**React Js for beginners (just to get you started)**

Is react a framework or a library? Well, don’t bother. Reactjs.org calls it a library (collection of precompiled objects based on a programming language in general) so let it be that while letting the world ponder over the differences between a library and a framework.

What is the difference between React Js and React Native?

React Js is for web/ browser apps while React Native is for mobile apps.

*Pre-requisite for this tutorial is a basic understanding of HTML, CSS and Javascript.*

The simplest way to get React installed is through Node js (a runtime/ environment\* that allows you to run javascript at the back-end/ server). Install Node js from here: <https://nodejs.org/en/download/>

\* Again you don’t need to stress yourself with what Node js actually is, consider it something similar to a back-end programming language (Javascript here) like PHP.

Select the LTS version and then your operating system.

Once Node js is installed now run:

npx create-react-app your-app-name

If you encounter an error saying “………………..as it does not contain a package,json file”, run:

npm i create-react-app *(both “i” and “install” can be used alternatively in node commands)*

*npm is a package manager of Node js which allows you to install various node packages create-react-app being one of such.*

*Now run:*

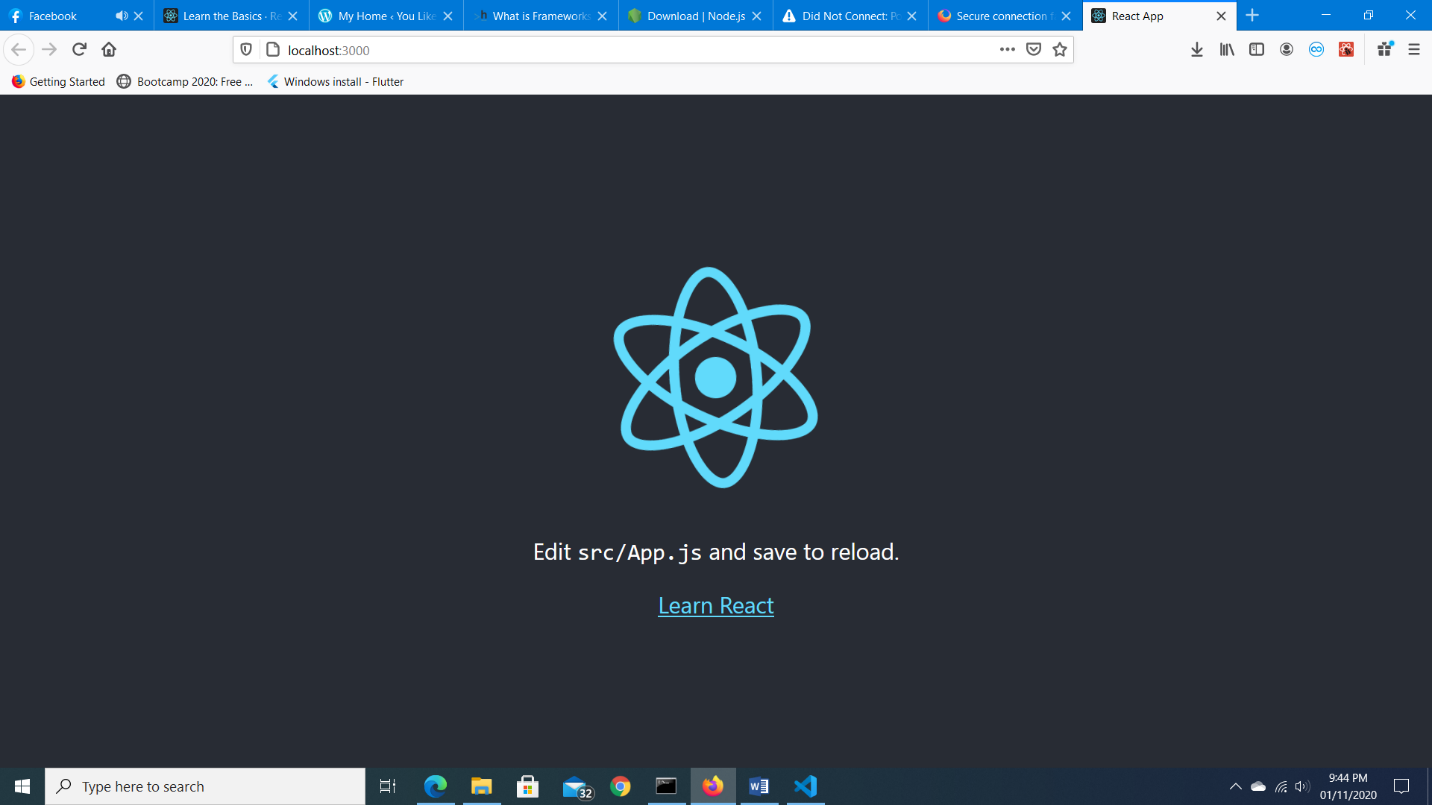
npx create-react-app your-app-name

and your first react app has already been created in the folder named “your-app-name” in this case. You can specify the app name of your choice.

Also run:

npm start

in terminal/ command prompt while in your app’s folder. This would run the development build of your basic react app in the browser at localhost:3000 which would look something like:



Now open that folder in a code editor of your choice.

A few files and folders of our concern for now would be:

The “public” folder containing the index.html

The “src” folder containing the App.js file along with some others.

**The hierarchy**

Let’s now connect some dots.

Open public\index.html in the code editor. Forget the meta and link tags for now and see the body where we have an empty <div> with the id “root”.

Leave that empty and now open: src\index.js which would be having this code:

ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);

*Yeah I know there would be something above this too. That code just imports the required modules like React, ReactDOM etc. to compile this code and connects the styling file index.css available in the src folder.*

Anyways so in the code above, a method (coming from ReactDOM which itself is a React module) we see is:

ReactDOM.render(

/\*blah blah blah\*/,

document.getElementById(‘root’)

);

This simply means that render my /\*blah blah blah\*/ stuff on the <div> with the id “root” in that public\index.html file. Try changing this “root” to something else and also change the div id in index.html to the same and the app would still render. (You don’t need to reload the page in your browser, it would reload automatically with the saved changes)

Now what is that:

<React.StrictMode>

    <App />

  </React.StrictMode>,

For the StrictMode part all you need to know now is that it is a React tool which highlights potential problems during development. What we are concerned with here is the <App />.

What is this <App/>? Looks similar to an HTML tag? Well that is because React uses an object oriented language called JSX which is an extension of Javascript. JSX allows us to use HTML tags and HTML like tags in our Javascript files. Again, we do not have to be concerned about the nitty gritty her.

<App/> is a component being exported from App.js (in the same src folder) and imported here in our index.js file through:

import App from './App';

**What are components in React?**

Components are what build a React UI. A React UI is a collection of components out of which some may be parents and some may be children of those parent components which all relate to a single grand parent, the <App /> here.

One component may have several components inside it.

React supports two types of components: class based and functional. We would be focusing on functional components in this tutorial.

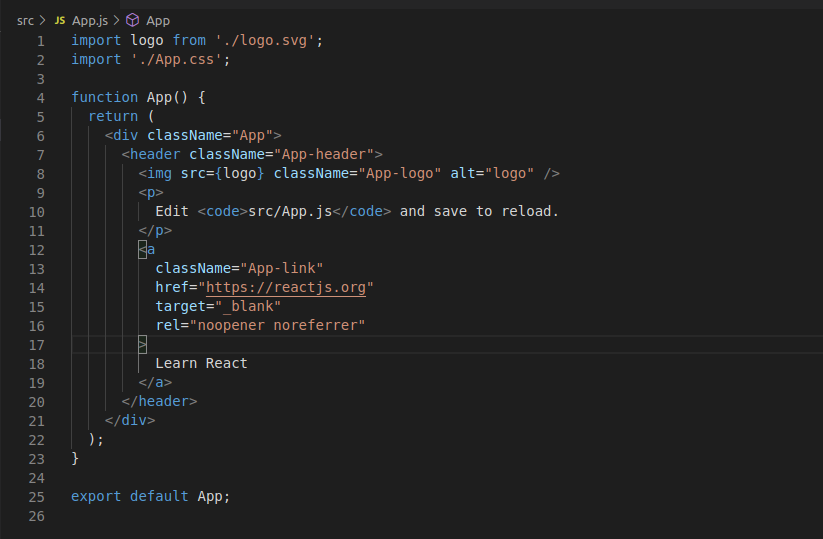
Now we can sum up the code we went through as:

“Render the App component from App.js in StrictMode on the div with the id “root” in our public\index.html file.”

Why all this mess though? Remember, React is Javascript/ JSX not HTML and browsers recognize only HTML for rendering/ displaying stuff. Js needs to connect with HTML somehow to display what it wants. So, it’s actually that index.html file being rendered with stuff coming from js files working in the background through that linkage we just saw.

**Let’s now see components in action**

Open App.js in the src folder and you’ll see some code similar to the one below:



You must have noticed that the return function starting at line 5 contains mainly HTML code with two small differences:

1. Some tags have attributes with the name “className”. This is the same as “class” in HTML, however, as we are actually writing js here rather than HTML, we use the “className” attribute recognized by Js, instead of “class”
2. Second difference you might have noticed is the way the “src” attribute in <img> tag refers to logo at line 8. Well that is because, whenever you have to write javascript within HTML you do not within curly braces like {logo}. In React you add images this way:

* import the image as in line 1 (any other name could have been given to the imported image instead of logo). ‘./logo.svg’ means the logo.svg file available in the same folder (this works just the way you have been giving paths in HTML files).

The next import on line 2 is of the App.css file available in the same “src” folder. You may use this file to style your HTML elements inside the return function the way you want. Just remember that instead of class, you have to use className, the rest is same, give them name, id or any other attribute you want and style them with the App.css file.

Now comes the function (line 4). A while ago I mentioned that we would be focusing on functional components, so here is one for you.

That <App /> we used in the index.js file was basically this function in App.js or in React terms a functional component.

This App function here simply returns some JSX which in simplest terms can be called “HTML or HTML like stuff put into Javascript”. Whatever you have to display/ render on the browser put that in HTML format (with the changes mentioned in (a) and (b) above) inside the return function’s parentheses.

Ok now if you remember we imported this App function in index.js file. For it to be imported there it had to be exported from here, hence the line:

export default App; (at 25)

The “default” keyword ensures that whenever this file is imported anywhere the App function would be exported from it. The function could have been imported with any name and used in index.js and because it was a default export, React would have known that it is the App function we are talking about.

The App function could also have been exported like:

export {App}; //However, this way it would not have been a default export and while importing it had to be specifically mentioned like:

import {App} from ‘./App’;

or we could also have written export default before function App() and could have excluded line 25.

That was the structure of a react app at its most basic level.