Route Parameters & Navigation

URL'in içine değişken değerler koymak (route parametreleri), useParams ile okumak, Outlet ile iç içe route'lar kurmak, ve useNavigate ile programatik yönlendirme -- basit örneklerle.

Orta 14 dk

Son güncelleme: 18 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/route-parameters-navigation

---

Route Parameters & Navigation

React Router Basics dersinde sabit URL'lerle (`/courses` , `/about`) çalıştık. Bu ders, URL'in İÇİNE değişken bir değer koymayı (`/courses/java` gibi) ve bir link'e tıklamadan, kod içinden sayfa değiştirmeyi anlatıyor.

Route Parametreleri: URL'den Veri Okumak

`/courses/java` ve `/courses/react` gibi URL'ler için ayrı ayrı `Route` yazmak yerine, URL'in bir kısmını DEĞİŞKEN olarak tanımlayabiliriz:

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

function CourseList() {
  return (
    <div>
      <h1>Courses</h1>
      <Link to="/courses/java">Java</Link>
      <Link to="/courses/react">React</Link>
    </div>
  );
}

function CourseDetail() {
  // The URL segment defined as ":courseSlug" is read as an object via
  // useParams() -- the key matches the name used in the Route (courseSlug).
  const { courseSlug } = useParams();

  return <h1>Course: {courseSlug}</h1>;
}

function RouteParamExample() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/courses" element={<CourseList />} />
        <Route path="/courses/:courseSlug" element={<CourseDetail />} />
      </Routes>
    </BrowserRouter>
  );
}
```

`path="/courses/:courseSlug"` yazdığımızda, `:courseSlug` kısmı bir **route parametresi** olur -- `/courses/java` URL'inde `courseSlug` `"java"` değerini alır. Component içinde bu değeri `useParams()` hook'u ile okuyoruz; dönen nesnenin anahtarı, Route'taki isimle (`courseSlug`) aynı.

İç İçe (Nested) Route'lar ve Outlet

Bazen bir sayfanın İÇİNDE, URL'e göre değişen ALT bir bölüm olur -- örneğin bir kurs sayfasının içinde, seçilen konuya göre değişen bir içerik alanı:

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

function CourseLayout() {
  // This is the "parent" route matched by /courses/:courseSlug. The
  // <Outlet /> inside it specifies where the matched CHILD route
  // (if any) should be rendered.
  const { courseSlug } = useParams();

  return (
    <div>
      <h1>Course: {courseSlug}</h1>
      <Link to={`/courses/${courseSlug}/enum`}>Enum Topic</Link>
      <Outlet />
    </div>
  );
}

function TopicDetail() {
  const { courseSlug, topicSlug } = useParams();

  return (
    <p>
      Topic: {topicSlug} (course: {courseSlug})
    </p>
  );
}

function NestedRouteExample() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/courses/:courseSlug" element={<CourseLayout />}>
          {/* A nested Route -- it is only rendered at the <Outlet />
              position INSIDE CourseLayout, for a URL like
              /courses/java/enum. */}
          <Route path=":topicSlug" element={<TopicDetail />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

Bir `Route`'u başka bir `Route`'un İÇİNE yazarak (`:topicSlug` , `:courseSlug` route'unun içinde) iç içe (nested) bir yapı kurarız. Üst component'in (`CourseLayout`) içine koyduğumuz `<Outlet />`, eşleşen alt route'un TAM OLARAK nereye render edileceğini belirtir -- `Outlet` olmadan alt route hiçbir yerde görünmez.

useNavigate ile Programatik Yönlendirme

`Link`, kullanıcının TIKLAMASINI gerektirir. Bazen bir işlemin SONUCUNDA (bir butona tıklayınca, bir hesaplama bitince) kod içinden sayfa değiştirmek isteriz:

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

function CourseList() {
  const navigate = useNavigate();

  function handleSelect(courseSlug) {
    // Link always requires a click -- useNavigate(), on the other hand,
    // lets us CHANGE the URL from INSIDE a function (e.g. an event
    // handler).
    navigate(`/courses/${courseSlug}`);
  }

  return (
    <div>
      <button onClick={() => handleSelect("java")}>Go to Java</button>
      <button onClick={() => handleSelect("react")}>Go to React</button>
    </div>
  );
}

function CourseDetail() {
  return <h1>Course Detail</h1>;
}

function UseNavigateExample() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/courses" element={<CourseList />} />
        <Route path="/courses/:courseSlug" element={<CourseDetail />} />
      </Routes>
    </BrowserRouter>
  );
}
```

`useNavigate()` hook'u bize bir `navigate` fonksiyonu verir; bu fonksiyonu bir event handler İÇİNDEN çağırarak URL'i değiştirebiliriz -- `Link`'in aksine, bu bir tıklamaya değil, kod içindeki bir KOŞULA bağlıdır.

Bir İşlemden Sonra Yönlendirmek

`useNavigate`'in en yaygın kullanımlarından biri, bir form gönderildikten sonra kullanıcıyı başka bir sayfaya yönlendirmek:

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

function NewCourseForm() {
  const [title, setTitle] = useState("");
  const navigate = useNavigate();

  function handleSubmit(event) {
    event.preventDefault();

    // A common pattern is to redirect the user to another page AFTER
    // the form is submitted (e.g. "saved").
    navigate("/courses");
  }

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        value={title}
        onChange={(event) => setTitle(event.target.value)}
        placeholder="Course title"
      />
      <button type="submit">Save</button>
    </form>
  );
}

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

function NavigateAfterActionExample() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/new-course" element={<NewCourseForm />} />
        <Route path="/courses" element={<CourseList />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Form Handling dersinde gördüğümüz `onSubmit` + `preventDefault` deseni burada da aynı -- tek fark, form "gönderildikten" (bu örnekte gerçek bir kayıt işlemi olmadığı için doğrudan) sonra `navigate("/courses")` ile kullanıcıyı kurs listesine yönlendiriyoruz.

Geri Gitmek: navigate(-1)

`navigate`'e bir URL yerine bir SAYI da verilebilir -- bu, tarayıcı geçmişinde ileri/geri gitmek için kullanılır:

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

function CourseList() {
  return (
    <div>
      <h1>Courses</h1>
      <Link to="/courses/java">Java</Link>
    </div>
  );
}

function CourseDetail() {
  const navigate = useNavigate();

  function handleBack() {
    // navigate(-1) does the same thing as the browser's "back" button --
    // it goes one step back in the history. navigate(-2) goes two steps
    // back, and so on.
    navigate(-1);
  }

  return (
    <div>
      <h1>Course Detail</h1>
      <button onClick={handleBack}>Back</button>
    </div>
  );
}

function GoBackNavigateExample() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/courses" element={<CourseList />} />
        <Route path="/courses/:courseSlug" element={<CourseDetail />} />
      </Routes>
    </BrowserRouter>
  );
}
```

`navigate(-1)`, tarayıcının "geri" butonuyla aynı işi yapar: history'de bir adım geriye gider. Bu, kullanıcıyı belirli bir sayfaya (`/courses` gibi) sabitlemek yerine, "neredeyse geldiyse oraya" dönmesini sağladığı için genellikle "Back" butonlarında tercih edilir.

Özet ve Terimler Sözlüğü

Bir URL'in bir kısmını `:isim` ile tanımlayarak route parametresi oluşturabilir, değerini `useParams()` ile okuyabiliriz. Bir `Route`'u başka bir `Route`'un içine yazıp üst component'e `<Outlet />` ekleyerek iç içe (nested) route'lar kurabiliriz. `useNavigate()`, bir event handler içinden ya da bir işlemin sonucunda, tıklamaya gerek kalmadan sayfa değiştirmemizi sağlar; `navigate(-1)` ise tarayıcı geçmişinde geriye gitmek için kullanılır.

Terimler Sözlüğü

Route Parametresi — Bir URL kalıbında `:isim` ile tanımlanan, gerçek URL'de değişken bir değer alan kısım.

Nested Route (İç İçe Route) — Başka bir Route'un içine yazılan, yalnızca üst Route eşleştiğinde ve `<Outlet />` konumunda render edilen Route.

Outlet — Üst (parent) route component'inin içinde, eşleşen alt (child) route'un render edileceği yeri belirten component.

Programatik Navigasyon — Bir link'e tıklamak yerine, kod içinden (örneğin bir event handler'dan) `useNavigate()` ile sayfa değiştirmek.

Pratik Proje

Bu kategoride (React Router Basics, Route Parameters & Navigation) öğrendiğimiz kavramları bir arada kullanan, gerçek ve çalıştırılabilir bir örnek proje var: **Routing Demo** -- öğrenme platformunun kendi kurs yapısına benzeyen küçük bir kurs gezinme uygulaması (`/courses` , `/courses/java` , `/courses/java/enum` gibi).

Proje; `BrowserRouter` + `Routes` + `Route` ile sayfa tanımlamayı, `Link` / `NavLink` ile gezinmeyi, route parametreleriyle (`useParams`) URL'den veri okumayı, `Outlet` ile iç içe route'ları ve `useNavigate` ile programatik yönlendirmeyi bir arada gösteriyor. Bilgisayarına indirip çalıştırabilir, kodunu satır satır inceleyebilirsin:

```
git clone https://github.com/cdurgun/react-course-projects.git
cd react-course-projects
npm install
cd projects/routing
npm run dev
```

`react-course-projects` deposu **npm workspaces** kullanır -- `npm install` yalnızca bir kez, depo kökünde çalıştırılır ve tüm proje klasörleri ortak bağımlılıkları paylaşır (her klasörde ayrı bir `node_modules` oluşmaz). Kök dizinde `npm install` 'ı daha önce çalıştırdıysan, doğrudan `cd react-course-projects/projects/routing` yapıp `npm run dev` demen yeterli.