Web Dev Mastery

1️⃣ `Link` vs `NavLink` — Key Difference

Feature	`Link`	`NavLink`
Purpose	Navigate between routes	Navigate and **apply styles for active route**
Active Styles	❌ Not automatic	✅ Can detect active route using `isActive`
Typical Use	Simple navigation links	Navigation bars / menus with active highlighting

---

2️⃣ Example App

File: `App.jsx`

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

// Pages
function Home() {
  return <h2>Home Page</h2>;
}
function About() {
  return <h2>About Page</h2>;
}
function Contact() {
  return <h2>Contact Page</h2>;
}

function App() {
  return (
    <BrowserRouter>
      <div style={{ padding: "20px" }}>
        <h1>React Router v7 Demo</h1>

        {/* ===== Using Link ===== */}
        <div>
          <h3>Simple Links:</h3>
          <Link to="/">Home</Link> |{" "}
          <Link to="/about">About</Link> |{" "}
          <Link to="/contact">Contact</Link>
        </div>

        {/* Open Link in new tab */}
        <div>
          <h4>Open in New Tab (Link):</h4>
          <Link to="/about" target="_blank" rel="noopener noreferrer">
            About Page
          </Link>
        </div>

        <hr />

        {/* ===== Using NavLink ===== */}
        <div>
          <h3>Navigation Bar (NavLink with active styles):</h3>
          <NavLink
            to="/"
            end
            style={({ isActive }) => ({
              color: isActive ? "white" : "blue",
              backgroundColor: isActive ? "green" : "transparent",
              padding: "5px 10px",
              borderRadius: "5px",
              textDecoration: "none",
            })}
          >
            Home
          </NavLink>{" "}
          |{" "}
          <NavLink
            to="/about"
            style={({ isActive }) => ({
              color: isActive ? "white" : "blue",
              backgroundColor: isActive ? "green" : "transparent",
              padding: "5px 10px",
              borderRadius: "5px",
              textDecoration: "none",
            })}
          >
            About
          </NavLink>{" "}
          |{" "}
          <NavLink
            to="/contact"
```

```
            style={({ isActive }) => ({
              color: isActive ? "white" : "blue",
              backgroundColor: isActive ? "green" : "transparent",
              padding: "5px 10px",
              borderRadius: "5px",
              textDecoration: "none",
            })}
          >
            Contact
          </NavLink>
        </div>

        {/* Open NavLink in new tab */}
        <div>
          <h4>Open in New Tab (NavLink):</h4>
          <NavLink
            to="/contact"
            target="_blank"
            rel="noopener noreferrer"
            style={{ color: "purple" }}
          >
            Contact Page
          </NavLink>
        </div>

        <hr />

        {/* Routes */}
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/contact" element={<Contact />} />
        </Routes>
      </div>
    </BrowserRouter>
  );
}

export default App;
```

3️⃣ Key Points in Code

1. `Link`
    - Simple navigation.

Can open a page in a **new tab** by adding:

```
target="_blank" rel="noopener noreferrer"
```

    -

2. `NavLink`
    - Automatically detects **active route**.
    - Use `style={({ isActive }) => ...}` or `className={({ isActive }) => ...}` to style active links.
    - Can also open in **new tab** with the same `target="_blank"`.
3. **`end` prop in NavLink**
    - Ensures exact path matching for root `/`.
    - Without `end`, `/about` might also activate `/`.

✅ Summary

Task	How to do it
Simple link to route	`<Link to="/about">About</Link>`
Open link in new tab	`<Link to="/about" target="_blank" rel="noopener noreferrer">About</Link>`
Navigation with active styles	`<NavLink to="/about" style={({ isActive }) => ({color: isActive ? "red" : "blue"})}>About</NavLink>`
Open NavLink in new tab	`<NavLink to="/about" target="_blank" rel="noopener noreferrer">About</NavLink>`

React Router v7 Nested Routes – Products Example

1️⃣ BEFORE `<Outlet />` (Child routes won't render)

Here, the `Products` component **does not have `<Outlet />`**, so child routes like `/products/phones` or `/products/laptops` will **not appear**.

```
// App.jsx
import React from "react";
import { BrowserRouter, Routes, Route, Link } from "react-router-dom";

// Layout component
function Layout() {
  return (
    <div>
      <header>
        <h1>My Shop</h1>
        <nav>
          <Link to="/">Home</Link> |{" "}
          <Link to="/products">Products</Link> |{" "}
          <Link to="/about">About</Link>
        </nav>
      </header>

      <main>
        {/* ❌ Child routes will NOT render here */}
      </main>

      <footer>
        <p>© 2025 My Shop</p>
      </footer>
    </div>
  );
}

// Pages
function Home() {
  return <h2>Welcome to My Shop!</h2>;
```

```
}

function About() {
  return <h2>About Us</h2>;
}

// Products parent page without Outlet
function Products() {
  return (
    <div>
      <h2>Products Page</h2>
      <nav>
        <Link to="phones">Phones</Link> |{" "}
        <Link to="laptops">Laptops</Link> |{" "}
        <Link to="camera">Camera</Link>
      </nav>
      {/* ❌ No Outlet → child categories will NOT render */}
    </div>
  );
}

// Child pages
function Phones() {
  return <h3>Phones Category</h3>;
}

function Laptops() {
  return <h3>Laptops Category</h3>;
}

function Camera() {
  return <h3>Camera Category</h3>;
}

// App component
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="about" element={<About />} />
          {/* Products parent route */}
          <Route path="products" element={<Products />}>
            {/* Child routes */}
            <Route path="phones" element={<Phones />} />
            <Route path="laptops" element={<Laptops />} />
            <Route path="camera" element={<Camera />} />
          </Route>
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

✅ Result BEFORE `<Outlet />`

URL	Content Rendered
`/products/phones`	Only **Products Page header**
`/products/laptops`	Only **Products Page header**
`/products/camera`	Only **Products Page header**

> Child routes **won't appear** because there is **no `<Outlet />`** in `Products`.

2️⃣ AFTER `<Outlet />` (Child routes render properly)

Here, the `Products` component **includes `<Outlet />`**, so child routes render inside it.

```
// App.jsx
import React from "react";
import { BrowserRouter, Routes, Route, Link, Outlet } from "react-router-dom";

// Layout component
function Layout() {
  return (
    <div>
      <header>
        <h1>My Shop</h1>
        <nav>
          <Link to="/">Home</Link> |{" "}
          <Link to="/products">Products</Link> |{" "}
          <Link to="/about">About</Link>
        </nav>
      </header>

      <main>
        <Outlet /> {/* Child routes render here */}
      </main>

      <footer>
        <p>© 2025 My Shop</p>
      </footer>
    </div>
  );
}

// Pages
function Home() {
  return <h2>Welcome to My Shop!</h2>;
}

function About() {
  return <h2>About Us</h2>;
}

// Products parent page with Outlet
```

```
function Products() {
  return (
    <div>
      <h2>Products Page</h2>
      <nav>
        <Link to="phones">Phones</Link> |{" "}
        <Link to="laptops">Laptops</Link> |{" "}
        <Link to="camera">Camera</Link>
      </nav>

      <Outlet /> {/* ✅ Child categories will render here */}
    </div>
  );
}

// Child pages
function Phones() {
  return <h3>Phones Category</h3>;
}

function Laptops() {
  return <h3>Laptops Category</h3>;
}

function Camera() {
  return <h3>Camera Category</h3>;
}

// App component
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="about" element={<About />} />

          {/* Products parent route */}
          <Route path="products" element={<Products />}>
            {/* Nested child routes */}
            <Route path="phones" element={<Phones />} />
            <Route path="laptops" element={<Laptops />} />
            <Route path="camera" element={<Camera />} />
          </Route>
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

✅ Result AFTER `<Outlet />`

URL	Content Rendered
`/products/phones`	Products Page + **Phones Category**
`/products/laptops`	Products Page + **Laptops Category**
`/products/camera`	Products Page + **Camera Category**

Now the child routes render **inside the Products page** because of `<Outlet />`.

Key Notes for Viewers

1. **Before `<Outlet />`**
    - Parent layout exists, but child routes **won't render**.
    - Useful **only if there are no nested routes**.
2. **After `<Outlet />`**
    - Parent layout + child routes render correctly.
    - **Always use `<Outlet />`** for nested routes.
3. **Nested Routing Structure**

```
/
├─ Home
├─ About
├─ Products
   ├─ Phones
   ├─ Laptops
   └─ Camera
```

4. Use `<Outlet />` in **every parent route** that has nested children.

React Router v7 – Modern Setup with Data Fetching

1️⃣ Introduction

React Router v7 introduces the **Data Router API**, which allows you to:

- Fetch data **before the component renders** using **loaders**.
- Prefetch data **on hover** using `prefetch="intent"`.
- Handle errors at the route level with `errorElement`.
- Organize nested routes cleanly with `<Outlet />`.

This modern approach reduces the need for `useEffect` in components and makes routing more predictable and efficient.

2️⃣ Key Concepts

2.1 Loader

- A **loader** is a function associated with a route.
- It **fetches data before the component renders**.

- Example: fetching a GitHub profile from an API.
- Loader functions are async and can throw errors if fetching fails.

2.2 `useLoaderData`

- Inside the component, we use `useLoaderData()` to access the data returned by the loader.
- This ensures your component always gets the data ready when it renders.

2.3 `<Outlet />`

- `<Outlet />` is a placeholder for **child routes**.
- Any nested route element will render **inside the `<Outlet />`** of the parent component.

2.4 Prefetch on Hover

- Using `prefetch="intent"` on `<Link>` starts loading the data **before the user clicks**, improving perceived performance.

---

3 Folder Structure

```
src/
│
├─ pages/
│   ├─ Home.jsx       # Home page component
│   ├─ About.jsx      # About page component
│   ├─ Products.jsx   # Products page component
│   └─ Profile.jsx    # GitHub Profile page component
│
├─ App.jsx            # Router setup and loader function
└─ index.js           # App entry point
```

4️⃣ Pages Components

4.1 Home.jsx

```
import React from 'react';
import img from "../Images/4.jpg";

const Home = () => {
  return (
    <div className="home-container">
      <div className="home-text">
        <h1>GitHub Profile</h1>
        <p>
          View GitHub user profiles fetched dynamically using <strong>React Router
v7 loaders</strong>.
          This demonstrates preloading data, smooth navigation, and professional UI
design.
        </p>
        <p>
          Explore user information like username, bio, followers, and more in a
clean layout.
        </p>
      </div>

      <div className="home-image">
        <img src={img} alt="GitHub illustration" />
      </div>
    </div>
  );
}

export default Home;
```

4.2 About.jsx

```
import React from "react";

function About() {
  return <h2>About Us</h2>;
}
```

```
export default About;
```

4.3 Products.jsx

```
import React from "react";

function Products() {
  return <h2>Products Page</h2>;
}

export default Products;
```

4.4 Profile.jsx

```
import React from "react";
import { useLoaderData } from "react-router-dom";

function Profile() {
  const user = useLoaderData(); // Access data from loader in App.jsx

  return (
    <div className=”profile-card”>
      <h2>GitHub Profile</h2>
      <img src={user.avatar_url} alt="avatar" width={100} />
      <p><strong>Username:</strong> {user.login}</p>
      <p><strong>Bio:</strong> {user.bio}</p>
      <p><strong>Followers:</strong> {user.followers}</p>
      <p><strong>Following:</strong> {user.following}</p>
      <a href={user.html_url} target="_blank" rel="noreferrer">View on GitHub</a>
    </div>
  );
}

export default Profile;
```

5️⃣ App.jsx (Router Setup and Loader)

```
import React from "react";
import { createBrowserRouter, RouterProvider, Link, Outlet } from
"react-router-dom";
Import “./style.css”;

// Pages
import Home from "./pages/Home";
import About from "./pages/About";
import Products from "./pages/Products";
import Profile from "./pages/Profile";
```

```
// ---------------------------------------------------
// Loader function for GitHub profile
// ---------------------------------------------------
async function githubProfileLoader() {
  const res = await fetch("https://api.github.com/users/sumanmalakar");
  if (!res.ok) throw new Error("Failed to fetch GitHub profile");
  return res.json();
}

// ---------------------------------------------------
// Layout component
// ---------------------------------------------------
function Layout() {
  return (
    <div>
      <header>
        <h1>My Shop</h1>
        <nav>
          <Link to="/">Home</Link> |{" "}
          <Link to="/products">Products</Link> |{" "}
          <Link to="/about">About</Link> |{" "}
          <Link to="/profile" prefetch="intent">Profile</Link>
        </nav>
      </header>
      <main>
        <Outlet /> {/* Child routes render here */}
      </main>

      <footer>
        <p>© 2025 My Shop</p>
      </footer>
    </div>
  );
}
```

```
// ------------------------------------------------------
// Router setup
// ------------------------------------------------------
const router = createBrowserRouter([
  {
    path: "/",
    element: <Layout />,
    children: [
      { index: true, element: <Home /> },
      { path: "about", element: <About /> },
      { path: "products", element: <Products /> },
      { path: "profile", element: <Profile />, loader: githubProfileLoader },
    ],
  },
]);

// ------------------------------------------------------
// App component
// ------------------------------------------------------
function App() {
  return <RouterProvider router={router} />;
}

export default App;
```

6️⃣ index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

7️⃣ How It Works

1. **Navigation**
    - Header links use `<Link>` to navigate.
    - `prefetch="intent"` on Profile link starts loading data **on hover**.
2. **Route-level Data Fetching**
    - `/profile` uses `githubProfileLoader` defined in `App.jsx`.
    - `useLoaderData()` inside `Profile.jsx` gets the data **ready before rendering**.
3. **Layout with `<Outlet />`**
    - All child routes (`/`, `/about`, `/products`, `/profile`) render inside `<Outlet />`.
4. **Folder Organization**
    - Each page has its **own component** for better maintainability.

This setup is **modern, simple, and perfect for teaching React Router v7 in 2025**.