# React Router

## 1. Introduction to Routing in React

Routing is the process of navigating between different views or pages in an application. In traditional websites, clicking a link triggers a full-page reload, but in React applications, routing is handled client-side without a full-page refresh, which creates a smoother, faster user experience.  
React Router allows us to define different routes in the app and navigate between them easily. With React Router, you can:  
• Define routes for different views or components.  
• Navigate between routes without refreshing the page.  
• Pass data between components (through route parameters or state).  
• Implement nested routing to create multi-level routes.

**2. Setting Up React Router**

To get started with React Router, you need to install it first. Follow these steps:

1. **Install React Router:**  
   Open your terminal and navigate to your React app directory. Then install React Router:

npm install react-router-dom

1. **Wrap your app with Router:**  
   In your src/index.js or src/main.js, wrap your app with the Router component. This will allow you to use routing throughout your application.

**Example (src/index.js):**

import React from 'react';

import ReactDOM from 'react-dom';

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

import App from './App';

ReactDOM.render(

<Router>

<App />

</Router>,

document.getElementById('root')

);

In this case, we’re using BrowserRouter (often aliased as Router) to handle the routing. This sets up history API-based routing in the browser.

**3. Using <Route>, <Link>, <Routes>, and useParams**

**<Route> Component**

The <Route> component is the core of routing in React. It defines a mapping between a URL and a React component to render when that URL is matched.

**Basic Example:**

import React from 'react';

import { Route, BrowserRouter as Router, Routes } from 'react-router-dom';

function Home() {

return <h1>Home Page</h1>;

}

function About() {

return <h1>About Page</h1>;

}

function App() {

return (

<Router>

<Routes>

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

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

</Routes>

</Router>

);

}

export default App;

**Explanation:**  
• <Route>: Each <Route> component listens for a path (e.g., /home) and renders the corresponding component (e.g., Home).  
• **<Routes>**: In React Router v6, the <Switch> component has been replaced with <Routes>. <Routes> ensures that only one <Route> is rendered at a time.  
• **Note:** React Router v6 introduced a new way of defining components to be rendered using the element prop, instead of component.

**Dynamic Routing with useParams**

You can create dynamic routes where the URL can have variables. For example, a user profile page can have a URL like /user/:userId, where userId is a dynamic parameter.

**Example: Using useParams for Dynamic Routes**

import React from 'react';

import { Route, BrowserRouter as Router, Routes, useParams } from 'react-router-dom';

function UserProfile() {

const { userId } = useParams(); // Retrieve the dynamic parameter from the URL

return <h1>User Profile for {userId}</h1>;

}

function App() {

return (

<Router>

<Routes>

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

</Routes>

</Router>

);

}

export default App;

**Explanation:**  
• useParams(): This hook allows you to access the dynamic parameters from the URL (e.g., userId).  
• In the example above, if the URL is /user/123, the UserProfile component will render User Profile for 123.

**<Link> Component**

The <Link> component is used to navigate between different routes without causing a full-page reload. It’s like an anchor (<a>) tag, but it leverages React Router's client-side navigation.

**Example:**

import React from 'react';

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

function App() {

return (

<div>

<h1>Welcome to React Router!</h1>

<nav>

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

<Link to="/about">About</Link>

</nav>

</div>

);

}

export default App;

**Explanation:**  
• <Link to="/path">: When clicked, the <Link> component navigates to the specified route (/home or /about), without triggering a full-page reload.

**<Routes> Component**

React Router v6 replaces <Switch> with <Routes>. The <Routes> component ensures that only the first matched route gets rendered.

**Example with <Routes>:**

import React from 'react';

import { Route, BrowserRouter as Router, Routes } from 'react-router-dom';

function Home() {

return <h1>Home Page</h1>;

}

function About() {

return <h1>About Page</h1>;

}

function NotFound() {

return <h1>Page Not Found</h1>;

}

function App() {

return (

<Router>

<Routes>

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

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

<Route path="\*" element={<NotFound />} /> {/\* Fallback route if no match \*/}

</Routes>

</Router>

);

}

export default App;

**Explanation:**  
• <Routes> ensures that only the first matched route gets rendered.  
• If no route matches, the fallback route (path="\*") will render the NotFound component.

**4. Nested Routing and Dynamic Routes**

**Nested Routes**

You can define nested routes inside a parent route, breaking down complex pages into smaller components and creating a hierarchy of views.

**Example: Nested Routes**

import React from 'react';

import { Route, BrowserRouter as Router, Routes, Link } from 'react-router-dom';

function Dashboard() {

return (

<div>

<h1>Dashboard</h1>

<nav>

<Link to="/dashboard/overview">Overview</Link> |

<Link to="/dashboard/settings">Settings</Link>

</nav>

<Routes>

<Route path="/dashboard/overview" element={<Overview />} />

<Route path="/dashboard/settings" element={<Settings />} />

</Routes>

</div>

);

}

function Overview() {

return <h2>Overview Page</h2>;

}

function Settings() {

return <h2>Settings Page</h2>;

}

function App() {

return (

<Router>

<Route path="/dashboard" element={<Dashboard />} />

</Router>

);

}

export default App;

**Explanation:**  
• The Dashboard component has two nested routes: /dashboard/overview and /dashboard/settings.  
• These routes are only accessible when the user visits /dashboard, and the component renders based on the active route.

**Dynamic Nested Routes**

Nested routes can also be dynamic, meaning that parts of the route can change based on parameters in the URL.

**Example: Dynamic Nested Routes**

import React from 'react';

import { Route, BrowserRouter as Router, Routes, useParams } from 'react-router-dom';

function UserProfile() {

const { userId } = useParams();

return <h1>User Profile for {userId}</h1>;

}

function App() {

return (

<Router>

<Routes>

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

</Routes>

</Router>

);

}

export default App;

**Explanation:**  
• The UserProfile component dynamically renders the profile based on the userId in the URL (e.g., /user/123 will render "User Profile for 123").