**Namaste React**

**Ep07 Finding the Path**

**Revise Ep06**

What is the use of useEffect hook?

It is a hook which react gives us which is called after the component is rendered.

It takes two parameters, callback function and a dependency array.

Call back function is called after the render. And if we have an empty dependency array then it will be called only once after the initial render, also if we don’t have dependency array it will be called only once after the render (same as empty dependency array).

If we have a state variable inside dependency array then callback function will be called after every variable state change.

|  |
| --- |
| **🔴Things to remember all the time very important🔴**   * Never create a component inside another component. Is because of rendering. * Never ever write useState hook inside if else. * Never write useState inside a for loop, as it will creates so many state variables and will make our application slow. * useState is a hook that react gives us to create local state variables inside functional component, so never use useState outside functional component. * We can create any number of useEffect hook in our code according to use cases. |

For routing we use the most popular librabry named as React Router and we need to install it using the npm install commant.

mpm i react-router-dom

after installing we will get react router in our package.json. we will be using the createBrowserRouter as this is the recommended react router though there are other options also.

First of all, we will import the react router from react as below,

Import {createBrowserRouter from “react-router-dom”}

Now to create the router we need to create as below

const appRouter = createBrowserRouter([

{

path: "/",

element: <AppLayout />,

},

{

path: "/about",

element: <About />,

},

]);

This configuration will decide what needs to be render on what url path.

Here path in createBrowserRouter we have array of paths in which the path is the url path after domain name, in our case it is localhost. element is the component that we want to render.

NOTE: Always create router below the AppLayout as code executes in sequential manner.

After creating the configuration of router, we need to provide this configuration to our app in order to run, and to do that there is a one component we need to use named as RouterProvider.

**How to use RouterProvider?**

Instead of rendering our app directly to the root we will give the responsibility to Router. So instead of app we will use RouterProvider which will take the router configuration as the props.

root.render(<RouteProvider router={appRouter} />);

so whatever the root will render it will be according to the configuration that we are passing as an prop to the RouteProvider.

**what if we give wrong url in browser for which we have not define the reouter behavior?**

* we can handle this by adding error page and giving that error page path to the createBrowserRouter. we have something known as errorElement, so we will give the error page path here like below.

const appLayout = createBrowserRouter([

{

path: "/",

element: <AppLayout />,

errorElement:<ErrorPage/>,

},

{

path: "/about",

element: <About />,

},

]);

* suppose we have error component which shows the error page on wrong url entered but user will not get the actual reason of the error and if we want to show more information about that error but how do we show that information?
* React router dom gives us a hook named as useRouteError and can be imported as below.

Import {useRouteError} from “react-router-dom”;

**what is useRouteError?**

* This is a hook which gives us an error object. this error object contains lots of information about what type of error do we have. we can use this error information however we want to use. the code will look something like below.

import {useRouteError} from `react-router-dom`

const ErrorPage = ()=>{

const err=useRouteError();

console.log(err);

return (

<div>

<h1>Oops!!!</h1>

<h2>Something went wrong </h2>

<h2>{err.status + ` : `+ err.statusText}</h2>

</div>

);

}

What are the types of routing?

1. **Server-side Routing**

Server side routing id the way that all are the pages comes from the server.

e.g. If we change the route or changes the url then the new page will come from server, if navigate to About Us page form the Home page then this About Us page will be directly coming from the Server.

1. **Client-side Routing**

Client side routing allows your app to update the URL from a link click without making another request for another document from the server. Instead, your app can immediately render some new UI and make data requests with fetch to update the page with new information.

**What if we want to click on the links or button on the navbar? e.g. Home, About, Contact US etc**

usually we are using the anchor tag for this but there are some problems with the anchor tag.

- If we click on any anchor tag it will refresh the whole page.

- as we are building a single page application this is bad practice to have refresh on every click of anchor tag.

instead of anchor tag we can use the Link, this Link is an component which named exported from the react-router-dom library.

How to use -> instead of anchor tag we will replace it with the Link tag and it will work fine.

**Conditional Routing**

What if we want to render the body component inside the Header and footer component when route is “/” and About Us component when the route is “about”? this is called as conditional routing.

To do this we need to make the Body and About Us component as the children of Root Component in our case AppLayout component. We will do as below.

const appRouter = createBrowserRouter([

{

path: "/",

element: <AppLayout />,

errorElement: <ErrorPage />,

children: [

{

path: "/about",

element: <About />,

},

],

},

But after doing this also will not render the conditional routed page and we need to use one more component named as Outlet.

**Outlet:** An <Outlet> should be used in parent route elements to render their child route elements. This allows nested UI to show up when child routes are rendered. If the parent route matched exactly, it will render a child index route or nothing if there is no index route.

Our appLayout will becomes as below,

const AppLayout = () => {

return (

<>

<HeaderComponent />

<Outlet />

<Footer />

</>

);

};

Here the <Outlet/> will get filled by the children configuration we did before.

**Dynamic Segment/Routing**

When we are at the home page of our food ordering App we can see number of restaurants and if we click on any of them then it should navigate us to that specific restaurant info page. This is something known as Dynamic segment.

**Dynamic Segments:** If a path segment starts with : then it becomes a "dynamic segment". When the route matches the URL, the dynamic segment will be parsed from the URL and provided as params to other router APIs.

const appRouter = createBrowserRouter([

{

path: "/",

element: <AppLayout />,

errorElement: <ErrorPage />,

children: [

.

.

.

{

path: "/restaurant/:id",

element: <RestaurantMenu />,

},

],

},

]);

Here to read the restaurant id from the url and pass it to the RestaurantMenu component we will use the useParams hook.

**useParams:** The useParams hook returns an object of key/value pairs of the dynamic params from the current URL that were matched by the <Route path>. Child routes inherit all params from their parent routes.

import { useParams } from "react-router-dom";

const RestaurantMenu = () => {

*// Get the Id param from the URL.*

const { id } = useParams();

return (

<div>

<h1>Restaurant Id: {id}</h1>

</div>

);

};