Router

* Router

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

* Advantages of Router
* Routing between components is fast as the amount of data that renders is less. The rest of the data is rendered by the DOM, and even when there's tons of HTML and CSS to render, the DOM handles that part in the blink of an eye. Using lazy loading, any delay in rendering HTML is compensated for.
* For better user experience, animations and transitions can be easily implemented when switching between different components.
* It gives a real sense of a single-page application in action. No separate pages are rendered, and the current page doesn't refresh to load a new view.
* Router Setup

**Step 1 - Install a React Router**

## Add React Router

To add React Router in your application, run this in the terminal from the root directory of the application:

npm i -D react-router-dom

## Step 2 - Create Components

## Folder Structure

To create an application with multiple page routes, let's first start with the file structure.

Within the src folder, we'll create a folder named pages with several files:

src\pages\:

* Layout.js
* Home.js
* Blogs.js
* Contact.js
* NoPage.js

Each file will contain a very basic React component.

## Basic Usage

Now we will use our Router in our index.js file.

### **Example**

Use React Router to route to pages based on URL:

index.js:

import ReactDOM from "react-dom/client";

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

import Layout from "./pages/Layout";

import Home from "./pages/Home";

import Blogs from "./pages/Blogs";

import Contact from "./pages/Contact";

import NoPage from "./pages/NoPage";

export default function App() {

return (

<BrowserRouter>

<Routes>

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

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

<Route path="blogs" element={<Blogs />} />

<Route path="contact" element={<Contact />} />

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

</Route>

</Routes>

</BrowserRouter>

);

}

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

root.render(<App />);

## Example Explained

We wrap our content first with <BrowserRouter>.

Then we define our <Routes>. An application can have multiple <Routes>. Our basic example only uses one.

<Route>s can be nested. The first <Route> has a path of / and renders the Layout component.

The nested <Route>s inherit and add to the parent route. So the blogs path is combined with the parent and becomes /blogs.

The Home component route does not have a path but has an index attribute. That specifies this route as the default route for the parent route, which is /.

Setting the path to \* will act as a catch-all for any undefined URLs. This is great for a 404 error page.

## Pages / Components

The Layout component has <Outlet> and <Link> elements.

The <Outlet> renders the current route selected.

<Link> is used to set the URL and keep track of browsing history.

Anytime we link to an internal path, we will use <Link> instead of <a href="">.

The "layout route" is a shared component that inserts common content on all pages, such as a navigation menu.

Layout.js:

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

const Layout = () => {

return (

<>

<nav>

<ul>

<li>

<Link to="/">Home</Link>

</li>

<li>

<Link to="/blogs">Blogs</Link>

</li>

<li>

<Link to="/contact">Contact</Link>

</li>

</ul>

</nav>

<Outlet />

</>

)

};

export default Layout;

Home.js:

const Home = () => {

return <h1>Home</h1>;

};

export default Home;

Blogs.js:

const Blogs = () => {

return <h1>Blog Articles</h1>;

};

export default Blogs;

Contact.js:

const Contact = () => {

return <h1>Contact Me</h1>;

};

export default Contact;

NoPage.js:

const NoPage = () => {

return <h1>404</h1>;

};

export default NoPage;