Slim

React js tips

React Router DOM V6 Basics

Quick & Easy Guide for Beginners!

Slim toumi

Slim toumi

Slim

React js tips

Why Use React Router?

React Router DOM is a tool that helps you add multiple pages to your React app, so users can click around without reloading the page. It’s perfect for creating a smooth, fast experience in a Single Page Application (SPA), where only parts of the page update as users navigate.

With **React Router DOM**, you can :

- **Set Up Different Pages:** Create routes so users can visit URLs like /home or /about to see specific content.
- **Link Pages Together:** Use links to let users navigate easily without reloading the page.
- **Handle 404 Pages:** Show a custom "Page Not Found" message if someone tries to visit a page that doesn’t exist.

Install React Router DOM

In your project folder, open the terminal and install React Router with this command:

```
npm install react-router-dom
```

Slim toumi
Slim toumi

Slim

React js tips

Basic Setup in App.js

In your main App.js file, you need to set up BrowserRouter, Routes, and Route to define each page (or route) in your app.

app.js

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Home from './Home';
import About from './About';

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Explanation

- **BrowserRouter:** Wraps the entire app to enable routing.
- **Routes:** Holds all individual <Route> elements.
- **Route:** Defines a page. The path is the URL, and element is the component to display.

Slim toumi

Slim toumi

Slim

React js tips

Creating Page Components

Let’s create two simple pages: **Home** and **About**.
In your src folder, make two new files: **Home.js** and **About.js.**

Home.js :

Home.js

```
function Home() {
  return <h1>Welcome to the Home Page!</h1>;
}

export default Home;
```

About.js :

About.js

```
function About() {
  return <h1>This is the About Page!</h1>;
}

export default About;
```

Slim toumi

Slim toumi

Slim

React js tips

Adding Navigation Links

To move between pages without reloading, use **<Link>**. Add a simple navigation menu using **<Link>** in a new component called **Navbar.js**.

Navbar.js :

Navbar.js

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

export default  function Navbar() {
  return (
    <nav>
      <Link to="/">Home</Link>
      <Link to="/about">About</Link>
    </nav>
  );
}
```

Explanation

- **Link**: Creates clickable navigation to different pages in the app.
- **to** : Specifies the URL path to navigate to when clicked.
- **Difference from <a> Tag:** Unlike <a>, which reloads the entire page, Link updates the URL and content without a full reload, ensuring a smoother user experience while preserving the application state.

Slim toumi

Slim toumi

Slim

React js tips

Now, add this **<Navbar />** component to App.js so it appears on every page:

Navbar.js :

App.js

```
import Navbar from './Navbar';

export default function App() {
  return (
    <BrowserRouter>
      <Navbar />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Explanation

- **Navbar Outside of Routes:** By placing **<Navbar />** outside of **<Routes>**, the navigation bar is always displayed, no matter which page or route the user is on. This makes the navbar accessible on all pages.

Slim toumi

Slim toumi

Slim

React js tips

Handling Unknown Routes (404 Page)

To handle routes that don't exist, you can display a message using a simple NotFound component.

NotFound.js :

NotFound.js

```
import React from 'react';

export default function NotFound() {
  return <h1>404 - Page Not Found</h1>;
}
```

Import the **NotFound** component in **App.js** and include it in the JSX. The * in the route path serves as a catch-all for undefined routes, directing users to the **NotFound** component.

App.js :

App.js

```
import NotFound.js from './NotFound'
.
.
// In Routes (in App.js)
<Route path="*" element={<NotFound />} />
```

Slim toumi

Slim toumi

Slim

React js tips

Navigating with a Button (Optional)

You can also navigate with a button click. Use the **useNavigate** hook to do this.

About.js :

About.js

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

export default function About() {
  const navigate = useNavigate();

  return (
    <div>
      <h1>About Page</h1>
      <button onClick={() => navigate('/')}>Home</button>
    </div>
  );
}
```

Explanation

- The **useNavigate** hook from **react-router-dom** allows you to programmatically navigate between routes in your application.
- In this example, it creates a button that navigates the user back to the home page ('/') when clicked, allowing for smooth transitions without traditional link elements.

Slim toumi

Slim toumi

Slim

React js tips

Your Final Project Structure

Your **src** folder should now look like this:

```
src
├── App.js          // Main app with routing
├── Navbar.js       // Navigation links
├── Home.js         // Home page component
├── About.js        // About page component
└── NotFound.js     // 404 Not Found page
```

This is a basic structure of your React application. You can further develop it by creating additional folders and components as needed to organize your code better.

Slim toumi

Slim toumi

Slim

React js tips

Hopefully You Found It Usefull!

"Be sure to save this post so you can come back to it later"

like Comment Share

Slim toumi

Slim toumi