When using react we first import the required module there are two ways to import the modules-

1.

import React from “react”

import ReactDOM from “react-dom”

2.

var react= require(“react”);

var reactDOM= require(“react-dom”);

after importing we can use react as required.

For showing webpage in browser we use-

It only render one value at a time that’s why we use div to render multiple value in it. react.DOM.render(html code without “”, getElementById() or other targetor)

**Note:-** we can add js variable inside the html code for render method by using {} sign to cover our js variable or expressions but we can not use js statements inside them.

That’s why if we want to add a statement then we would have to create a new function and then pass that function the {}.

We have to change type as **text/JSX** in the script tag in index.html while linking the JS file when we use react.

Note:- contenteditable=”true” makes the text under tag editable in HTML.

Note:- We have to use JS properties as attributes for Html. When using any attribute in JSX we have to use camel cased version of the HTML tags if JS version is not available as for attributre ‘class’ is ‘className’ in JSX which is got from JS.

/ is used before > to close img tag in JSX.

Inline CSS implement-

We have to use style={{property 1: “value1”, property 2: “value2”}} //value should always be in strings and two properties are seaparated by ‘,’ .

This works similar to JS object as style accepts value as JS object. Another {} is used to give instruction to treat the JS object as JS code as in case of implementing JS inside the HTML section of .render() method.

We can target as objectname.property=’value’ if we created the object outside the render() method part. EX- style={objectname}.

Note:- CSS properties should be written as camel cased for example font-weight should be written as fontWeight in the case of Inline CSS implementation in JSX.

We can use jsx file to add custom html tag to our index.js file.

This can be done by creating a function with Functionname starting with capital letter in the jsx file which returns a part of html code using ‘return’.

At the top we have to import ‘react’ and at the end we have to use

export default Functionname;

Now we can use the <Functionname/> custom tag to add that part of code in our index.js file by importing that function as module in our index.js file –

import Functionname from ./Functionname

We can add multiple componenets JSX files like heading, list etc in a single App.jsx file for making the index.js file more simple.

For this we have to use (in every component JSX file)-

import React from “react”

at the start

// Componentfunctionname starts with

function Componentfunctionname(){ // capital letter in jsx file.

return (htmlcode)

}

//whereComponetfunctionname is the function which return a value

so that

// we can use it later using <Componetfunctionname/>.

export default Componentfunctionname;

at the end in every JSX file. when the function returns a html code as shown below-

Now we have to use App.jsx file which returns different compents tags to the main index.js file by only using <App/> tag inside the render() method in the index.js file. The format of App.jsx function part is like

import Componentfunction1name from ./Componentfunction1name

import Componentfunction2name from ./Componentfunction2name

// and so on

function(){

return (<Componentfunction1name/>

< Componentfunction2name />

}

export default App.js

Note:-

For any Component to work we have to use

import Componentfunctionname from ./Componentfunctionname

at the top where we want the < Componentfunctionname /> tag to work.

// where ./Componentfunctionname is the path of file .jsx extension is not necessary.

We can also import constant and function from other files in the index.js file by using

export {function1,constant1,etc} //in the component file and using

import {function1,constant1,etc} from ./componentfilename.js //in the index.js file

Note:- We can also import some of the exported functions, constants etc the ones we want for our purpose.

Note:- When using export default in the component file which can only export only one constant or function etc the name of the function or constant etc is not important while importing that function or constant or variable.

We can also use –

import kutta from “./filename.js

to import the exported default function, constant etc.

Note:- WE can also use wildcard ‘\*’ while importing to import all functions or constants etc to index.js all at once as an object with the name which we use while importing.

Ex-

import \* as kutta from ./filename.js

To use react environment on windows-

We have to use -

**npx create-react-app appname**

and then enter appname folder

and then run

**npm start**

**//** It will install required react modules and start the server with port 3000 to render our react code.

When we want to add custom attributes to our custom created tags. We have to use the format of tags as example given below

<Customtagname property1= “value1” property2= “value2”/> in the App.jsx. It creates an object in the respective function in the component jsx file with the name given by us. Usually named props.

Parameter of function becomes an object witch properties given by us in App.jsx file as <Customtagname property1= “value1” property2= “value2”>

These value can be used in the component tag as-

function Customtagname(props){

<h1>props.value1</h1>

<h1>props.value2</h2>

}

Note:- The property that we give to the cards in App.jsx function get returned to the respective component file like Cards or Header etc function and we can use them later according to our use.

Note:- We can also use props value from grandparent component to children component.

In the children component (In the Detail.jsx)--

function Detail(props) {

return <p className="info">{props.detailF}</p>;

}

In the father component(In the Card.jsx)--

<Detail detailF={props.tel} />

<Detail detailF={props.phone} />

<Detail detailF={props.email} />

In the grandfather component (In the App.jsx)-

<Card

name={contacts[0].name}

img={contacts[0].imgURL}

tel={contacts[0].phone}

email={contacts[0].email}

/>

Where contacts is an array of objects.

Map function-

It is used to process different object from an array of objects by processing the objects with a function.

To use map function first we have to create a new function in the App.jsx( or any but then we would have to link that file ) with which we want to process all objects like example given below by using an example of Header custom component (can be any custom element)-

function functionname (defaultvarforallobjectsfromarray){

return(

<Header key= “defaultvarforallobjectsfromarray.property1”

name= “defaultvarforallobjectsfromarray,property2” /> )

}

//Here Header is a custom component with the blueprint for the section which we want to apply to every objects from the array of objects.

// key should always be specified with using loop functions. It cannot be associated with props for that Custom element.

Now we have to use-

{arrayname.map(functionname without ‘()’)}

in the App return function so that it can be returned to the index.js to render correctly. this will process all objects from the array with the function given in parenthesis.

Above map function can be made shorter by using Fat Arrow function method as-

{arrayname.map( defaultvarforallobjectsfromarray =>

(<Header key= “defaultvarforallobjectsfromarray.property1”

name= “defaultvarforallobjectsfromarray,property2” />)

)};

**Using render to work with a JS statement in the html code:-**

When rendering a statement inside the {} in the Html code it cannot be done simply we would have to add a function() in the {} for this case.

**Ternary Operator:-**

But with the help of ternary operator we can achieve this much faster without the use of a function. It is used in given way-

conditiontocheck ? htmlcodetoruniftrue : htmlcodetoruniffalse

//This can be used in pure JS also not only limited to React.

Note:- null can be used in any case to not render anything.

**&& Opeator-**

&& operator can also be used to do the same thing as-

conditiontocheck && htmlcodetodisplay

//It is different from the usual && condition in which it checks both sides from true or false.

OnClick function:-

onclick function can be use with <button> tag by using **onClick={**functionname **}** or

**onClick={**JScode**}**

Note:- There is no parenthesis after functionname.

useState hook-

**useState(**initialvalueofstate**)** hook grabs the state of a variable and helps in performing a function process through it.

To do this we first have to assign the result of **useState(**initialvalueofstate**)** which is in the form of array-

const [var1,varfunct]= **useState(**initialvalueofstate**)**

where var1 captures the initial value given by us which is at index position 0.

And varfunct captures the function which is at index position 1 becomes a function.

Then we can later use the varfunct function and var1 variable inside another function to change the var1 with using onClick function in html <button> tag to run the new function as **useState** hook renders automatically when a change is detected in var1. As var1 changes the changed value renders at the webpage.

It can be achived through-

**function** newfunctionname **(){**

varfunct**(**newvalueofvar1**)**

**}**

**Note:-** varfunction() method contains new value of var1 in the parenthesis.

**Note:-** The change to the var1 will only be reflected when varfunct function performs an operation to it. As varfunct takes var1 value as parameter in its work.

Ex code for useState hook (used inside App.jsx file )-

import React, { useState } from "react";

function App() {

const [initial, changer] = useState(0);

function faltu() {

changer(initial - 1);

}

return (

<div className="container">

<h1>{initial}</h1>

<button onClick={faltu}>-</button>

</div>

);

}

export default App;

Note:- We can also use JS methods and functions like ternary operator in any part of our react code such as inline style section to change page design according to our requirement.

Note:-

To make react page stop automatically making get and post request when form is submitted and makes page reload-

event.preventDefault();

//event is the event associated with the form element and not input element.

where onSubmit={functiontorunwhensubmitclicked} attribute is used inside the form tag as an attribute or onClick={functiontorunwhensubmitclicked} is used as an attribute inside the button tag.

Note:- We can use anomymous function inside the useState() function part to get the hold of current state so that we can use it later. The parameter of the anonymous function inside the useState’s current state changer function holds the current state value.

EX-

function setC(event) {

const name = event.target.name;

const value = event.target.value;

setContact((previous) => { //”previous” parameter holds the

//current state value for later use.

if (name === "fName") {

return {

fName: value,

lName: previous.lName,

email: previous.email

};

} else if (name === "lName") {

return { fName: previous.fName, lName: value, email: previous.email };

} else if (name === "email") {

return { fName: previous.fName, lName: previous.lName, email: value };

} else {

return previous;

}

});

}

In this example we can get the current state details with previous word as parameter. In above example the setC() function is triggered by the input onChange attribute.

Spread Operator-

We can add elements of one array to another using spread operator.

Ex-

first=[a,b,c]

second=[a,b,c,…first] will includes all elemnts of first to the second array.

Note:- We can use …first at any indexposition in the first arrry from where we want to add the elements.

Spread Operator also works with object-

object1={

p1=val,

p2=val2}

object2={

…object1,

p3=val3,

p4=val4

}

This will add all properties value pair from object1 to object2 accordingly to the specified index position of placement.

Note:- If we are storing the “key” that we want to use when creating a new object in a variable then we can use the key which is the value of the variable and assign a value pair to it.

Ex-

var myVar= “mystring”;

Then we can use the “mystring” as our key for our object.

myobj={

[myVar]=val;

}

This will create an object with key:value pair as mystring:val.