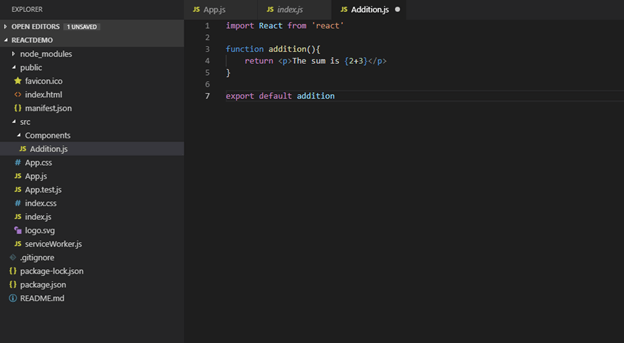
Open App.js and remove additional HTML code except for the div. Now, add a new folder named Components and add a file named addition.js.



Create a new file named addition.js.

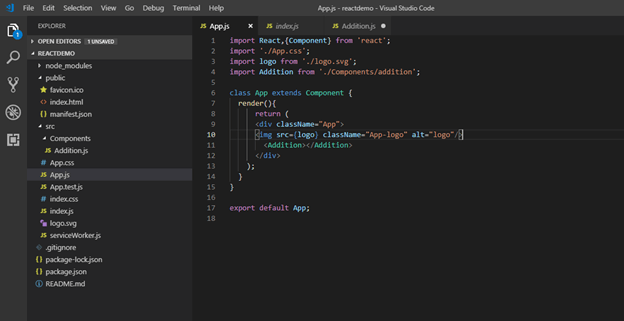


Add Stateless functional component code.



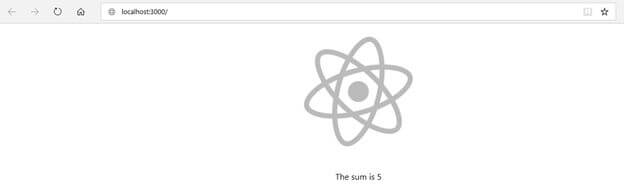
Please note in Addition.js, the last line exports default addition that is a must. Only through this line, it can be used in the parent component.

Now, again in App.js, import Addition from ‘./Components/Addition’. After importing, we can use the imported component as tag <Addition></Addition>.



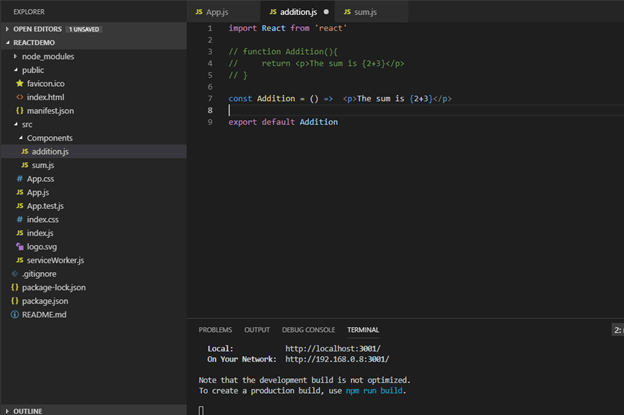
Now in terminal, run the command npm start.

This will run the React application and in the browser, the result will be displayed as below.

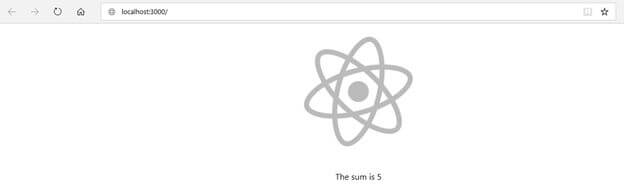


So, this way, we can call the functional component.

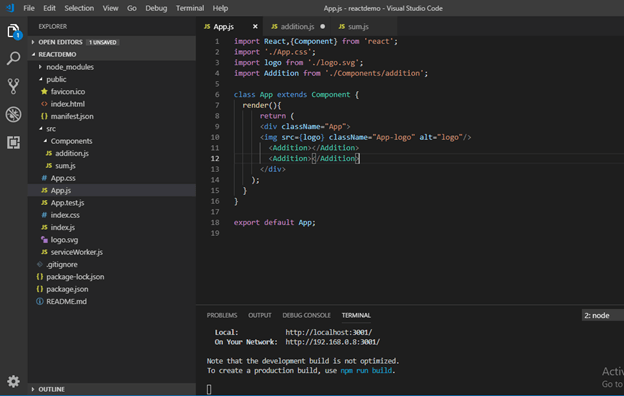
In another way, we can write addition function as arrow function like in the snap below.

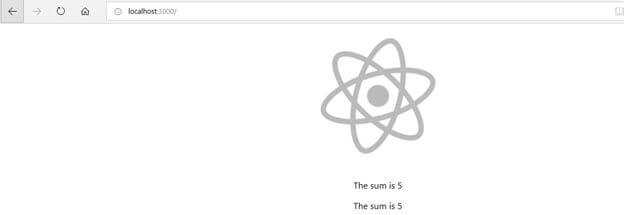


The output will remain the same,



We can add as many tags for the additional component. It will be displayed in the browser.





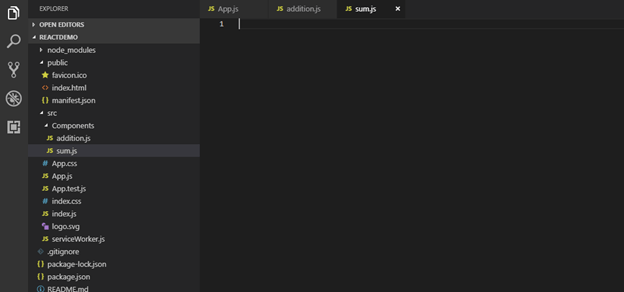
Stateful Class Component

The Class component is ES6 classes which extend the Component class from React library. The class component must include the render method which returns HTML.

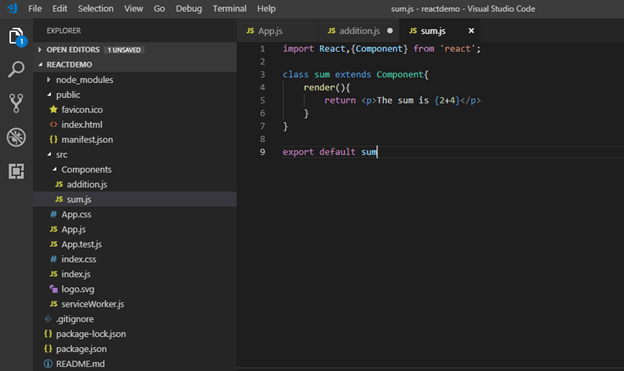
1. **class** Demo **extends** React.Component{
2. render(){
3. **return** <h1> Hello, {props.Name} </h1>;
4. }
5. }

**Demo**

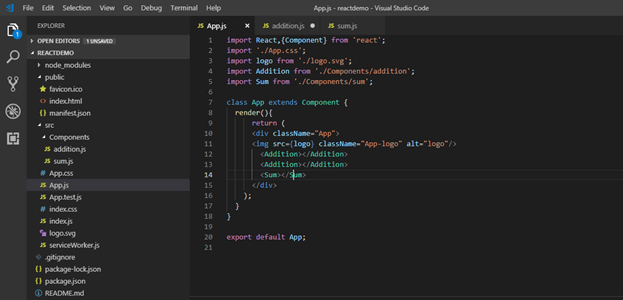
For class component, create another JS file named sum.js in the component folder.



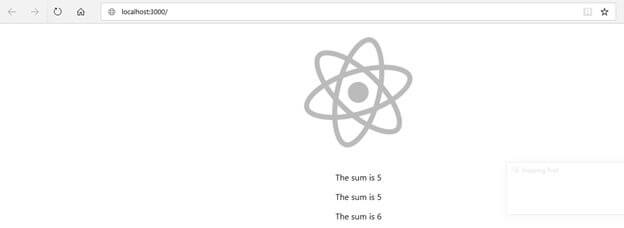
Now, import React and component from the React library. After that, declare the class extending component.



Now, import the exported sum class to App.js.



Save file it will automatically compile and result will be updated in the browser,



**Important Point**

* You can use it without exporting/importing class/functional component.
* The imported class/function must be in upper case; else it will give an error.
* Class/Functional component returning JSX i.e) Javascript XML.

Difference between Stateless Functional and Stateful Class Component

|  |  |
| --- | --- |
| Stateless Functional Component | Stateful Class Component |
| These are a simple javascript function | These classes extend the Component class from React library |
| Stateless function uses Hooks to change properties value in View | Stateful class uses State and setState to change properties value |
| Stateless doesn't use ‘this’ keyword | Stateful component provide complex UI data |

Summary

In this article, we have reviewed about components and their importance in React applications. Components are building a block of React js and can be reused and nested within other components. They play a very important role in React applications. React provides 2 types of components, Stateless Functional, and Stateless Class Component.

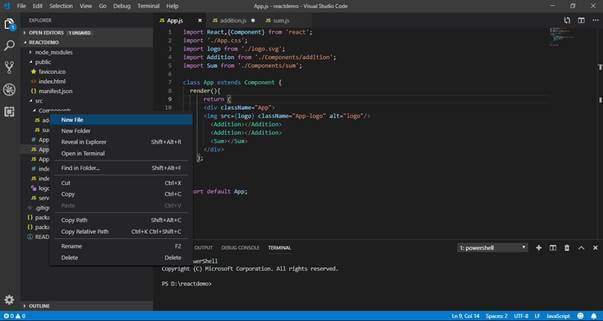
What is JSX?

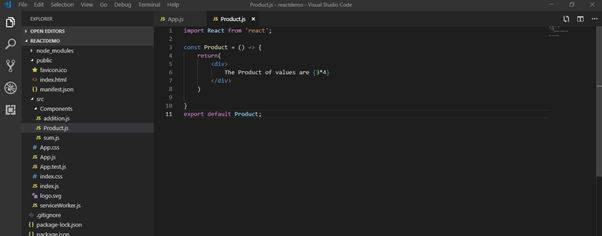
JSX is a JavaScript XML used in React applications. JSX is not compulsory to use in React applications but it makes your code more readable, reliable, and easy to modify. JSX is an extension to JavaScript.

JSX uses HTML syntax to create elements and components. It is just like a templating syntax. It has the tag name, attributes, and children. JSX compiles the code into pure JavaScript which can be understood by the browser.

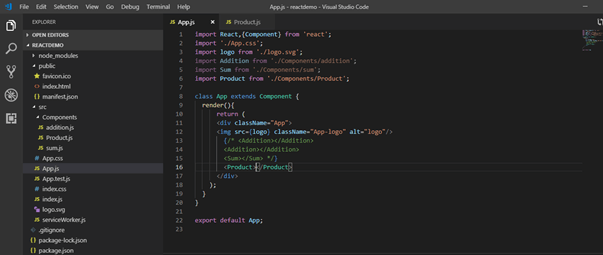
Example of JSX Code

Create a new file in a React demo app, named Product.js. Add code for displaying the product values as text.

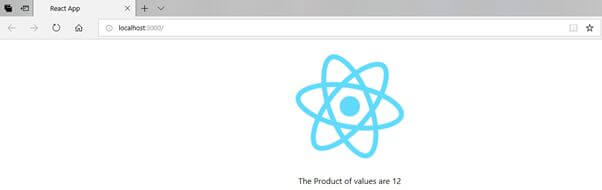




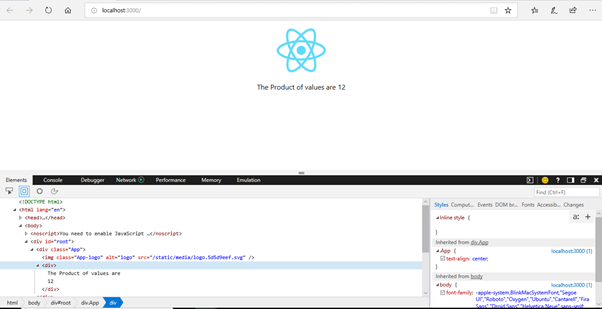
After exporting Product.js, import into App.js and define the <Product> tag in App.js.



The result will be displayed as below.



Now, the result in console will be displayed as below.



The above code is as per JSX syntax. Now, we will review how the code looks and rendered by the browser when we don’t use JSX.

Another way to create the React application, i.e., without JSX

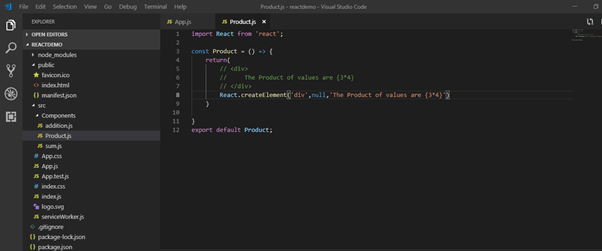
The Non-JSX version of the code which will be used in React.createElement() method is provided by React library. When the code is written using JSX, it is compiled in React.createElement() method.

React.createElement() method takes 3 arguments.

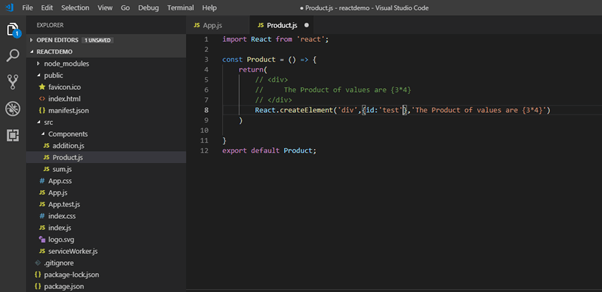
* type - It includes the type of elements we need to create.
* props - It includes the key-value pair value for properties and attributes defined for the element provided in type.
* children - It includes the child element to be created between then DOM element provided.

**Example of React app without JSX code**

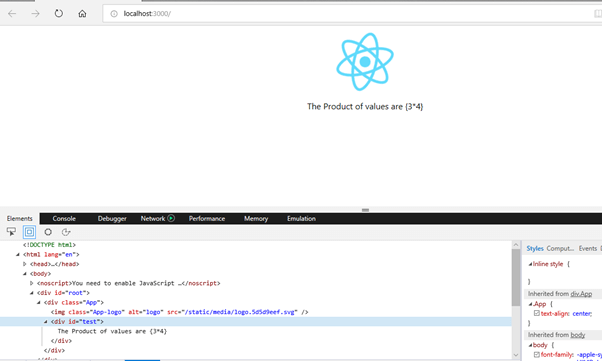
Update the code as below by using React.createElement() method.



In the current situation, the values are not evaluated as they are by using JSX. Now, we are adding the key-value pair for defining id using createElement() method.

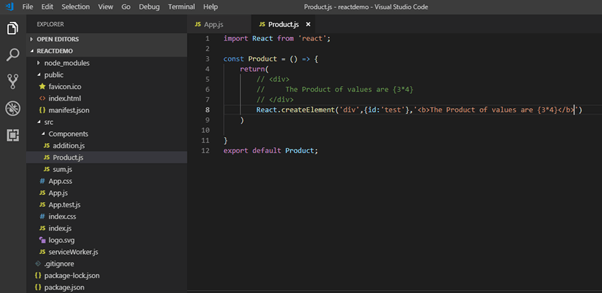


Let us look at the output in the browser console.

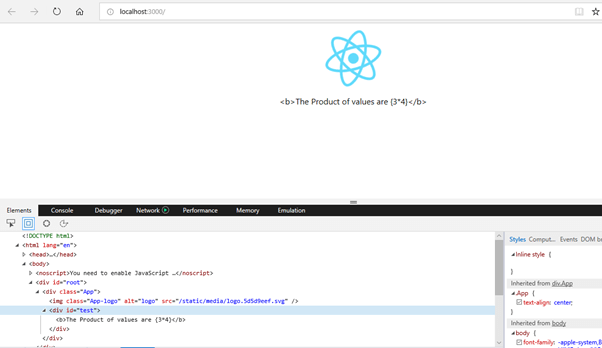


As per the output, we reviewed that the id is defined to div class.

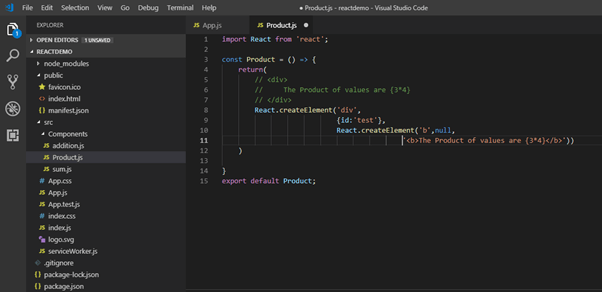
Now, let’s make the text bold in Non-JSX code.



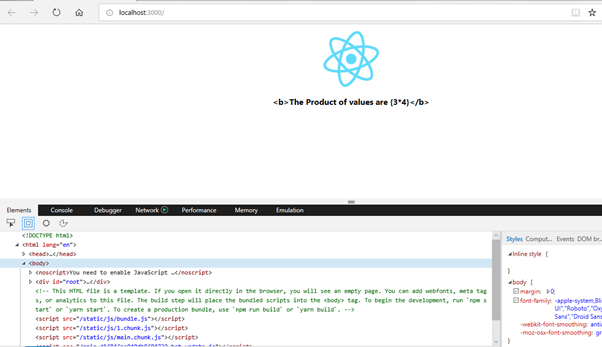
The output will be shown as below.



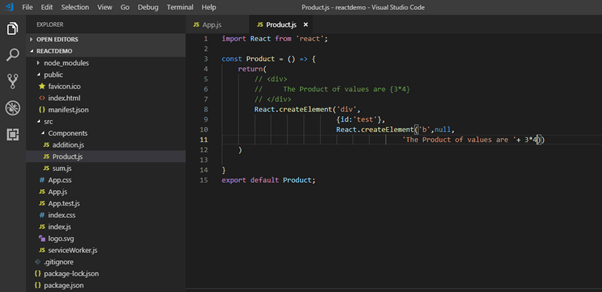
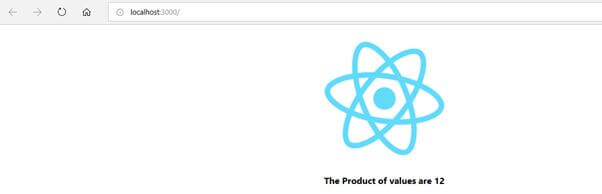
Now, to create a proper output, we need to add a nested React.createElement() method.



Now, the output will be displayed in the browser as below.



To display the calculated value using Non-JSX code, we need to remove the curly braces. Then, the calculated values will be displayed in the browser.

Hooks

Hooks are newly added in React 16.8 as a feature which doesn’t need to add classes to maintain a state which was possible in Stateful class component but was not possible in Stateless Functional component.

Hooks allow using State and other React features without creating a class.

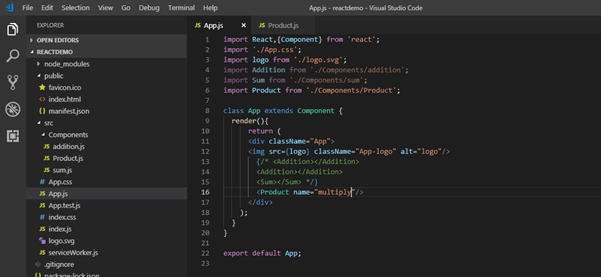
Before going ahead, it is important to know that the Hooks feature doesn't break any previous feature, any component can be modified without breaking any other component. It is backward compatible so no other changes are required to use Hooks. We will go in details of hooks in a further article.

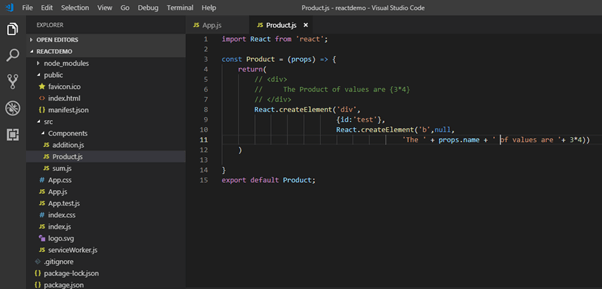
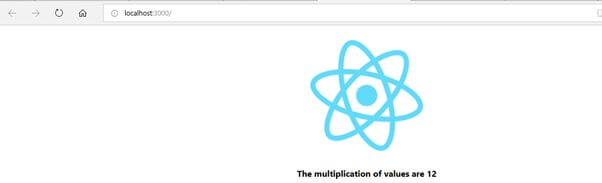
props

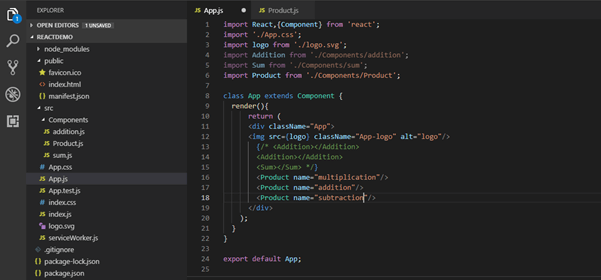
"props" is the short form for properties used in React application. "props" contains a single value or objects in key-value pair having a set of values, that is used, similar to HTML tag attributes, when passing to React Component while creating.

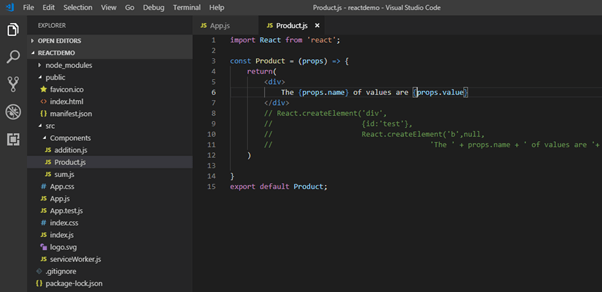
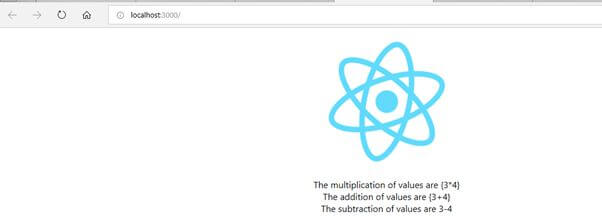
**Example**

When passing the attribute from a Component tag in App.js.





Currently, values are displayed the same because "props" is immutable and its value cannot be changed in the component. To achieve this functionality, we have a new feature named State which we will discuss in the next article.

**Notes**

* JSX uses camel case while using method, tag name, or attributes; just like - createElement() or using attribute for className in createElement() method.
* While creating a new class component, if we are returning more than 1 element, it should be bound under the main div as it can return only 1 element.
* props are immutable and their value cannot be changed. If we assign props a new value in component, it will not change though it will return a runtime error.