React Router

1. What is React Router ?

It is ais a fully-featured client and server-side routing library for React.

Helps create and navigate between different URLs that make up your web application.

Provides unique URLs for different components in the app and makes the Ul easily shareable with other users.

- Course Content:

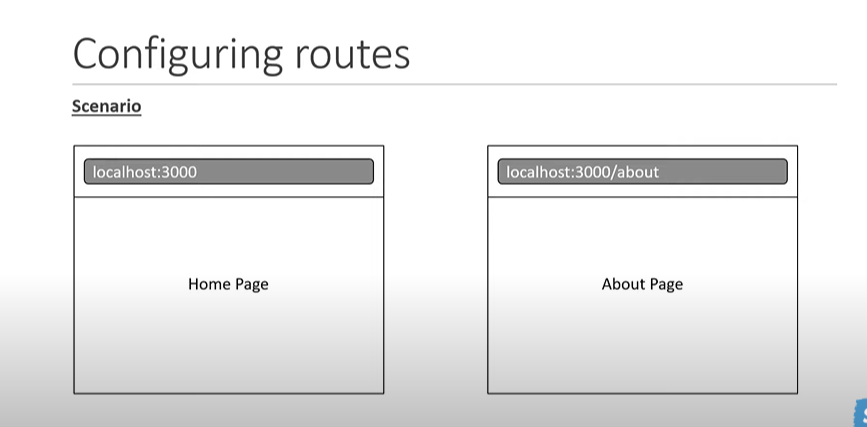
* Configuring routes
* Navigating on button click
* Navigating programmatically
* Dynamic routes
* Nasted routes
* Route parameters
* Lazy loading
* Authentication
* And a lot mor

- Installation

npm i react-router-dom@6 🡺pour la 6eme version

1. React Router ?
2. Configuring Routes

In this example we will learn how to configure routes with React Router for our scenario we’re going to set up two routes the first route is the home route if the user navigate to localhost port 3000 they should be to see the home page and if the visit localhost 3000/about they should be to see the about page.



Index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

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

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <BrowserRouter>

    <App />

    </BrowserRouter>

  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

BrowserRoter Allows us to us all the features from React-Router within the app component tree

App.js

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

import {About} from './components/About'

import {Home} from './components/Home'

function App() {

  return (

    <Routes>

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

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

    </Routes>

  );

}

export default App;

In App.js we will configurate the Routes using Routes and Route

Home.js

import React from 'react'

export const Home = () => {

  return (

    <div>Home</div>

  )

}

About.js

import React from 'react'

export const Home = () => {

  return (

    <div>Home</div>

  )

}

1. Links

In this example we will learn how to navigate to different routes using an element in the UI.



Navbar.js

import React from 'react'

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

export const Navbar = () => {

  return (

    <nav>

        <Link to='/'>Home</Link>

        <Link to='/about'>about</Link>

    </nav>

  )

}

App.js

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

import {About} from './components/About'

import {Home} from './components/Home'

import { Navbar } from './components/Navbar';

function App() {

  return (

    <>

    <Navbar/>

    <Routes>

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

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

    </Routes>

    </>

  );

}

export default App;

1. Active Links using NavLink

The NavLink component is specifically meant for building components like a navbar or breadcrumbs Or a set of tabs where you would like to highlight the current selected item provide useful context with screen readers if you want to navigate to routes from other parts of the app link should be the component to use as you wouldn’t want the active class being applied when it is not needed .

NavLink help us to see the active Link.

Change the css in the css file

Index.css

nav a.active{

  text-decoration: none;

  font-weight: bold;

  color: green;

}

Change the css in the js file

Navbar.js

import React from 'react'

import {NavLink } from 'react-router-dom'

export const Navbar = () => {

    const navLinkStyle=({isActive})=>{

        return{

            fontWeight:isActive ? 'bold' : 'normal',

            textDecoration:isActive ? 'none' :'underline'

        }

    }

  return (

    <nav>

        <NavLink style={navLinkStyle} to='/'>Home</NavLink>

        <NavLink style={navLinkStyle} to='/about'>about</NavLink>

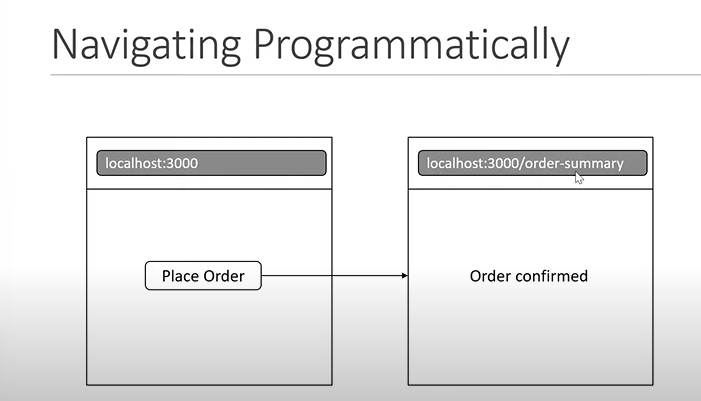
    </nav>

  )

}

1. Navigating Programmatically using useNavigate

In this Exampl we will implement a button click handler where we programmatically navigate to a different route in our application



App.js

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

import {About} from './components/About'

import {Home} from './components/Home'

import { Navbar } from './components/Navbar';

import { OrderSummary } from './components/OrderSummary';

function App() {

  return (

    <>

    <Navbar/>

    <Routes>

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

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

            <Route path='order-summary' element={<OrderSummary/>}></Route>

    </Routes>

    </>

  );

}

export default App;

Home.js

import React from 'react'

import {useNavigate} from 'react-router-dom'

export const Home = () => {

    const navigate=useNavigate()

  return (

    <>

    <div>Home</div>

    <button onClick={()=>navigate('order-summary')}>Place order</button>

    </>

  )

}

Home.js with Replace

import React from 'react'

import {useNavigate} from 'react-router-dom'

export const Home = () => {

    const navigate=useNavigate()

  return (

    <>

    <div>Home</div>

    <button onClick={()=>navigate('order-summary',{ replace:true})}>Place order</button>

    </>

  )

}

Replace katmn3o yrj3e loure b douk l asshome li f navigateur :

Chr7e bzyane hada:

In React Router, the **useNavigate** hook is used to programmatically navigate to a different route within the app. It is a more recent addition to the React Router library and is recommended for use in functional components.

The **replace** method is used to update the current entry in the browser's history stack, so that when the user clicks the back button, they will not return to the previous route. Instead, they will be taken back to the route that existed before the current one.

Using **replace** in conjunction with **useNavigate** allows for more fine-grained control over the behavior of the back button, and can be useful in situations where you want to prevent the user from returning to a certain route, or where you need to update the current history entry with new information.

In summary, useNavigate is the newer version of the **navigate** function and it's recommended to use it instead of **useHistory.push** and replace method is used to update the current entry in the browser's history stack.

OrderSummary.js

import React from 'react'

import {useNavigate} from 'react-router-dom'

export const OrderSummary = () => {

    const navigate=useNavigate()

  return (

    <>

    <div>OrderSummary confirmes.</div>

    <button onClick={()=>navigate(-1)}>Go back</button>

    </>

  )

}

1. No Match Route

Route will match only when no other routes do (Min kaydire l user chi Route makaynche)

App.js

<>

    <Navbar/>

    <Routes>

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

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

            <Route path='order-summary' element={<OrderSummary/>}></Route>

            <Route path='\*' element={<NoMatch/>}></Route>

    </Routes>

    </>

NoMatch.js

export const NoMatch = () => {

  return (

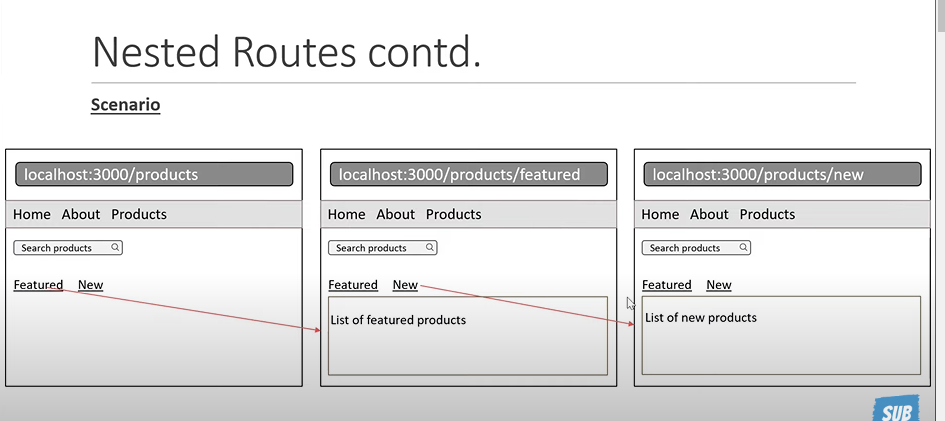
    <div>Page not found</div>

  )

}

1. Nested Routes
   1. Nested Routes

Nested Routes is route in route ../../..



App.js

function App() {

  return (

    <>

    <Navbar/>

    <Routes>

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

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

            <Route path='order-summary' element={<OrderSummary/>}></Route>

            <Route path='products' element={<Products/>}>

{/\* the special of Neted route is that React Router automatically  forms the full path to the childern routes\*/}

               <Route path='feature' element={<FeatureProducts/>}/>

               <Route path='new' element={<NewProducts/>}/>

            </Route>

            <Route path='\*' element={<NoMatch/>}></Route>

    </Routes>

    </>

  );

}

product.js

import React from 'react'

import { Link,Outlet } from 'react-router-dom'

export const Products = () => {

  return (

    <div>

        <input type='search' placeholder='Search a product'/>

        <nav>

            <Link to='feature'>Featured</Link>

            <Link to='new'>   New</Link>

        </nav>

        <Outlet/>

        {/\* Outlet help the parent component to render the child component  \*/}

    </div>

  )

}

FeatureProducts.js

export const FeatureProducts = () => {

  return (

    <div>FeatureProducts</div>

  )

}

NewProducts.js

export const NewProducts = () => {

  return (

    <div>NewProducts</div>

  )

}

* 1. Index

It will help us to render one of the child routes at the perent route level render (B7ale par default il konte f chi parent route radi y afficher lia wahd l child bla man7taje nbrke)

App.js

<Route path='products' element={<Products/>}>

{/\* the special of Neted route is that React Router automatically  forms the full path to the childern routes\*/}

               <Route index element={<FeatureProducts/>}/>

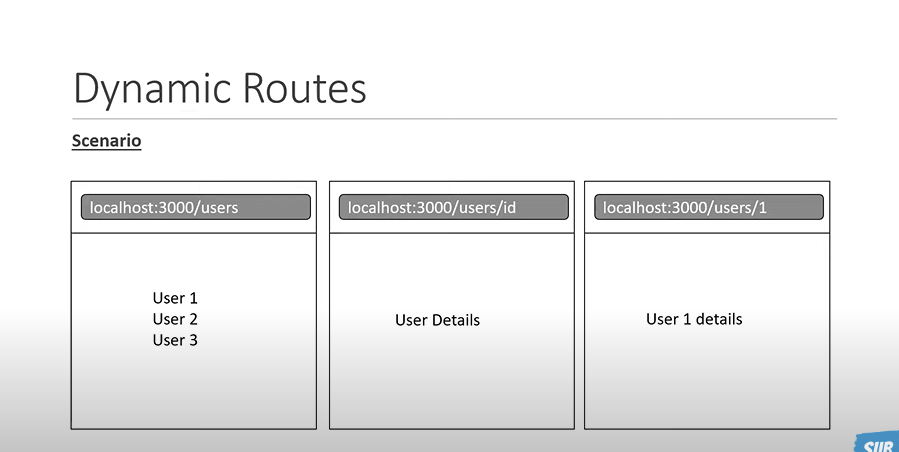
               <Route path='feature' element={<FeatureProducts/>}/>

               <Route path='new' element={<NewProducts/>}/>

            </Route>

1. Dynamic Routes
   1. Dynamic Routes with Url Params

Imagine if the user Navigate to /users we should display a list of three users ,if the user navigate users followed by the id of the user we need to display detail about that user.



Pour Dynamic Routes on peut utilise des nested routes si on veux si non on peut utiliser des routes normal

App.js

<Route path="users" element={<Users />}>

          <Route path=":userId" element={<UserDetails />} />

          <Route path="admin" element={<Admin />} />

        </Route>

Users.js

import { Outlet } from 'react-router-dom'

export const Users = () => {

  return (

    <div>

        <h2>User1</h2>

        <h2>User2</h2>

        <h2>User3</h2>

        <Outlet/>

    </div>

  )

}

UserDetails.js

import { useParams } from 'react-router-dom'

export const UserDetails = () => {

    const params=useParams()

    const userId=params.userId

  return (

    <div>Details about user {userId}</div>

  )

}

Admin.js

export const Admin = () => {

  return (

    <div>Admin user</div>

  )

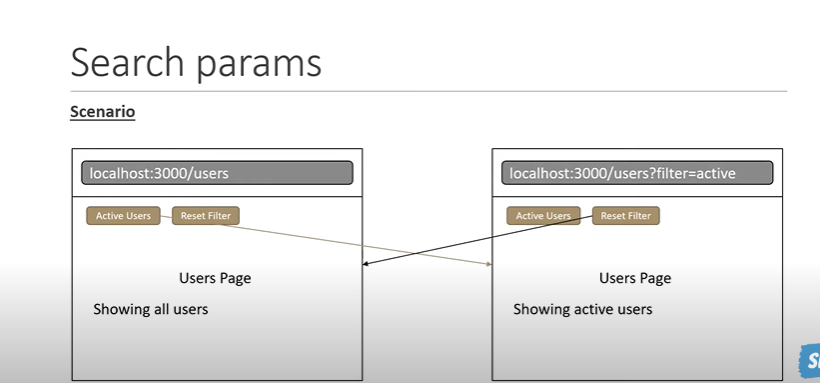
}

Voir user.js pour que tu comprend pourquoi y a admin.js

* 1. Dynamic Routes with search Params

- Url params is not the only way to add parameters to a route we can also add on optional query string

- to deal with react params react router provide a hook called useSearchParams this hook is semelar to useState hook in React instead of storinf state in memory though it is stored in the url, the hook return two value of which th first is a object the second isa fct to set the search pateterns.



Users.js

import { Outlet, useSearchParams } from "react-router-dom";

export const Users = () => {

  //It's similar to useState

  const [searchParams, setSearchParams] = useSearchParams();

  const showActiveUsers = searchParams.get("filter") === "active";

  return (

    <div>

      <h2>User1</h2>

      <h2>User2</h2>

      <h2>User3</h2>

      <Outlet />

      <div>

        <button

          onClick={() => setSearchParams({ filter: "active", test: "Rca" })}

        >

          Active Users

        </button>

        <button onClick={() => setSearchParams({})}>Reset Filter</button>

        {showActiveUsers ? (

          <h2>Showing active user</h2>

        ) : (

          <h2>Showing all user</h2>

        )}

      </div>

    </div>

  );

};

1. Relative Links

<link to =’/about’> 🡺 localhost:3000/about 🡺 this is am absolute path

<link to =’about’> 🡺 localhost:3000/(Ila kante chi 7aja 9ble jaya mn l parent)/about 🡺this is a relative Link is does not start with a forward slash and will in her it the closest route in which they are rendered

1. Lazy Loading

Lazy loading is a technique where components not required on the home page can be split into separate cod bundles and downloaded only when the user navigate to that page ,It help reduce load time thereby improving performance ,for our Example :

App.js

const lazyAbout = React.lazy(() => import("./components/About"));

<Route

          path="about"

          element={<React.Suspense fallback="Loading.."><lazyAbout/></React.Suspense>}

        ></Route>

About.js

import React from "react";

export const About = () => {

  return (

    <div>

      Bzf dial text ………………………………………………………………..

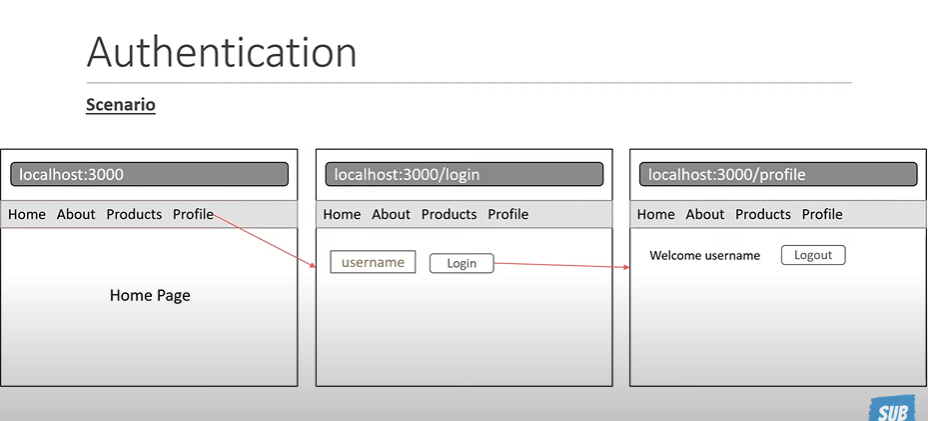
    </div>

  );

};

1. Authentication and Protected Routes

When we building applications we need to protect some route in the app from users who are not loggerd in . in this video we will see how to protect routes using react roter and the context api from react .



We wil summarize the steps ,First we created the profile route which should be protected ,next we implemented the functionality to loGin and logout a user for that we relied on react context and provided the context value to the entire component tree , after that we implement the login page we used the auth context to sign in and the navigate function to redirect in the profil page we used the same auth cotext to display the logged in user and handle the logout button , finally we created the require auth component that checks if the user is logged in or not if the user is not logged in it redirects to the login route ,the user is logged in it render the children prop , we wrap any component that needs to be protected with the requiredAuthn component and this is done when we configure the particular route , last but not the least we used replace to prevent the user from viewing the login page after loggin in and a bit of useLocation magic to keep track of the redirection .

All the necessary file are here

Une image contenant texte

Description générée automatiquement

Auth.js

import React, { useState, createContext, useContext } from "react";

const AuthContext = createContext(null);

export const AuthProvider = ({ children }) => {

  const [user, setUser] = useState(null);

  const logIn = (user) => {

    setUser(user);

  };

  const logOut = () => {

    setUser(null);

  };

  return (

    <AuthContext.Provider value={{ user, logIn, logOut }}>

      {children}

    </AuthContext.Provider>

  );

};

export const useAuth=()=>{

    return useContext(AuthContext)

}

RequireAuth.js

import React from "react";

import { Navigate, useLocation } from "react-router-dom";

import { useAuth } from "./Auth";

export const RequireAuth = ({ children }) => {

  const location = useLocation();

  const auth = useAuth();

  if (!auth.user) {

    return (

      <Navigate to="/login" state={{ path: location.pathname }}/>

    );

  }

  return <div>{children}</div>;

};

Login.js

import React, { useState } from "react";

import { useLocation, useNavigate } from "react-router-dom";

import { useAuth } from "./Auth";

export const Login = () => {

  const [user, setUser] = useState("");

  const auth = useAuth();

  const navigate = useNavigate();

  const location = useLocation();

  const redirectPath = location.state?.path || "/";

  const handleLogin = () => {

    auth.logIn(user);

    navigate(redirectPath, { replace: true });

  };

  return (

    <div>

      <label>

        UserName:

        <input type="text" onChange={(e) => setUser(e.target.value)} />

      </label>

      <button onClick={() => handleLogin()}>Login</button>

    </div>

  );

};

Profil.js

import React from "react";

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

import { useAuth } from "./Auth";

export const Profile = () => {

  const auth = useAuth();

  const navigate = useNavigate();

  const handleLogout = () => {

    auth.logOut();

    navigate('/');

  };

  return (

    <div>

      Welcome {auth.user}

      <button onClick={()=>handleLogout()}>Logout</button>

    </div>

  );

};

App.js

import { Routes, Route } from "react-router-dom";

import { About } from "./components/About";

import { Admin } from "./components/Admin";

import { AuthProvider } from "./components/Auth";

import { FeatureProducts } from "./components/FeatureProducts";

import { Home } from "./components/Home";

import { Login } from "./components/Login";

import { Navbar } from "./components/Navbar";

import { NewProducts } from "./components/NewProducts";

import { NoMatch } from "./components/NoMatch";

import { OrderSummary } from "./components/OrderSummary";

import { Products } from "./components/Products";

import { Profile } from "./components/Profile";

import { RequireAuth } from "./components/RequireAuth";

import { UserDetails } from "./components/userDetails";

import { Users } from "./components/Users";

function App() {

  return (

    <>

      <AuthProvider>

        <Navbar />

        <Routes>

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

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

          <Route path="order-summary" element={<OrderSummary />}></Route>

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

            {/\* the special of Neted route is that React Router automatically  forms the full path to the childern routes\*/}

            <Route index element={<FeatureProducts />} />

            <Route path="feature" element={<FeatureProducts />} />

            <Route path="new" element={<NewProducts />} />

          </Route>

          <Route path="users" element={<Users />}>

            <Route path=":userId" element={<UserDetails />} />

            <Route path="admin" element={<Admin />} />

          </Route>

          <Route

            path="profile"

            element={

              <RequireAuth>

                <Profile />

              </RequireAuth>

            }

          ></Route>

          <Route path="login" element={<Login />}></Route>

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

        </Routes>

      </AuthProvider>

    </>

  );

}

export default App;