React Router Basics

react-router ile bir React uygulamasına sayfa geçişi eklemek -- BrowserRouter, Routes, Route, Link, NavLink ve tanımsız URL'ler için Not Found sayfası, basit örneklerle.

Orta 12 dk

Son güncelleme: 18 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/react-router-basics

---

React Router Basics

Şimdiye kadar gördüğümüz her örnek TEK bir sayfaydı. Gerçek uygulamalarda ise genellikle birden fazla "sayfa" olur -- bir ana sayfa, bir kurs listesi, bir hakkında sayfası. Bu ders, bunu React'te nasıl yapacağımızı, **React Router** kütüphanesiyle anlatıyor.

Neden Bir Router'a İhtiyacımız Var?

Klasik bir web sitesinde, her sayfa (`/`, `/courses`, `/about`) ayrı bir HTML dosyasıdır -- bir linke tıklayınca tarayıcı YENİ bir sayfa yükler. React uygulamaları ise **tek bir HTML dosyasıyla** (Single Page Application, SPA) çalışır; "sayfa değiştirmek" aslında hep aynı sayfada, URL'e göre FARKLI component'leri render etmek demektir. Bunu yönetmek için **react-router** kütüphanesini kullanıyoruz -- URL'i okuyup hangi component'in gösterileceğine karar veriyor.

BrowserRouter ve Routes ile Sayfa Tanımlamak

Bir React uygulamasını router ile kullanmanın ilk adımı, uygulamayı `BrowserRouter` ile sarmalamak, içine de `Routes` ve `Route`'lar eklemek:

```
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`, tarayıcının URL'ini izleyip React'e haber veren bileşendir. `Routes`, URL'e göre HANGİ `Route`'un eşleştiğine bakar -- her `Route`'un bir `path` (URL kalıbı) ve bir `element` (o URL'de gösterilecek component) değeri vardır. Aynı anda yalnızca eşleşen `Route` render edilir.

Link ile Sayfalar Arası Geçiş

Sayfalar arasında geçiş yapmak için `<a href="...">` KULLANMIYORUZ -- bu, tarayıcıya tüm sayfayı yeniden yüklettirir. Onun yerine react-router'ın `Link` component'ini kullanıyoruz:

```
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">`, ekranda bir `<a>` etiketi gibi görünür ve davranır, ama tıklandığında sayfayı YENİDEN YÜKLEMEZ -- yalnızca URL'i değiştirir, React da bu değişikliğe göre uygun `Route`'u render eder. Bu, geçişi çok daha hızlı ve akıcı yapar.

NavLink ile Aktif Sayfayı Vurgulamak

Bir navigasyon menüsünde, kullanıcının HANGİ sayfada olduğunu görsel olarak belirtmek isteriz -- bunun için `Link` yerine `NavLink` kullanılır:

```
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`, `Link` ile tamamen aynı şekilde çalışır, ama `className`'e (ya da `style`'a) bir FONKSİYON vermene izin verir; bu fonksiyon, `{ isActive }` nesnesini alır ve `isActive` şu an o link'in sayfasında olup olmadığını söyler.

Birden Fazla Sayfayı Bir Arada Kullanmak

Gerçek bir uygulamada genellikle birden fazla sayfa ve ortak bir navigasyon menüsü bir arada bulunur:

```
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>
  );
}
```

Burada üç ayrı `Route` (`/`, `/courses`, `/about`) ve bunlara giden üç `Link` bir arada -- bu, küçük bir çok sayfalı uygulamanın temel iskeleti.

Eşleşmeyen URL'ler: Not Found Sayfası

Kullanıcı tanımlı olmayan bir URL'e giderse (`/olmayan-sayfa` gibi) ne olur? Bunu yakalamak için özel bir `Route` tanımlanır:

```
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="*"`, diğer HİÇBİR `Route` ile eşleşmeyen her URL'i yakalar -- bu yüzden her zaman `Routes` içindeki EN SON `Route` olarak yazılır; React yukarıdan aşağı sırayla eşleşme arar.

Özet ve Terimler Sözlüğü

`BrowserRouter`, uygulamayı sarmalayıp URL'i izler; `Routes` ve `Route`, URL'e göre hangi component'in gösterileceğine karar verir. Sayfalar arası geçiş için `<a>` yerine `Link` (aktif sayfayı vurgulamak gerekiyorsa `NavLink`) kullanılır -- ikisi de sayfayı yeniden yüklemeden URL'i değiştirir. Tanımsız URL'ler, `path="*"` ile yazılan ve `Routes`'un en sonuna eklenen bir `Route` ile yakalanır.

Terimler Sözlüğü

SPA (Single Page Application) — Tek bir HTML dosyasıyla çalışan, "sayfa değişimlerini" tarayıcıyı yeniden yüklemeden JavaScript ile yöneten uygulama türü.

Route — Bir URL kalıbını (`path`) belirli bir component'e (`element`) eşleyen tanım.

Client-Side Routing — URL değişikliklerinin, sunucuya yeni bir istek göndermeden, tarayıcıda JavaScript tarafından yönetilmesi.