24/02/24

Episode-07 Finding the path.

→ we will learn about Routing and we will dive deep into HOOKS.

☆ useEffect → Syntax `useEffect( () => { }, [ ]);`

↑ call back func. ↑ dependency array

In dependency array you can write any state variable or even func. If there will be any change in that state variable or func then the call back func. of useEffect HOOK will be called.

Remember: If you are writing anything in dependency array then useEffect will be called twice:

- → once after initial rendering. ☆ (of that component)
- → once every if there is any change in state varible or func. inside dependency array. ☆

Note: dependency array is optional, call back func. is mandatory

↳ If ~~there is not any dependency or~~ dependency array is empty then call back func. of useEffect will be called after every rendering of that component.

```
import — — —

const component = () => {

→ console.log("I am rendered");

→ useEffect (() => { [console.log("useEffect called")}, [ ]);
→ console.log ("I am written after useEffect");
  return (

  );
};
```

output

```
I am rendered
I am written after rendering
useEffect called ← { it is called after comp-
                     -leting the rendering }
```

case 1 :- You have not given any dependency arry in useEffect

```
useEffect(() => { console.log("Hie") });
```

In this case useEffect will be called on every rendering.

case 2 :- You have given an empty dependency array.

```
useEffect(() => {console.log("Hie")}, []);
```

In this case useEffect will be called only once after the initial rendering.

case 3 :- You have give any State variable or func. in dependency array.

```
useEffect(() => {console.log("Hie")}, [stateVariable]);
```

In this case useState will be called on 2 occasions :-

- once after the initial rendering
- Every time when State variable changes.

* Whenever my state variable changes react re-renders that component.
* Hooks (useState(), useEffect()) can only be called inside your component ⟶ Remember this.

⟶ useState is used to create local State variables inside your compo

⟶ Always write State variables on top inside the component

⟶ Never use your useState Hook inside any condition (if/else)
↳ it will create in-consistency in your program.

→ If you are using useState() inside any condition then for ~~some~~ it may not get created or called on initial rendering and after any re-rendering, the condition might get's changed & that useState Hook will be called. This is what inconsistency means.

→ Never call useState inside any for loop / function.

*** Always keep useState on top of your component.

→ A ~~framework~~ library is used to create ~~the~~ a specific task / functionality

→ A framework on the other hand provides everything like the basic structure and architecture of the application, required to develop a complete application.

→ For example react-router-dom → it is a ~~same~~ JS library which is specifically used for routing.

→ Root level component: The component in which we are assembling all other component and which is getting rendered into our root div in html.

→ How to use react-router-dom.

Step 1:- Install using npm i react-router-dom.

Step 2:- Import createBrowserRouter as named import from react-router-dom

When ever we need to do routing we need to create a routing configuration

Step 3:- We will create a routing configuration inside our app compo using createBrowserRouter

```
const appRouter = createBrowserRouter();
```

→ configuration means, what will happen on a specific route. The information.

→ means if I am calling about, what should happen?

→ createBrowserRouter takes a list (of objects) as an input

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

If my path is "/about" then load "About" element/component

Step 4:- Also install RouterProvider component as named import from react-router-dom.

→ Now we need to provide our configuration to our ~~root level components~~ app. ↗ ~~AppLayout~~

Instead of rendering our to applayout component directly.

```
root.render(<AppLayout />);
```

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

↓

It will basically provide our appRouter configuration to our app

→ We are basically providing our appRouter configuration to our app.

→ Now if I goto /about it will load About component.

→ In react-router-dom we have different routers, but we are using createBrowserRouter because it is recommended by react-router-dom library itself for all react projects

→ If you will write any random stuff on your app url for example :

localhost:1234/HiebroW

Then it will provide you an error page 404 Not found, which is created by react-router-dom. It will not throw any error.

→ We can also create our own error page.

→ In our AppRouter which is basically our router configuration we can also write errorElement. It will be called during any error

```
{
  path: "/",
  element: <AppLayout />,
  errorElement: <Error />,
},
```

→ Error component which we have created manually.

→ How to identify any component: It's 1st letter will always be capital.

→ How to identify any HOOK: Every HOOK includes "use" keyword.

→ React-router-dom also provides us a hook named as "useRouteError" It basically provides us more information about error.

→ How to use useRouteError.

1) Import it as named import from react-router-dom
2) const error = ~~useReact~~ useRouteError → It will give us an object which will have all informations related to error.

↳ 2) Then we can destructure it to get information like :-

{error.status}, {error.statusText}

Because it is a JS variable inside an object.

* How can we create children route

↳ If i am going to about page, keep my header & footer intact, just in place of body render about component.

To use this type of functionality we need to create children route.

→ Step 1 :- import `Outlet component` as a named component from react-router-dom.

→ Step 2 :- Inside your router configuration do below stuff :

```
const AppRouter = createBrowserRouter([
    {
        path: "/",
        element: <AppLayout />,
        children: [
            {path: "/",
             element: <Body />,
            },
            {path: "/about",
             element: <About />,
            },
        ],
        errorElement: <Error />,
    }
]);
```

children of AppLayout

→ and in AppLayout component do following :-

```
const AppLayout = () => {
    return (
        <div className="app">
            <Header />
            <Outline />
        </div>
    );
};
```

→ In AppLayout component my header will remain intact.

→ This outline component will be replaced by the children routes of AppLayout.

→ If path is "/" inplace of <Outline /> , <Body /> will come.

→ If path is "/about" inplace of <Outline />, <About /> will come.

→ So basically header will remain intact and other components will be replaced as per path & element mentioned in the children of AppLayout components.

→ But still now we can not route to these routes (About or contact page) by just click on it on header 😞

① You can do it using anchor tag. <a href="/about">
But you should not use anchor tag, because the whole page will be re-loaded.

② We can navigate to new page without reloading our page.
↳ For this we will use Link component.

✱ How to use Link component.

① Import it as a named import from react-router-dom.

② Link works same as anchor tag.

<Link to="path"> ——— </Link>.

→ With Link component our page will not get re-freshed.

→ That's why we call react apps as Single page application

→ We only have one page (AppLayout) and we are just change our components.

✱ Types of Routing :-

2 types of routing to go to new page

You can have into your web app.

1.) Client Side Routing : Where we doesn't make any network calls

2.) Server Side Routing : Suppose we have different pages like home.html, help.html etc & when I am clicking on help in nav bar then we are using an anchor tag to make a network call to help.html & we are fetching data from help.html. Basically our whole page gets reloaded.

→ So when we are using react-router-dom & Link we are doing client Side routing.

✱ Dynamic Routing

→ Why we use to store the api response.json() into a state variable ?

↳ Because after receiving the data from API we want to show it into our UI.

If we will use state variable then after receiving the data the component will be re-rendered and updated data received from API will be

shown on screen.

→ If we would have used normal variable then after receiving the data, the UI will not change, because the component will not get rendered again.

→ When you are using Link component, inted of anchor tag & goto your inspect and into your elements then in place of link you will see anchor tag.

→ Because internally link is using anchor tag.

→ How link is different than achor tag?

↳ Link is a wrapper on anchor tag. React-router-dom will keep a track that ~~link~~ is ~~an~~ ~~anchor~~ ~~tag~~; whenever inside Link tag is a link if → /abc or /help user should be navigated to that link without getting page refreshed.

→ Link is not understand by html or browser, it is given to us by our react-router-dom. HTML is browser only understands anchor tag.

→ If you are using a .map() and there are ~~min~~ nested components then you have to give the key to top level component or top level JSX, if you are using nested JSX

```
.map((items) => {
return(< Link key={ } > <Restraunts /> </Link>)';
});
```

↑ Here

↓ Not here.

26-02-24

✱ dynamic Routing.

→ Dynamic routing means using dynamic links.

For example: localhost:123/xyz → It can be randomly anything.

→ It can be used in different Scenario

1) When we want to create a unique room for our user with their roomid

localhost:123/roomid-123 → dynamic

2) When we want to create a product description page. And it will differ as per the Product & its details.

Basically dynamic programming is used in config driven UI.

→ How to use it:-

Step 1:- Add a new path into your routing configuration

```
const AppRout = createBrowserRoute([
  {
    path: "/restaurant/:restId",
    element: <FoodMenu/>
  },
]);
```

:restId → It is basically used to specify that this part of the path/link is dynamic.

restId is the name of the variable which presents dynamic routing.

Step 2: Add this restId into API call/link where you want to redirect. using `useParams();`

new HOOK.

→ useParams() is a hook which gives us the value of restId / or our variable dynamic value, which we haved used in path.

Step 3: Redirect to that link using LINK → component.

```
details.map((item) => {return (<LINK to="/restaurants/id" >
item </LINK>)});
```

This will be the value of restId

→ React event Handler

onError = {}; This is an envent handler in React is JS which is used to handle errors during the loading of external resources, Such as images, when error occurs, the onError event is triggere and the associated Javascript function is executed

for ex:- `<img src= details?.imgcode onError = {handleerror}/>`

OR

`<img src= details?.imgcode onError ={()=>{handleerror()}}`

Que: What will be the output of console.log(useState());

Ans In output we will get an array.

[undefined, function]

Represents the initial State is undefined.

Name of this func. will be "bound dispatchSetState"

Why we directly cannot use if else in JSX

Certainly! In React, JSX gets converted to JavaScript during the build process. JSX is a syntactic sugar over JavaScript, and JSX relies on expressions rather than statements.

An `if` statement is a JavaScript statement, not an expression. It doesn't produce a value that can be directly used in JSX. JSX expects expressions that can be evaluated to a value.

On the other hand, the ternary operator (`condition ? trueExpression : falseExpression`) is an expression. It returns a value based on the condition, and this value can be directly used in JSX.

So, in JSX, we prefer to use expressions like the ternary operator for conditions because they produce a result that can be rendered. This helps keep JSX simple, readable, and in line with the expression-oriented nature of the language.

Or if you want to use JS statement use them inside a function

```
{
          ()=>{
             if(isSearchNotFound === false)
            {
              <RestList topRestaurantsList={topRestaurants} />
            }
            else
            {
              <NotFound />
            }
          }
}
```

Read more about dependency array in useState();

Learn about fetch() in this documentation and this also.

Commented [Ma1]: