Routing in Web Apps – React 18

React Router v6 • Minimal, modern, and declarative

Why Routing Matters

- Organize UI into logical pages (Dashboard / Profile / Settings)
- Enable Single-Page App (SPA) behavior with fast navigation
- Preserve state across views
- Example paths:
  - /home • /users/:id • /settings/profile

Core Concepts – React Router v6 (Web)

- <BrowserRouter> wraps your app
- <Routes> & <Route path element> define pages
- <Link to> navigates declaratively
- Hooks:
  - useParams() – read dynamic URL segments
  - useNavigate() – programmatic navigation
  - useLocation() – URL & search state

Core Concepts – React Router v6 (Web)

```
import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
import Home from "./Home";
import User from "./User";

export default function App() {
  return (
    <BrowserRouter>
      <nav>
        <Link to="/">Home</Link>{" "}
        <Link to="/user/42">User 42</Link>
      </nav>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/user/:id" element={<User />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Hooks for Navigation (React 18)

- useNavigate() → redirect after actions (e.g., login, save)
- useParams() → read dynamic segments like :id
- useLocation() → access pathname and search params

Hooks for Navigation (React 18)

```
import { useNavigate, useParams, useLocation } from "react-router-dom";

export default function UserPage() {
  const { id } = useParams();
  const navigate = useNavigate();
  const location = useLocation();

  const goHome = () => navigate("/");

  return (
    <section>
      <h2>User {id}</h2>
      <p>Current path: {location.pathname}</p>
      <button onClick={goHome}>Go Home</button>
    </section>
  );
}
```

Nested Routes & Shared Layout

- Use <Outlet /> inside a layout to render child routes
- Great for shared headers/sidebars
- Keeps code modular and pages focused

Nested Routes & Shared Layout

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

function DashboardLayout() {
  return (
    <div style={{ display: "grid", gridTemplateColumns: "200px 1fr", gap: 16 }}>
      <aside>
        <h3>Dashboard</h3>
        <nav>
          <Link to="profile">Profile</Link><br/>
          <Link to="settings">Settings</Link>
        </nav>
      </aside>
      <main>
        <Outlet /> {/* Child route renders here */}
      </main>
    </div>
  );
}

export default function AppRoutes() {
  return (
    <Routes>
      <Route path="/dashboard" element={<DashboardLayout />}>
        <Route path="profile" element={<Profile />} />
        <Route path="settings" element={<Settings />} />
      </Route>
    </Routes>
  );
}
```

Recap & Tips

- Define routes declaratively with <Routes>/<Route>
- Use hooks (useNavigate, useParams, useLocation) for logic
- Build shared layouts with nested routes and <Outlet />
- Keep routes small, focused, and composable
- React 18 + React Router v6 = clean SPA navigation