REACT

npx create-react-app my-app => command to create a react application.

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!

npm cache clean --force – To clean the cache of npm

Code to display Hello World!

import React from 'react'

import ReactDOM from 'react-dom'

function App() {

return(

<h1>Hello World!</h1>

)

}

ReactDOM.render(<App />,document.getElementById('root'))  
  
Inside a style Tag we need { { } } to add a style for that tag or so.

We can only return only element from a given component.

* Export default is used to pass a component so it can be used by other components.
* We can use return() in a functional component but an alternative is ()
* We can reuse a component by just passing a parameter {} and give the value as {}  
  Example: {title} and the value will be {title} while calling a component we need to call it as <Component title=”Text” />
* If we have () then function is expecting something to return and no braces then no return type.

**PROPS**: Values which are being passed from one component to the other.  
**STATE**: A central variable which store the data which can be used anywhere and it can be used only in class component.

* Never Update state variable directly. (Mutate)
* Any change in state react re-renders on its own and no need of page reload.

**Reactstrap**: Bootstrap is added in react and can be used here as Reactstrap.\

**React-icons**: All the embedded icons can be found here in react-icons.

**react-toastify**: Get Tosting messages.

<https://react-icons.github.io/react-icons/icons?name=fa>

* TypeError: Cannot read property 'createElement' of undefined  
  SOLUTION: import React,{useState} from 'react'; which was   
  import { React ,useState} from 'react';
* Custom based CSS need to be called last prior to any other CSS.
* Redux is used to maintain a centralized state variable.
* Fragment can be used as a div and can be used to pass multiple components.

**CONTEXT**:

* We need to create a context by using the syntax **React.createContext()**
* Then create a provider where in we declare all the variables and method which can be used everywhere. (context.provider)
* Then we just need to call the component inside the provider Component and then in that component inside return statement create a context.consumer component which can be used to access all the data which is being created in provider.
* SYNTAX for consumer:
  + <Context.Consumer>
  + {
  + context => (
  + <Fragment>
  + <h2>Name of the Mission: {context.data.name}</h2>
  + <h3>Code of the Mission: {context.data.code}</h3>
  + <button onClick={context.isAccepted}>Changing the Acceptence</button>
  + <h3>Acceptence: {context.data.accepted}</h3>
  + </Fragment>
  + )}
  + </Context.Consumer>
* Context has all the data which is being stored in the provider using which we can access those data.
* SYNTAX for provider:

<PackageContext.Provider value={{

data: mission, //Pass all the data

isAccepted: () => {

setmission({...mission, accepted: "Yes"})

}//...gets all the data declared for a state varibale

}}>

{props.children}

</PackageContext.Provider> value should have {{}} and all the data which is passed into this can be used to by the consumer.

* Context provides a way to pass data through the component tree without having to pass props down manually at every level.

***Reducers in context API***: It can be used to trigger the actions which are being written in actions js file