**Introduction:-**

* Javascript library to create single page applications.
* In single page applications, server only sends the single html page(js bundle) and then react takes over the control for routing.
* Js bundle will intercept the request and will stop from going to a server and update the dom.
* Simple react snippets install in vs code as extension.
* To use emmet go to settings and add type:-javascript and value:-javascriptreact.

**Adding styles:-**

* The styling part will be applied to all components that is currently rendering into the browser as react append styles to head of web page.
* Inline styling- { { } } the outer box is for dynamic value and inner box is for javascript object.
* Jsx uses camelcase so need to be carfeul. (In html background-color but in jsx it is backgroundColor).

**Components and Templates:-**

* Components are building block of react app. They are rendered by react and seen in the browser.
* Components contain template and logic(coding).
* Function name starts with capital letter.
* Jsx allows to write html code in react component.
* className should be used in jsx as class is reserved keyword.Then it is changed to class while rendering.
* Type div.content and press tab.

**Dynamic values in Templates:-**

* We can write valid JS before return statement in the function or in the { } box.
* React will convert all data types to string before outputting it to the browser.
* We cant’t output boolean and objects for dynamic values.
* <a href={link}>Click Here</a> or <a href=” ”>Click Here</a>

**Multiple Components:-**

* App.js is the root component.
* To make component tree we should nest it inside the root component.
* Type sfc and press tab.
* Follow the practice of creating create arrow functions.
* <App/> or <App></App> both will work.

**Click Events:-**

* Pass the refernece of function that we want to invoke during event. (handleClick should be used instead of handleClick() ).
* If we want to pass parameter to function wrap it around anonymous function.
* When event occurs the event object is triggered automatically.(like event or e).It can be passed anywhere in the function like first or second or last argument.
* e.target will give the component that is targeted like(button)

**useState Hook:-**

* The variables are not reactive.So, the changes values are not rendered it the browser.So, we need to use hook.
* useState has two components – initial value and function to be called when the value changes.Due to function rerendering happens.
* useState is a Hook (function) that **allows you to have state variables in functional components**. You pass the initial state to this function and it returns a variable with the current state value (not necessarily the initial state) and another function to update this value.

**React Developer tool:-**

* Go to chrome web store and install it.
* Go to components to have look of component tree.

**Outputting Lists:-**

* To itererate over array of objects use map.
* e.g:-

{ // javascript bracket to put code

blogs.map((blog)=> //the anynomous function calling

(

        <div className="blog-preview" key={blog.id}>//to keep track it in DOM

        <h2>{blog.title}</h2>

         <p>Written by {blog.author}</p>

        </div>

      )// the anynomous function returning the jsx component

)

}

**Props:-**

* To have a reusable component props are used.
* Instead of using props.title directly we can destructure it.

**Reusing component:-**

* We can use same component but with different logic like using same component but filtering the content based on user’s preference.
* **E.g:-**

<BlogList blogs={blogs} title="All Blogs!"/>

<BlogList blogs={blogs.filter((blog)=>blog.author==="mario")}title="Mario's Blog"/>

**Functions as Props:-**

* We can define it it reusable component and use it according to user’s preference.

**useEffect Hook:-**

* This hook runs the functions for every render of component if don’t use [ ] .
* We can use it for fetching data and authentication service.
* We can access the state inside hook.

**useEffect Dependencies:-**

* To avoid rendering again and again.
* And do rendering for certain cases.
* Initial rendering is by default.
* Passing the empty array [ ] to useEffect insures that function in useEffect is called for only one time(initial rendering).After that it won’t call.
* We can define the dependency name in the array so that useEffect will only call the function inside it when only that value changes.

**Json Server:-**

* Allow us to create fake REST API using json file.
* While using json server each top level property is considered as resource and create endpoints for resource like edit,update ,delete.
* npx json-server --watch data/db.json --port 8000 run this command in different terminal.
* Resources:- <http://localhost:8000/blogs>

**Fetching data with useEffect:-**

useEffect(async()=>) This is not valid

useEffect(()=>

  {

//fetch is returning promise with a response object

//when the response object is return data is return again it a promise

    fetch('http://localhost:8000/blogs')

    .then(res=>{

      return res.json() // converts to a javascript object

    })

    .then((data)=>{

      setBlogs(data);

    })

},[]);

{ blogs &&

<BlogList blogs={blogs} title="All Blogs!" handleDelete={handleDelete}/>

}//it will take time to have data into blogs from API so use conditional statemenrs to check

**Conditional Loading Message:-**

* It is used to put condition regarding fetching the data.
* We can use settimeout so that we can see the conditional message.

**Catching errors:-**

* Use catch block to hande error.
* While using fetch,catch block gives error only regarding the connection.(but what if there are other errors like endpoint is not specified in the code or request is denied).
* If you want to define own error type throw Error(“ ….”)
* Use err.message to ouput the error.
* You can use it with the useEffect hook.
* res object is sent by web app and explore it.
* Careful about setting the state. e.g:- when to set pending and error to null or true.

**Creating a Custom Hook:-**

* Need to start with a word use or else it won’t work.
* Custom Hooks are **a mechanism to reuse stateful logic** (such as setting up a subscription and remembering the current value), but every time you use a custom Hook, all state and effects inside of it are fully isolated.

**The React Router:-**

* npm install react-router-dom@5 will install version 5.
* import {BrowserRouter as Router,Route,Switch} from 'react-router-dom';
* **Router is for top page.**
* Inside router there will **Switch it will act like switch statement** and **Route have property path which will act as case in switch statement.**

**Exact Match Routes:-**

* Exact path should be given else react stops there as soon as any route path string character matches.

**Router Links:-**

* To intercept the request and preventing making new request to the server, react uses link tag.
* If we quickly navigate the pages, then react will show error if custom hook is present in one of the navigate pages.

**useEffect Cleanup:-**

* **Cant’t perform state update on unounted component.**
* const abortCont=new AbortController();//to stop fetch
* fetch(url,{signal:AbortSignal})
* return ()=>abortCont.abort;//function cleanup(abort)
* AbortController is used to handle the error regarding the custom hook.
* When the abort happens we don’t want to update the state.

**Route parameters:-**

* Variable inside the route that is changing(dynamic).

e.g:-/blogs/123

/blogs/345

* We will use useParams hook to retrieve the id.
* Link is a jsx component while rendering in the browser it will change to <a>.
* If the dynamic value is going to change and it is a link then use string literal and $ sign

**Reusing Custom Hooks:-**

* Just call the hook and manipulate it according to the user preference.

**Controlled inputs:-**

* This are used for forms.
* Initially set value to the null or any other variable then using **onChange** function make a **asynchronous call and use e.target.value to set the new value.**

**Submitting Events:-**

* When a button is pressed inside a form it fires a submit event on form itself.
* Or attach click event to button itself and react it to that.
* e.preventDeafult() prevent the apge from refreshing.
* Json server automatically add a unique id.So, we don’t need to create id.

**Post Requests:-**

* The disabled attribute is a boolean attribute. When present, it specifies that the button should be disabled. **A disabled button is unusable and un-clickable**.
* **headers: { "Content-Type": "application/json" } -> we are sending the json object**
* **body: JSON.stringify(blog) -> save it as a string and convert it to the string from Json.**

**Programmatic Redirects:-**

* Like redirecting the user back to homepage when the form is submitted.
* useHistory hook like moving forward and backward with the help of arrows.
* The useHistory hook **allows us to access React Router's history object**. Through the history object, we can access and manipulate the current state of the browser history.
* Use history.go(integer).If positive integer move forward and if neagative integer move to the previous page that is visited.
* Histrory.push(‘/’) go to the that page.

**Delete requests:-**

 fetch('http://localhost:8000/blogs/'+blog.id,{

      method:"DELETE"

    }).then(()=>{

      history.push('/');

    })// making a request and deleting it and again moving to the home page

**4o4 pages:-**

 <Route path="\*">//it should come at Last.It matches any path that is not mentioned before that route that were mentioned.

    <NotFound/>

  </Route>

**How to run the project:-**

* **In the first terminal type npm run start.**
* **In another terminal type npx json-server --watch data/db.json --port 8000 to run json server.**