[Lecture - 7]

[Finding the Path]

Date.....

```
useEffect(() => {

}, [])
```

- useEffect will be called everytime our component renders.
- This dependency array changes the behaviour of it's render.
- Dependency array is not mandatory, only callback function is compulsory inside useEffect.
- When we call useEffect with a dependency array it will render everytime our component renders.

- What if there is an empty dependency array

```
useEffect(() => {

}, []);
```

→ Then useEffect is called on only initial render (just once)

→ Once when the component render for the 1st time.

→ It not be called again and again even if the component re-render

Spiral

Date.....

→ The basic nature or the default behaviour of useEffect is to be called after each render but if we give it a dependency array then it is just called after once.

→ And what if we put something inside the dependency array
Then it will only be called when the dependency changes.

→ if dependency array is [btnName] → called everytime [btnName] update.

→ few more something about useState

(1) Never use useState or create state variable out side the component.

useState has a specific purpose it is used to create local state variable inside the functional component. so always call it inside functional component.

(2) Try to call this hook on the top, ~~so that~~ (when the function starts) so that we don't have a lot of inconsistency in the code because JS is a synchronous single threaded

Spiral

Date.....

language, the code will run line by line.

(3) Never use useState hook or create state variable inside if-else, we can do that but don't do this, this can create inconsistency in our programme.

(4) Never create state variable inside for-loop & inside the function.

→ State variable are meant to be created inside the functional component on the higher / 1st level.

→ Routing →

- npm i react-router-dom -

- App.js →

- import { createBrowserRouter } from "react-router-dom"

- It will create a routing configuration for us, we will create a routing configuration inside appRouter,

→ Configuration means that some information

Page = Component

Date.....

that some information that will happen define what will happen on a specific route.

→ "createBrowserRouter" takes a list of paths (path is nothing but an Object, and Object will contain 2 things)

```
const appRouter = createBrowserRouter([
    {
        path: "/",
        element: <AppLayout />
    },
    {
        path: "/about",
        element: <About />
    }
])
```

→ But just creating the configuration won't work we will have to pass this configuration → appRouter to render it onto the page.

→ How do I pass/provide this?

— For that we will have one more important component that we can import from react-router-dom

— React-router-dom gives us this functionality by giving us a component known as RouterProvider

Scanned with CamScanner

→ shortcut in VS-code for creating boiler plate code for component → "rafce" and enter

Date.....

→ import { createBrowserRouter, RouterProvider } from "react-router-dom";

→ The RouterProvider will actually provide this routing configuration (appRouter) to our app.

→ Earlier we were rendering the AppLayout directly,

root.render (<AppLayout />)

Now instead of this we will provide our router configuration.

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

→ we imported RouterProvider component, this component is exported from "react-router-dom" library.

→ for exceptional router —

Page Show →

Unexpected Application Error!

404 Not Found

→ This error is not like the common ugly error we got in console window or screen. It is a dedicated error page given by react-router-dom.

Spiral

Date.....

→ We can create our own error page:

```
const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />
    errorElement: <Error /> → a separate component
    =
    =
    =
    =
  }
]);
```

Error.js →

```
const Error = () => {
  return (
    <h1> OOps!! </h1>
    <h2> Something went Wrong </h2>
  );
};

export default Error;
```

Date.....

→ One more cool feature that react-router-dom gives us in terms of error, it gives us an access to an important hook useRouteError

→ using this hook it gives us more information about the error ~~error~~ instead of writing Oops!!! Something went wrong!! we should also tell some more details, a better message on our page

→ we can read the message using {useRouteError} we can show a specific detail to the user.

Error.js → import {useRouteError} from "react-router-dom"

```
const Error = () => {
  const err = useRouteError();
  console.log(err);
  return (
    <div>
      <h1>Oops!!!</h1>
      <h2>Something went wrong</h2>
      <h4>{err.status} : {err.statusText}</h4>
    </div>
  );
};
export default Error;
```

Spiral

Date.....

→ Now we will solve the problem of Children routes.

→ If we go to any particular route like /about page, we lose our Header & footer.

→ Ideally the Header should always be there on the App.
And About us page should load below.
Header should be intact, the lower portion (body) will change.

→ {Outlet} from "react-router-dom";
This Outlet is a component, whenever there is a change in the path so this Outlet will be filled with the children acc. to the path.

outlet will be replaced by the component acc. to the path

App.js →

```
import React from "react";
import ReactDOM from "react-dom/client";
import Header from "./components/Header";
import Body from "./components/Body";
import About from "./components/About";
import Contact from "./components/Contact";
import Error from "./components/Error";
```

Spiral

Date.....

```
import { createBrowserRouter, RouterProvider,
  Outlet } from "react-router-dom";

const AppLayOut = () => {
return (
  <div className="app">
  <Header />
  <Outlet />
  </div>
);
};

const appRouter = createBrowserRouter([
  {path: "/",
   element: <AppLayout />
   children: [
   {
    path: "/"
    element: <Body />,
   },
```

Spiral

55:37

Date.....

```
{
  path : "/about",
  element : <About />,
},
{
  path : "/contact",
  element : <Contact />,
},
]);

errorElement : <Error />,
},
]);
```

- Now we will create links in Header for these pages.
- When we are using react and we want to route to some other page never use an anchor tag.

```
<a href="/about"> About us </a>
```

Spinal

It impact on performance

Date.....

- When we use anchor tag the whole page got refreshed and we don't want to refresh our whole page but still we want to move on to a new route.
- We can navigate without reloading the whole page.
- import {Link} from "react-router-dom";
- Link component works exactly the same as anchor tag.

The only difference betn Link & Anchor tag is we use "to" in place of "href"

```
<ul>
  <li> Home </li>
  <li>
    <Link to="/about"> About Us </Link>
  </li>
```

- It just refreshes the component.
- There is a difference betn reloading the page and refreshing the component.

That's Why our React Application is known as Single Page Application (SPA).

Spiral

Date.....

- This App component is a whole single page and all the routing, all the new pages are just the component interchanging themselves Everything is component in React.

- There are 2 types of routing that we can have in our ~~React~~ web application.
  (1) Client Side Routing
  (2) Server side Routing

- Server Side Routing means we have index.html, aboutus.html, Contactus.html and if we click on our anchor tag "about", about.html, it reloads the whole page it sends the network call to Aboutus.html fetches that html and renders it on to the web page, refreshes the whole page. This is Server Side ~~rendering~~ routing.

- Now over here this is client side routing we are not making any network call while we are moving towards the page because all the components are already loaded into our App.

When we load our App for the first time it already has the code for Aboutus

Spiral

Date.....

- The only network call is made when we make our network call in API.
- We are not fetching a new page that's why this is known as client side Routing.
- We are implementing Client Side Routing.

→ Now, we will see how we can do the Dynamic Routing.

- How we can create different routes for different Restaurant in the page?
- We will create a new component RestaurantMenu.js
- We will create one more children route in App.js.

App.js → const appRouter = createBrowserRouter([

{
path: "/",
element: <AppLayOut />,
children: [

{ path: "/"
element: <Body />,
},

Date.....

```
{ path : "/about"
  element : <About />
},

{ path : "/contact"
  element : <Contact />,
},

{ path : "/restaurants/:resId"
  element : <RestaurantMenu />,
},

],
errorElement : <Error />,
},
```

this part of Url is dynamic, resId can be change acc to the Id of the restaurant (every restaurant has its diff. Id)

→ When we have RestaurantMenu page, let's dynamically get the data from Swiggy API

RestaurantMenu.js → import {useEffect} from "react"

useState

```
Const RestaurantMenu = () => {
  const [resInfo, setResInfo] = useState(null)
  useEffect(() => {
```

Scanned with CamScanner

Date.....

fetchMenu();

}, []);

const fetchMenu = async () => {

const data = await fetch("..... .....

");

const json = await data.json();

setResInfo(json.data

};

~~return (~~

~~<div class Name = "menu" >~~

~~<h1>~~

~~if (resInfo == null) return <Shimmer />;~~

~~return (~~

~~ ~~

return resInfo ~~==~~ === null ? (

<Shimmer />

) :

(

const {name} = resInfo?.cards[0]?.card?.card?.info;

or

{name, cuisines, ~~[illegible]~~, costForTwoMessage}

Date.....

up↑

return (<h1> {resInfo?.cards[0]?.card?.card?.info?.name} </h1>

or

<h1> {name} </h1>

<h3> {cuisines.join(", ")} </h3>

<h3> {costForTwoMessage} </h3>

But this thrown error because initially resInfo → useState was null, so we don't have any data in resInfo and we are trying to find name inside it, so we have to move condition on top

if (resInfo === null) return <Shimmer />

const {name, ayers . . . . . .

return (

we should not use ternary operator blindly

Before return keyword →

Date.....

```
const {itemCards} =
resInfo?.cards[2]?.groupedCard?.cardGroupMap
?.REGULAR?.cards[1]?.card?.card;

<ul>
{itemCards.map((item) => <li key={item.card.info.name}></li>)}
</ul>
```

```
<ul>
{itemCard.map((items) => (
<li key={item.card.info.id}>
{item.card.info.name} - {item.card.info.price || item.card.defaultPrice}
</li>
))}
</ul>
```

→ we had :resId, how will we read this → :resId in our component?

→ we have a superpower which is given by react-router-dom.

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

Spiral

• Params is an Object with the resId, that's why we can just destructure it on the fly.

Date.....

→ const RestaurantMenu = () => {

const [resInfo, setResInfo] = useState(null);

const {resId} = useParams();

→ ideally we should keep API in constants.js →

export const Menu_API = " ------ ";

→ In RestaurantMenu.js →

useEffect(() => {
fetchMenu();
}, []);

const fetchMenu = async () => {

const data = await fetch(Menu_API + resId);

const json = await data.json;

setResInfo(json.data)

};

Date.....

→ Body.js – import {Link} from "react-router-dom"

→ we will link the restaurant card so that they could become clickable.

```
<Link to={"/restaurants/"+restaurant.data.id}
    key={restaurant.data.id}
    ↓
    Key should be here
<RestaurantCard resCard={restaurant} />
</Link>
))}
```

- <Link> is a component which is given to us by react-router-dom which is a special component and behind the scenes Link is using anchor tag.
- Link is wrapper over anchor tag ~~it's a~~ when we make anything a link that means now react-router-dom will keep a track that this Home is a link that we don't need to refresh the page. It's a special type of a link Behind the scene HTML doesn't understand Link. Browser understand anchor tag

Spatial