Component Composition

children prop'u, iç içe component'ler ve composition ile inheritance arasındaki fark -- yeniden kullanılabilir bir Card component'i mini projesiyle.

Başlangıç 12 dk

Son güncelleme: 18 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/component-composition

---

Component Composition

Components ve Props derslerinde küçük component'ler yazıp onlara veri gönderdik. Bu ders, o küçük parçaları nasıl BİRLEŞTİRİP daha büyük arayüzler kurduğumuzu gösteriyor -- buna "composition" denir.

children Prop'u Nedir?

`children`, her component'in otomatik olarak sahip olduğu özel bir prop. Bir component'in açılış ve kapanış etiketi arasına ne yazarsan, o içerik `children` olarak o component'e ulaşır:

```
// "children" is a special prop that every component automatically has.
// Whatever you write between a component's OPENING and CLOSING tags becomes
// that content, reaching the component as props.children.
function Box({ children }) {
  return <div className="box">{children}</div>;
}

function App() {
  return (
    <Box>
      <p>This text is Box's children.</p>
    </Box>
  );
}

console.log(App());
```

`<Box><p>...</p></Box>` yazdığımızda, `<p>...</p>` kısmı `Box`'un `children`'ı olur. Bu, Props dersinde gördüğümüz normal prop'lardan biraz farklı çalışır -- `children`'ı bir attribute gibi değil, etiketin İÇİNE yazarsın.

İç İçe Component'ler (Nested Components)

Component'ler iç içe kullanılabilir -- bir component başka component'ler içerebilir, onlar da kendi içlerinde başka component'ler içerebilir:

```
// Components can be nested -- a component can contain other components,
// which can in turn contain their own components. This is how a large UI
// is built up from small pieces.
function Avatar() {
  return <img src="/avatar.png" alt="Profile picture" />;
}

function UserName() {
  return <span>Jane Smith</span>;
}

function UserProfile() {
  return (
    <div>
      <Avatar />
      <UserName />
    </div>
  );
}

function App() {
  return <UserProfile />;
}

console.log(App());
```

`UserProfile`, `Avatar` ve `UserName`'i içeriyor; `App` de `UserProfile`'ı içeriyor. Küçük, tek işi olan component'lerden büyük bir arayüz kurmanın yolu bu -- her component kendi işine odaklanır.

Composition vs Inheritance (Kısa Bakış)

Nesne yönelimli programlamada "inheritance" (kalıtım), bir sınıfın başka bir sınıftan özellik devralmasıdır. React'te component'ler arasında böyle bir kalıtım YOKTUR:

```
// "Inheritance" is when, in Object-Oriented Programming, a class inherits
// properties from another class. React components have NO such inheritance --
// instead, "composition" is used: you combine small components by placing
// them INSIDE a larger component.

// Composition -- the way this project (and React) does it:
function Card({ children }) {
  return <div className="card">{children}</div>;
}

function ProfileCard() {
  return (
    <Card>
      <h3>Jane Smith</h3>
      <p>Frontend Developer</p>
    </Card>
  );
}

// You would NOT write inheritance like "class ProfileCard extends Card { ... }"
// in React -- the React team says composition is sufficient and simpler for
// nearly every scenario.

console.log(ProfileCard());
```

React'te component'ler, kalıtımla değil, composition ile birleştirilir -- küçük component'leri, `children` ile büyük bir component'in içine koyarsın. React ekibi, composition'ın neredeyse her senaryo için yeterli ve daha basit olduğunu söylüyor -- bu yüzden React'te `extends` ile bir component'i genişletmek gibi bir kalıp görmezsin.

Özet ve Terimler Sözlüğü

`children`, bir component'in açılış/kapanış etiketleri arasına yazılan içeriği taşıyan özel bir prop. Component'ler iç içe kullanılarak büyük arayüzler kurulur. React'te component'leri birleştirmenin yolu composition'dır, kalıtım (inheritance) değil.

Terimler Sözlüğü

`children` — Bir component'in açılış ve kapanış etiketleri arasına yazılan içeriği taşıyan özel prop.

Composition (Birleştirme) — Küçük component'leri bir araya getirerek daha büyük bir arayüz kurma yöntemi.

Inheritance (Kalıtım) — Bir sınıfın başka bir sınıftan özellik devraldığı, React'in component'ler arasında KULLANMADIĞI bir yöntem.

Pratik Proje

Bu kategoride (Components, Props, Component Composition) öğrendiğimiz kavramları bir arada kullanan, gerçek ve çalıştırılabilir bir örnek proje var: **Components & Props Demo**.

Proje; birden çok component'i, props'u (destructuring ve default değerlerle) ve `children` ile composition'ı 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/components-props
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/components-props` yapıp `npm run dev` demen yeterli.