Props

Props nedir, parent'tan child'a veri gönderme, birden fazla prop, destructuring, varsayılan değerler ve props'un normal fonksiyon parametrelerinden farkı.

Başlangıç 10 dk

Son güncelleme: 6 Ekim 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/props

Props

Components dersinde `Button` component'ini üç kez kullanmıştık, ama üçü de birbirinin aynısıydı. Peki her kullanımda farklı bir metin, farklı bir renk göstermek istersek? Bunun cevabı **props**. Bu ders, bir component'e dışarıdan veri göndermenin yolunu basit örneklerle anlatıyor.

Props Nedir?

Props (properties), bir component'e dışarıdan veri göndermenin yoludur -- tıpkı bir HTML etiketine attribute vermek gibi (JSX dersindeki "Attribute'lar ve className" bölümünü hatırla), ama burada değer, component fonksiyonuna bir parametre olarak ulaşır.

Parent'tan Child'a Veri Göndermek

Bir component'i kullanırken (yani onu "render" ederken), ona attribute gibi değerler verebilirsin -- bunlar, o component'in props'u olur:

```
// Props (properties) are a way to pass data INTO a component from the
// outside -- just like giving an HTML tag an attribute, except here the
// value reaches your component function as a parameter.
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}

function App() {
  return <Greeting name="Emma" />;
}

console.log(App());
// Result: <h1>Hello, Emma!</h1>
```

Burada `App`, `Greeting`'i kullanan (yani "parent") component; `Greeting` ise kullanılan (yani "child") component. `name="Ayşe"` yazarak, `Greeting`'e `name` adında bir prop gönderiyoruz.

Birden Fazla Prop Kullanmak

Bir component'e istediğin kadar prop gönderebilirsin:

```
// You can pass as many props as you like to a component -- each one
// reaches the function as a separate field of the props object.
function UserCard(props) {
  return (
    <div>
      <h2>{props.name}</h2>
      <p>Age: {props.age}</p>
      <p>City: {props.city}</p>
    </div>
  );
}

function App() {
  return <UserCard name="Emma" age={28} city="Boston" />;
}

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

Her prop, `props` nesnesinin ayrı bir alanı olarak component'e ulaşır -- `props.name`, `props.age`, `props.city` gibi.

Destructuring ile Props Okumak

Her yerde `props.name` yazmak yerine, çoğu React kodunda destructuring kullanılır:

```
// Instead of writing "props.name", "props.age" everywhere, most React code
// uses destructuring -- inside the function parameter, you pull the fields
// you need out directly as variables. Both examples do the SAME thing,
// the second one is just shorter and the more commonly used style.

// 1) WITHOUT destructuring:
function UserCardLong(props) {
  return <p>{props.name} -- {props.age} years old</p>;
}

// 2) WITH destructuring (the commonly used style):
function UserCard({ name, age }) {
  return <p>{name} -- {age} years old</p>;
}

console.log(UserCardLong({ name: "Emma", age: 28 }));
console.log(UserCard({ name: "Emma", age: 28 }));
```

İki yazım da tamamen aynı işi yapar -- destructuring, yalnızca `props.` tekrarını ortadan kaldırıp kodu biraz daha kısa yapar. Yaygın kullanılan biçim bu -- ilerleyen derslerde de bunu kullanacağız.

Varsayılan Değerler (Default Props)

Bir prop hiç gönderilmezse ne olur? Destructuring sırasında bir varsayılan değer tanımlayabilirsin:

```
// What should happen if a prop is never passed at all? You can define a
// "default value" during destructuring -- exactly the same idea as default
// values for regular JavaScript function parameters.
function Greeting({ name = "Guest" }) {
  return <h1>Hello, {name}!</h1>;
}

console.log(Greeting({ name: "Emma" })); // Hello, Emma!
console.log(Greeting({}));               // Hello, Guest! (default was used)
```

`name = "Misafir"`, normal JavaScript fonksiyon parametrelerindeki varsayılan değerlerle tamamen aynı fikir -- `name` gönderilmezse, `Misafir` kullanılır.

Props vs Normal Fonksiyon Parametreleri

Props, aslında sıradan bir fonksiyon parametresinden fazlası değil -- React'in özel bir mekanizması yok burada. Fark yalnızca kullanım şeklinde: normal bir fonksiyonu `Greeting({ name: "Ayşe" })` diye çağırırken, bir component'i JSX içinde `<Greeting name="Ayşe" />` diye "çağırırsın". Ayrıca önemli bir kural var: props'lar **salt okunurdur** (read-only) -- bir component, kendisine gelen bir prop'u asla değiştirmemelidir. Veriyi değiştirmek istiyorsan, bunun yolu "state" -- onu State ve Event'ler kategorisinde göreceğiz.

Özet ve Terimler Sözlüğü

Props, bir component'e dışarıdan veri göndermenin yoludur. Parent component, child'a attribute gibi değer verir; child bunu `props` (ya da destructuring ile doğrudan değişken olarak) okur. Bir prop gönderilmezse, varsayılan bir değer tanımlanabilir. Props salt okunurdur.

Terimler Sözlüğü

Props — Bir component'e dışarıdan gönderilen, salt okunur veriler.

Parent Component — Başka bir component'i kullanan (render eden) component.

Child Component — Bir parent component tarafından kullanılan component.

Destructuring — Bir nesnenin alanlarını doğrudan değişken olarak çıkarma söz dizimi (`{ name, age }` gibi).

Default Prop (Varsayılan Değer) — Bir prop gönderilmediğinde kullanılacak, önceden tanımlanmış değer.