REACTJS Day6

Wednesday, November 27, 2024 12:16 PM

What is React Router DOM?
React Router DOM is a library for implementing routing in a **React** web application.
It enables you to create single-page applications (SPAs) where navigation between different views or pages doesn't require a full page reload. Instead, React Router dynamically renders components based on the current URL, maintaining a seamless user experience.

Key Features of React Router DOM

1. **Declarative Routing**: Define routes in a declarative manner using components like <Route>, <Switch>, and <BrowserRouter>.
2. **Dynamic Routing**: Routes are updated dynamically based on user actions or navigation, allowing more interactive and responsive applications.
3. **Nested Routing**: Define nested routes for rendering child components based on the URL hierarchy.
4. **URL Parameters**: Pass and access parameters through the URL, enabling dynamic rendering.
5. **Programmatic Navigation**: Navigate between routes programmatically using the useNavigate hook or other utilities.
6. **Code Splitting**: Supports lazy loading of components to optimize performance.

Main Components in React Router DOM

1. **<BrowserRouter>**: The primary wrapper for enabling routing in web applications. It uses the HTML5 history API for navigation.
2. **<Routes> and <Route>**:
   - <Routes> replaces the older <Switch> component to render the first matching route.
   - <Route> defines the path and the component to render.
3. **useNavigate**: A hook to programmatically navigate between routes.
4. **useParams**: A hook to access route parameters.
5. **useLocation**: A hook to access information about the current URL.
6. **useSearchParams**: A hook to work with query parameters in the URL.
7. **Link and NavLink**:
   - Link: Used to navigate between routes without reloading the page.
   - NavLink: Similar to Link, but adds active styles to the currently selected route.

Dynamic routing in React Router allows you to create routes that depend on parameters, such as user IDs, product IDs, etc. To implement this, use the **useParams** hook from react-router-dom to access route parameters.

```
import React from 'react'
import { useParams } from 'react-router-dom';
function User() {
  let params = useParams();
```

```
  let {name} = params;
  return (
    <div>User : {name}</div>
  )
}
export default User
```

useSearchParam() hook :

The useSearchParams hook from react-router-dom allows you to work with query string parameters in your URL. Query parameters are typically used for things like filtering, sorting, or pagination in your application.

```
import React from 'react'
import { useSearchParams } from 'react-router-dom'
function Filter() {
    const [searchParams,setSearchParams] = useSearchParams();
    const age = searchParams.get('age');
  return (
    <>
        <div>Filter</div>
        <h2>Age : {age}</h2>
        <h2>CIty : {searchParams.get('city')}</h2>
        <input type="text" onChange={(e)=>setSearchParams({age:e.target.value})}
/>
        <button onClick={()=>setSearchParams({age:40})}>Set Age</button>
    </>
  )
}
export default Filter
```

"By using the useNavigate() hook, we can perform navigation."

```
import React from 'react'
import { Link,useNavigate } from 'react-router-dom'
function Home() {
  const navigate = useNavigate();
  const navtoigate = (url) =>{
    if (url == '/filter') {
      navigate("/filter");
    } else {
      navigate("/about");
    }
  }
  return (
    <>
    <h1>Homepage</h1>
    <p>This is homepage of our awesome app</p>
    <Link to='/about'>Go to About Page</Link>
    <br />
    <br />
    {/* <button onClick={()=>navigate("/about")}>Go to about Page</button> */}
    <button onClick={()=>navtoigate("/about")}>Go to about Page</button>
    <br />
    <br />
    {/* <button onClick={()=>navigate("/filter")}>Go to filter page</button> */}
    <button onClick={()=>navtoigate("/filter")}>Go to filter page</button>
    </>
  )
}
export default Home
```

Nested Routing :

```
<Route path="/contact/" element={<Contact></Contact>}>
            <Route path="company" element={<Company></Company>}></Route>
            <Route path="channel" element={<Channel></Channel>}></Route>
          </Route>
```

```
import React from 'react'
import { Link, Outlet } from 'react-router-dom'
function Contact() {
  return (
    <>
    <h1>Contact Page</h1>
    <Link to="company">Company</Link>
    <Link to="channel">Channel</Link>
    <Outlet></Outlet>
    </>
  )
```

```
}
export default Contact
```

In React Router, the <Outlet /> component is used to render child routes within a parentroute.It serves as a placeholder for where the matched child route's component should be displayed.

The useLocation() hook in React Router is used to access the current location object, which contains information about the URL. This hook is especially useful for understanding the current pathname, search parameters, or state associated with the navigation.

Structure of useLocation Object
The useLocation() hook returns an object with the following properties:

- **pathname**: The path of the current URL (e.g., /dashboard/profile).
- **search**: The query string in the URL (e.g., ?key=value).
- **hash**: The hash fragment in the URL (e.g., #section1).
- **state**: Any state passed during navigation (optional).

When to Use useLocation()

- To retrieve query parameters for filtering or sorting data.
- To access state passed during navigation.
- To track the user's current location for analytics or logging.
- To dynamically update UI based on the current route.

Example :

```
import React from 'react'
import { useSearchParams,useLocation } from 'react-router-dom'
function Filter() {
    const [searchParams,setSearchParams] = useSearchParams();
    const age = searchParams.get('age');
    const location = useLocation();
    console.log(location);

  return (
    <>
        <div>Filter</div>
        <h2>Age : {age}</h2>
        <h2>CIty : {searchParams.get('city')}</h2>
        <input type="text" onChange={(e)=>setSearchParams({age:e.target.value})}
/>
        <button onClick={()=>setSearchParams({age:40})}>Set Age</button>
    </>
  )
}
export default Filter
```

O/P :

1. {pathname: '/filter/', search: '?age=40', hash: '', state: null, key: 'e3mdkkp8'}
    1. hash: ""
    2. key: "e3mdkkp8"
    3. pathname: "/filter/"
    4. search: "?age=40"
    5. state: null
    6. [[Prototype]]: Object

How to protect route :

```
import { BrowserRouter, Route, Routes } from "react-router-dom";
import Home from "./component/Home";
import About from "./component/About";
import Navbar from "./component/Navbar";
import PageNot from "./component/PageNot";
import User from "./component/User";
import Filter from "./component/Filter";
import Contact from "./component/Contact";
import Company from "./component/Company";
import Channel from "./component/Channel";
import Login from "./component/Login";
import Protected from "./component/Protected";
function App() {
```

```
  return (
    <div className="App">
      <BrowserRouter>
        <Navbar />
        <Routes>
          {/* Home Page */}
          <Route path="/" element={<Protected Cmp={Home} />} />
          {/* About Page */}
          <Route path="/about" element={<Protected Cmp={About} />} />
          {/* User Page with Dynamic Parameter */}
          <Route path="/user/:name" element={<Protected Cmp={User} />} />
          {/* Filter Page */}
          <Route path="/filter" element={<Protected Cmp={Filter} />} />
          {/* Contact Page with Nested Routes */}
          <Route path="/contact" element={<Contact></Contact>}>
            <Route path="company" element={<Company />} />
            <Route path="channel" element={<Channel />} />
          </Route>
          {/* Login Page */}
          <Route path="/login" element={<Login />} />
          {/* 404 Page Not Found */}
          <Route path="/*" element={<PageNot />} />
        </Routes>
      </BrowserRouter>
    </div>
  );
}
export default App;
```

```
import React, { useEffect } from 'react'
import { useNavigate } from 'react-router-dom';
function Protected(props) {
    const  {Cmp} = props;
    const navigate = useNavigate();
    useEffect(()=>{
        const login = localStorage.getItem('login');
        if (!login) {
            navigate("/login")
        }
    })
  return (
    <>
        <div>
            <Cmp/>
        </div>
    </>
  )
}
export default Protected
```

```
import React, { useEffect } from 'react'
import { useNavigate } from 'react-router-dom';
function Login() {
    const login=()=>{
        localStorage.setItem("login",true);
        navigate("/");
    }
    const navigate = useNavigate();
    useEffect(()=>{
        const login = localStorage.getItem('login');
        if (login) {
            navigate("/")
        }
    },[navigate])
  return (
    <>
        <div style={{display:'grid', justifyContent:'center',
alignItems:'center'}}>
        <h1>Login Form</h1>
        <input type="text"/>
        <br />
        <input type="password"/>
        <br />
        <button onClick={login}>Submit</button>
        </div>
    </>
  )
}
export default Login
```

Another way also availabel.

Interview qustions :

1] Where we can use react routing?
Ans :
On-web
On sever-side with node.js
With react native

2] Link v/s Navlink
Ans :
With navlink we can add custom style and class
With navlink with can add active link feacture
style is not recommented with link

3] What is nested routes?
Make route inside route
we can makes tab or pages inside pages
multiple nested routes also possible

4] How to add n404 page ?
<Route path="/*" element={<PageNot />} />

5] Some important hooks
useParamsuseSearchParams
useLocation
useNavigate

6] Redirect on button :
<button onClick={()=>navigate("/about")}>Go to about Page</button>

7] <a> v/s <Link>
No refresh with link
refresh with link

8] Hash Route v/s Browser Route
Mostly use browser route
hash route used for hiding from server
example : localhost:3000/about
example of hash : localhost:3000#about