ROUTING

It is about url. A well designed website have best and simple url. Router is used to handle routes in your web app

INSTALLATION:

After creating react app, use this command

npm install react-router@next react-router-dom@next

or

npm install history react-router-dom@next

I have used the second one

IMPORTING ROUTER IN INDEX.Js

//importing react router

import {BrowserRouter as Router} from 'react-router-dom';

After importing, we wrap our app under it. It works like a context api, all tags under it have access to it

ReactDOM.render(

  <React.StrictMode>

    <Router>

      <App />

    </Router>

  </React.StrictMode>,

  document.getElementById('root')

);

Earlier before react router, we call all our app contents like this in App.js

function App() {

  return (

    <div className="App">

      <h1>Welcome to React Router</h1>

      <Home />

      <About />

    </div>

  );

}

export default App;

We used tags. But by that, we cannot show particular part of our app like, about us page, contact us page etc. For that we are using react router. Now we use main Routes tag and under that, we pass different route of our app using route tag. In <Route> tag, first one is the path, like “/” for home and “contact” for contact us etc and second one is the element to which we pass our components tag.

//importing router components

import { Routes, Route } from 'react-router';

function App() {

  return (

    <div className="App">

      <h1>Welcome to React Router</h1>

      <Routes>

        <Route path="/" element={<Home />}></Route>

        <Route path="about" element={<About />}></Route>

      </Routes>

    </div>

  );

}

Here we have called <Home /> and <About /> tag in Route and Routes.

Now we have create different routes, now we have to place these contact us home or about us(Generally routes) links somewhere in our app so people have access to it and by clicking on them, they can go to different route of our app. Now we are providing link in main app.js under h1 tag

import { Link } from 'react-router-dom';

function App() {

  return (

    <div className="App">

      <h1>Welcome to React Router</h1>

      <div>

        <Link to="/">Home</Link> {' '}

        <Link to="about">About</Link>

      </div>

      <Routes>

        <Route path="/" element={<Home />}></Route>

        <Route path="about" element={<About />}></Route>

      </Routes>

    </div>

  );

}

Now if some one write some wrong url like mainurl.com/hello, as hello route does not exist, so in that case we have to create a route to handle these uncertain circumstances, so it can show than Page not Found or path not found etc. Let see how is it done.

import { NotFound } from './NotFound'

import React from 'react'

export const NotFound = () => {

    return (

        <div>

            Path Not Found

        </div>

    )

}

<Route path="\*" element={<NotFound />}></Route>

To handle this, I have created some NotFound function in NotFound.js and return “Path Not Found” in it

Then I create another Route tag and pass path “\*” to it to handle all uncertain type of paths or url and pass <NotFound /> in its element.

PARAMETET(:):

Now we know how to create paths for different routes or pages. Now let say we have some online ecommerce shoe store. So from home to products page, we know how to go their, we again create some route and pass product page path and elements and then create link to display it so when someone click on it, it can redirect to product page. Now what if we have to redirect to multiple components in the same page, means we don’t want to go to some other page, instead when we click on some product item, it displays all its information but in that same page. For that react-dom provided a parameter. By that we can have multiple child urls or products url in the same parent or main product page.

import React from 'react';

import { useParams } from 'react-router';

function Product() {

    const {productId} = useParams();

  return (

    <div>

        Welcome to Products page. you are viewing this {productId} product.

    </div>

  );

}

export default Product;

Now this is the structure of product.js. All same, the difference is that to handle different products or child components or different product id in the same product page, we use useParams() hook. It handle all of the components in the child page and when we click or any item or component within the product page, our url will be changes like mainurl.com/products/abc or any product name.

So we maintain the same products page and within the same page, we can see different products.

<Route path="products/:productId" element={<Product />}></Route>

Now as mentioned in above example url which was mainurl.com/products/abc or any product name. so this can be done by this, as we pass productId const to useParams so we pass it here also but with parameter : operator to handle different child components or product in the same page. We can change any name , but remember that useParams const name and route parameter name should be same which is productId here.

Now by doing that, the problem is when we click on some product in products page, it shows only the information of that product, not the content of the products page.

We need some thing, that whenever we click on some product or child component in the products page, the products page component remain exactly same, withadditon to the details of that child component or the selected product.

NESTED ROUTING:

We need some thing, that whenever we click on some product or child component in the products page, the products page component remain exactly same, withadditon to the details of that child component or the selected product.

For that we need to do nested routing.

We created products page or parent page. Now for product detail, we will going to create another component with name ProductDetails.js and now do all the stuff like fetching data etc in that, and then

use const {productId} = useParams()

in that so all child component or products comes into product id. Now instead of this,

<Route path="products/:productId" element={<Product />}></Route>

What we gonna do is do nested routing.

<Route path="products" element={<Product />}>

          <Route path=":productId" element={<ProductDetails />}></Route>

</Route>

So each child component or products comes under the main product route by doing that and different product can be selected from <ProductDetails/> and url changes to

Mainurl.com/products/productId

But again the main problem is how we can see product main component and the child or products details in the same product page. For that we are going to use

<Outlet>

What does it do it will show any chosen product or generally productId which we are getting from <ProductDetails/> wherever we use <Outlet>

So in this case we are using in it our main product page. So by that we can have our main product page component and also some chosen productDetails in the same product page

Updated Product.js

import React from 'react';

import { Outlet } from 'react-router';

function Product() {

  return (

    <div>

        <h1>Welcome to Products page</h1>

        <hr/>

        <div>

            <Outlet/>

        </div>

    </div>

  );

}

export default Product;

We have removed useParams from here as all product dealing is happening in productDetails.js

INTIAL PRODUCT CONTENT AND LATER WANT TO REMOVE SOME:

So what if initially I want to display content on my main product page and later want to remove some of the contents when user click on child component or products. For that:

        <Route path="products" element={<Product />}>

          <Route path="/" element={<ProductHome />}></Route>

          <Route path=":productId" element={<ProductDetails />}></Route>

        </Route>

We have added another route in the main product route which will show by default when you click on products page. But this contain the content that you want to replace later, so when you click on any products or child component in the product page then ProductDetails route will run and this route will stop and its content will be vanish.

useNavigate:

it is a hook which is used to navigate into different part of an app. It works like a link, but they are dyamic.

import { Routes, Route, useNavigate } from 'react-router';

  const navigate = useNavigate();

<button onClick={ () => {navigate('/about')}}>Dynamic Navigation</button>

By using button, I am navigating to about page