**React JS**

To start react project: npx create-react-app project-name

e.g: npx create-react-app hello-world

to start a react project: npm start

When react starts it finds a folder named src.

Everything that is statically displayed and publicly available is in the public folder. The index.html is the main file that is rendered when a react project starts. It a normal html file and all you need is a div with an id “root”. This is rendered in index.js. The main react component running by default is app.js.

So the flow is, we have a component which is App.js, whatever is returned from here goes to index.js. This index.js is getting DOM element root from index.html and is rendering it.

All react related code goes in src folder. If you write anything out of this folder, then it will not run.

Everything in react must be treated as a component. React component must have its first letter Capital, otherwise it will not be treated as a component. React component is basically a javascript function with

1. first letter capital. Every tag/component must be closed in react, either properly or as a self-closing tag.
2. It should always return a single html or jsx. Jsx is javascript html. Separate component files are saved as .jsx or .tsx if you are using typescript.
3. The jsx should be either on the same line as of return, otherwise () parenthesis should be used.
4. You can import one component into another.

To create a typescript react project:

npx create-react-app hello-ts --template typescript

**Bundlers in React**

When we use typescript in react, then the code is transpiled into javascript bundle. Create-react-app (CRA) is the default bundler, which is obsolete (it is no more supported) now. Instead of CRA, we can use other bundlers like Vite, Parcel etc. However, the base code remains the same and if you use CRA component elsewhere, then it will also work.

**React Hooks**

We can insert logic and interactivity in our application by using hooks.

**useState:**

If you need reaction on data, then you have to use state. useState first value is initial value and second value is a callback that sets the new value. Whenever a state changes react will re-render the component, while using useState.

import {useState} from react;

//within a component

let [counter,updateCounter] = useState(0);//0 is the initial value of counter, and updateCounter is the callback which can work upon counter.

**useEffect:**

When you need to do some work upon the update of a component during its lifecycle. It takes as input two arguments, a callback function and a dependency array. The callback is called upon mounting of a component if the dependency array is empty.

useEffect(()=>{

console.log("use effect called");

},[]);

If you add something in the dependency, then that particular useeffect is run once initially, and also whenever that dependency is update, for eg: