Finding The Path

Use Effect

* If No dependency array ⇒ useEffect is called on Every Render.
* If dependency array is empty = [ ] ⇒ use Effect is called on initial render (just once)
* If dependency array is [btnNameReact] ⇒ called everytime btnNameReact is updated.

e.g. ① useEffect (() ⇒ { fetchData(); });

{ only callback fⁿ is mandatory in Use Effect. }

② useEffect (() ⇒ {

```
    fetchData();
}, [ ]);
```

③ useEffect (() ⇒ {

```
    fetchData();
}, [btnNameReact]);
```

Use State:

* Never create your use state Component outside your ~~both~~ function Component
* Always try to call useState on top of functional Component.
* Never Create use State inside If-Else, for-loop, function.
* State Variables are meant to be created inside the functional Component on the higher level and no where else

rafce ⇒ to create boiler code for functional component.

Route Pages

* Use library react-router-dom.

* Install Command: npm i react-router-dom

$\boxed{\text{Whenever function start from use, It is hook}}$

~~Here~~. To create Route Pages first We ~~create~~ import Browser Router.

* It stores the current location in the browser's address bar using clean URLs and navigates using the browser's built-in-history stack.

* There are other routers are also present but prefer ~~Create~~ Browser Router.

```
Const. appRouter = createBrowserRouter([
        {
          path: "/",
          element: <Applayout />,
        },
        {
          path: "/about",
          element: <About />
              :
        ]);
```

Now we render appRouter using Router Provider.

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

this also need to import

import {createBrowserRouter, RouterProvider} from "....."

We can also handle Error using hook Named as {Use RouteError}

Use RouteError

The useRouteError hook returns an object with following properties:

* errors: The current error object, or null if there is no error.
* isError: A boolean indicating whether or not there is an Error
* message: The error message, or undefined if there is no error.

We need to import it from react-router-dom.

How to Create Children Routes

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

How to keep Header Intact:

Todo this we can use "react - router-dom" to import 'Outlet

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

It allow us to render child content inside of our parent component
This is perfect when we have a parent container, such as our
whole Application, when we have elements we want to
have present at all time, such as header and a footer.

e.g.

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

Note:
When you are using React and you want to Route some
other page never use a anchor tag <a></a>.
It is because if we click on link our whole page will be
Refreshed.

In React, we can navigate to router pages w/o reloading
whole page.

To do this, we can use Link component.

⇒ import {Link} from "react-router-dom";

This Link component works exactly same as anchor tag

```
<li>
  <Link to="/contact"> Contact Us </Link>
<li>
```

Link is superpowerful Component which react provide us it just refreshed the component, not reloads the whole page, that is why our React application known as single page application.

↓ it is just a One page only components are getting interchanged.

There are two types of Routing in Web Apps.

① Client Side Routing - Single Page Appln

② Server Side Routing. - Multipage Appln

For more learn from Notes of Akshay Saini Episode 5.

Dynamic Routing

dynamic path.

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

→ : shows after that it is dynamic

If You find Difficulty in writing Map function, then first write the values ~~m~~ manually.

e.g.

```
<ul>
    <li>{itemCards[0]. card? info?. name} </li>
    <li> { item Cards [1]. card? info?. name }</li>
    <li> { itemCards [2]. card ?. info?. name } </li>
</ul>
```

Now map function will by easy

हमें ItemCards पर map लगाना है और हर ItemCards के लिए हम car?. info?. name ~~के~~ चाहिए है

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

Use Param

The use Params hook returns an Object of key/value pairs of the dynamic params from the current URL that were matched by the <Route path>. child route inherit all param from their parent routes.

" Key should be On Parent JSX" while using MAP.

"Key should be On Parent JSX" while using MAP.