

Portals

react-dom'un createPortal fonksiyonu ile component'leri farklı bir DOM düğümüne render etmek -- modal'lar, event bubbling davranışı, ve portal hedefi ayarlamak, basit örneklerle.

İleri 12 dk

Oluşturulma tarihi: 5 Ekim 2026

Çevrimiçi oku: <https://www.learnforgex.com/tr/topics/portals>

Portals

Advanced React kategorisinin son konusu -- bir component'i, React ağacındaki KONUMUNDAN farklı bir DOM düğümüne render etmenin yolu: **Portal**'lar.

Portal Nedir? createPortal ile Başlangıç

react-dom'un createPortal fonksiyonu, bir component'i DOM'da farklı bir yere render etmemizi sağlar:

```
import { createPortal } from "react-dom";

function BasicPortalExample() {
  // createPortal(child, container) renders `child` into a different DOM
  // node (`container`) INSTEAD OF its place in the normal React tree --
  // here, `document.body` itself. In the component tree (in React
  // DevTools) it still appears INSIDE BasicPortalExample, but in the
  // actual DOM it's in a completely different location.
  return createPortal(<p className="tooltip">I'm rendered directly on body!</p>, document.
  body);
}
```

createPortal(child, container), child'ı normal React ağacındaki YERİNE değil, DOM'daki container düğümüne render eder. Component ağacında (React DevTools'ta) hâlâ olması gereken yerde görünür, ama gerçek DOM'da tamamen farklı bir konumdadır.

Modal'lar için Portal Kullanmak

Portal'ın en yaygın kullanım alanı modal'lar:

```

import { useState } from "react";
import { createPortal } from "react-dom";

function Modal({ onClose, children }) {
  // A modal is the most common use case for a Portal -- the modal's CSS
  // (position: fixed, z-index) makes it appear ON TOP of the rest of the
  // page, but its actual DOM position (e.g. "inside a card") can
  // sometimes block that (e.g. overflow: hidden). The Portal ELIMINATES
  // this problem by rendering the modal directly into `document.body`.
  return createPortal(
    <div className="modal-overlay" onClick={onClose}>
      <div className="modal-content" onClick={(event) => event.stopPropagation()}>
        {children}
        <button onClick={onClose}>Close</button>
      </div>
    </div>,
    document.body,
  );
}

function ModalWithPortalExample() {
  const [isOpen, setIsOpen] = useState(false);

  return (
    <div>
      <button onClick={() => setIsOpen(true)}>Open Modal</button>
      {isOpen && (
        <Modal onClose={() => setIsOpen(false)}>
          <p>This is a modal, rendered outside the normal DOM tree.</p>
        </Modal>
      )}
    </div>
  );
}

```

Bir modal'ın CSS'i (`position: fixed` , yüksek `z-index`) sayfanın geri kalanının ÜSTÜNDE görünmesini sağlamalı -- ama modal'ın gerçek DOM konumu (örneğin `overflow: hidden` olan bir kartın içi) bunu bazen engelleyebilir. Portal, modal'ı doğrudan `document.body` 'ye render ederek bu sorunu ORTADAN KALDIRIR.

Event Bubbling: Portal'ların Şaşırtıcı Davranışı

Portal'ların en önemli (ve en şaşırtıcı) özelliği, event'lerin nasıl davrandığı:

```
import { useState } from "react";
import { createPortal } from "react-dom";

function Popup() {
  return createPortal(<button>Click me (rendered in document.body)</button>, document.body);
}

function EventBubblingThroughPortalExample() {
  const [clicks, setClicks] = useState(0);

  return (
    // IMPORTANT: Popup is rendered in the DOM OUTSIDE this <div> (in
    // document.body). But when its <button> is clicked, onClick still
    // fires HERE (at its real position in the React tree) -- React
    // "bubbles" events according to its OWN component tree, not the
    // actual DOM tree. This is the most surprising yet most useful
    // feature of Portals.
    <div onClick={() => setClicks(clicks + 1)}>
      <p>Clicks: {clicks}</p>
      <Popup />
    </div>
  );
}
```

Popup , DOM'da dıştaki `<div>` 'in DIŞINDA (`document.body` 'de) render ediliyor. Ama içindeki butona tıklandığında, `onClick` yine de dıştaki `<div>` 'e kadar "bubble" ediyor -- React, event'leri GERÇEK DOM ağacına göre değil, KENDİ component ağacına göre yayar. Bu, Portal'ları kullanırken bilmen gereken en önemli davranış.

Portal Hedefini Ayarlamak

`document.body` yerine, genellikle özel olarak ayrılmış bir hedef kullanılır:

```
import { createPortal } from "react-dom";

function Tooltip({ text }) {
  // Instead of document.body, it's more common to use a target set aside
  // specifically in index.html -- for example <div id="tooltip-root"></div>,
  // added as a SIBLING to the app's #root. This makes it easier for the
  // portal content to manage its own styles/position.
  const target = document.getElementById("tooltip-root");

  if (!target) {
    return null;
  }

  return createPortal(<span className="tooltip">{text}</span>, target);
}

function PortalTargetSetupExample() {
  return (
    <div>
      <p>Hover for more info</p>
      <Tooltip text="This tooltip lives in its own DOM node." />
    </div>
  );
}
```

index.html 'de `<div id="tooltip-root"></div>` gibi, uygulamanın `#root` 'una KARDEŞ (sibling) bir eleman eklemek yaygın bir pratiktir -- bu, portal içeriğinin kendi stillerini ve konumunu yönetmesini kolaylaştırır.

Özet ve Terimler Sözlüğü

`createPortal(child, container)`, bir component'i React ağacındaki konumunda TUTARAK, gerçek DOM'da farklı bir düğüme render eder -- en yaygın kullanımı modal'lar, tooltip'ler ve dropdown'lardır (üst öğelerin `overflow: hidden` gibi CSS özelliklerinden kaçınmak için). Event'ler, gerçek DOM konumuna değil, React'in component ağacına göre bubble eder -- bu, Portal'ları normal component'ler gibi kullanmaya devam edebilmemizi sağlar.

Terimler Sözlüğü

Portal — Bir component'i, React ağacındaki konumunu koruyarak, DOM'da farklı bir düğüme render etme mekanizması.

Event Bubbling (Olay Yayılımı) — Bir event'in, tetiklendiği elemandan başlayıp ağaçtaki üst elemanlara doğru yayılması.

Pratik Proje

Bu kategoride (React Performance, Error Boundaries, Lazy Loading & Code Splitting, Suspense, Portals) öğrendiğimiz kavramları bir arada kullanan, gerçek ve çalıştırılabilir bir örnek proje var: [Advanced React Demo](#) -- `React.memo` ile optimize edilmiş bir kurs listesi,

bir Error Boundary, `React.lazy` + `Suspense` ile code splitting yapılan bir detay paneli, ve bir Portal modal'ı bir arada gösteren bir uygulama.

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/advanced-react
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/advanced-react` yapıp `npm run dev` demen yeterli.