React Router DOM - Complete Guide

Installation

bash

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

Core Concepts

1. Setup & Configuration

- Import necessary components in your main entry file (e.g., main.jsx)
- Create a router using `createBrowserRouter`
- Use `RouterProvider` to provide routing context to your app

2. Route Configuration

Two methods to define routes:

1. **Object-based** (as shown in your code):

   jsx

   ```jsx
   const router = createBrowserRouter([
     {
       path: '/',
       element: <Layout />,
       children: [
         {
           path: '',  // index route
           element: <Home />
         },
         // other routes...
       ]
     }
   ]);
   ```

2. **JSX-based** (commented out in your code):

jsx

```jsx
const router = createBrowserRouter(
  createRoutesFromElements(
    <Route path="/" element={<Layout />}>
      <Route index element={<Home />} />
      <Route path="about" element={<About />} />
    </Route>
  )
);
```

3. Core Components

`<RouterProvider>`

- Entry point to render your router
- Takes a router object as prop
- Example: `<RouterProvider router={router} />`

`<Outlet>`

- Acts as a placeholder for child routes
- Used in parent layout components
- Child routes specified in router config render here

`<Link>`

- Creates navigation links without full page reloads
- Example: `<Link to="/about">About</Link>`

`<NavLink>`

- Extended version of `Link` with active state awareness
- Can apply styling based on whether route is active
- Example from your code:

```jsx
<NavLink
  to="/"
  className={({isActive}) =>
    `block py-2 pr-4 pl-3 ${isActive ? "text-orange-700" : "text-gray-700"}`
  }
>
  Home
</NavLink>
```

4. Route Parameters

- Dynamic segments in URLs captured as parameters
- Defined with colon syntax: `path: 'user/:userId'`
- Access via `useParams()` hook in component

5. Nested Routes

- Parent routes can have child routes
- Children render within parent's `<Outlet>`
- Maintains UI consistency with shared layouts

6. Layout Pattern

- Use a Layout component with header, footer, and `<Outlet>`
- Children routes render in the `<Outlet>` position
- Keeps consistent UI elements across route changes

Implementation Example (From Your Code)

File Structure

```
├── src/
│   ├── components/
│   │   ├── header/
│   │   │   └── Header.jsx
│   │   ├── footer/
│   │   │   └── Footer.jsx
│   │   ├── home/
│   │   │   └── Home.jsx
│   │   ├── about/
│   │   │   └── About.jsx
│   │   └── user/
│   │       └── User.jsx
│   ├── Layout.jsx
│   ├── App.jsx
│   ├── main.jsx
│   └── index.css
│
```

Router Setup (main.jsx)

1. Import required modules
2. Define routes with createBrowserRouter
3. Wrap your app with RouterProvider

Layout Pattern (Layout.jsx)

```jsx
function Layout() {
  return (
    <>
      <Header/>
      <Outlet/>  // Child routes render here
      <Footer/>
    </>
  )
}
```

Navigation (Header.jsx)

- `<Link>` for basic navigation
- `<NavLink>` for navigation with active states
- Apply conditional styling based on active route

Best Practices

1. **Consistent Layout Structure**
    - Use Layout components with `<Outlet>` for consistent UI
2. **Dynamic Navigation Highlighting**
    - Use `<NavLink>` with `isActive` for visual feedback
3. **Organized Route Configuration**
    - Group related routes together
    - Use nested routes for related content
4. **Responsive Navigation**
    - Implement responsive designs with proper breakpoints (lg:flex-row, etc.)
5. **Clean URLs**
    - Use semantic paths for better usability and SEO

Advanced Features

1. **Route Parameters** (`user/:userId`)
    - Access with `useParams()` hook
2. **Programmatic Navigation**
    - Import and use `useNavigate()` hook
    - Example: `const navigate = useNavigate(); navigate('/about');`
3. **Route Guards/Protection**
    - Implement with nested routes and conditional rendering
4. **Loaders and Actions** (React Router 6.4+)
    - Data fetching tied to routes
    - Form handling with actions
5. **Error Boundaries**
    - Add errorElement to routes for error handling