React Router Basics

Adding page navigation to a React app with react-router -- BrowserRouter, Routes, Route, Link, NavLink, and a Not Found page for undefined URLs, with simple examples.

Intermediate 12 min

Last updated: August 18, 2026
Read online: https://www.learnforgex.com/en/topics/react-router-basics

---

React Router Basics

Every example we've seen so far has been a SINGLE page. Real applications usually have more than one "page" -- a home page, a course list, an about page. This lesson covers how to do that in React, with the **React Router** library.

Why Do We Need a Router?

In a classic website, every page (`/` , `/courses` , `/about`) is a separate HTML file -- clicking a link makes the browser load a NEW page. React applications, on the other hand, run as a **single HTML file** (a Single Page Application, SPA); "changing pages" actually means rendering DIFFERENT components on that same page, based on the URL. We use the **react-router** library to manage this -- it reads the URL and decides which component to show.

Defining Pages with BrowserRouter and Routes

The first step to using a router in a React app is wrapping the application in `BrowserRouter` , then adding `Routes` and `Route` s inside it:

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

function Home() {
  return <h1>Home Page</h1>;
}

function Courses() {
  return <h1>Courses Page</h1>;
}

function BasicRouterSetupExample() {
  return (
    <BrowserRouter>
      {/* Routes decides WHICH Route to render based on the URL -- only the
          matching one is rendered at any given time. */}
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/courses" element={<Courses />} />
      </Routes>
    </BrowserRouter>
  );
}
```

`BrowserRouter` is the component that watches the browser's URL and informs React of changes. `Routes` looks at the URL to figure out WHICH `Route` matches -- each `Route` has a `path` (a URL pattern) and an `element` (the component to show for that URL). Only the matching `Route` is rendered at any given time.

Moving Between Pages with Link

We DON'T use `<a href="...">` to move between pages -- that makes the browser reload the entire page. Instead, we use react-router's `Link` component:

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

function Home() {
  return (
    <div>
      <h1>Home Page</h1>
      {/* We use <Link to="..."> instead of <a href="..."> -- Link changes
          the URL WITHOUT RELOADING the page (no full page reload), React
          only re-renders the part that needs to change. */}
      <Link to="/courses">View Courses</Link>
    </div>
  );
}

function Courses() {
  return (
    <div>
      <h1>Courses Page</h1>
      <Link to="/">Back to Home</Link>
    </div>
  );
}

function LinkNavigationExample() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/courses" element={<Courses />} />
      </Routes>
    </BrowserRouter>
  );
}
```

`<Link to="/courses">` looks and behaves like an `<a>` tag on screen, but clicking it does NOT reload the page -- it only changes the URL, and React renders the matching `Route` in response. This makes navigation much faster and smoother.

Highlighting the Active Page with NavLink

In a navigation menu, we usually want to show WHICH page the user is currently on -- for that, we use `NavLink` instead of `Link`:

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

function Home() {
  return <h1>Home Page</h1>;
}

function Courses() {
  return <h1>Courses Page</h1>;
}

function NavLinkActiveExample() {
  return (
    <BrowserRouter>
      <nav>
        {/* NavLink does the same job as Link -- but by passing a function
            to className, we can distinguish the "I'm currently on this
            link's page" state (isActive). */}
        <NavLink
          to="/"
          className={({ isActive }) => (isActive ? "nav-link active" : "nav-link")}
        >
          Home
        </NavLink>
        <NavLink
          to="/courses"
          className={({ isActive }) => (isActive ? "nav-link active" : "nav-link")}
        >
          Courses
        </NavLink>
      </nav>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/courses" element={<Courses />} />
      </Routes>
    </BrowserRouter>
  );
}
```

`NavLink` works exactly like `Link`, but lets you pass a FUNCTION to `className` (or `style`); that function receives an `{ isActive }` object telling you whether that link's page is the current one.

Combining Multiple Pages

A real application usually has several pages together with a shared navigation menu:

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

function Home() {
  return <h1>Home</h1>;
}

function Courses() {
  return <h1>Courses</h1>;
}

function About() {
  return <h1>About</h1>;
}

function MultiPageNavExample() {
  return (
    <BrowserRouter>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/courses">Courses</Link>
        <Link to="/about">About</Link>
      </nav>

      {/* Three separate pages, three separate Routes -- each renders its
          own component, and React switches between them as the URL
          changes. */}
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/courses" element={<Courses />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Here there are three separate `Route`s (`/`, `/courses`, `/about`) and three `Link`s pointing to them -- this is the basic skeleton of a small multi-page application.

Unmatched URLs: A Not Found Page

What happens if a user goes to a URL that isn't defined (like `/does-not-exist`)? A special `Route` catches this:

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

function Home() {
  return <h1>Home</h1>;
}

function Courses() {
  return <h1>Courses</h1>;
}

function NotFound() {
  return (
    <div>
      <h1>404 - Page Not Found</h1>
      <Link to="/">Go back home</Link>
    </div>
  );
}

function NotFoundRouteExample() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/courses" element={<Courses />} />
        {/* path="*" CATCHES any URL that doesn't match any of the other
            defined Routes -- it's always written LAST inside Routes. */}
        <Route path="*" element={<NotFound />} />
      </Routes>
    </BrowserRouter>
  );
}
```

`path="*"` catches every URL that doesn't match any OTHER `Route` -- that's why it's always written as the LAST `Route` inside `Routes`; React looks for a match from top to bottom, in order.

Summary and Glossary

`BrowserRouter` wraps the application and watches the URL; `Routes` and `Route` decide which component to show based on that URL. To move between pages, we use `Link` instead of `<a>` (or `NavLink` if we need to highlight the active page) -- both change the URL without reloading the page. Undefined URLs are caught by a `Route` written with `path="*"` and placed at the end of `Routes`.

Glossary

SPA (Single Page Application) — An application that runs on a single HTML file, managing "page changes" with JavaScript instead of reloading the browser.

Route — A definition that maps a URL pattern (`path`) to a specific component (`element`).

Client-Side Routing — Managing URL changes in the browser with JavaScript, without sending a new request to the server.