What Are Hooks?

Hook nedir, neden var, Hooks kuralları (Rules of Hooks) ve function component'lerle ilişkisi -- basit örneklerle.

Orta 10 dk

Son güncelleme: 6 Ekim 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/what-are-hooks

---

What Are Hooks?

State ve Event'ler kategorisinde `useState`'i defalarca kullandık, ama hiç "hook nedir" diye durup bakmadık. Bu ders, hook kavramını baştan anlatıyor -- ve bu noktadan sonra React kursunda konular biraz daha derinleşecek.

Hook Nedir?

Bir **hook**, adı `use` ile başlayan ve function component'lere React özellikleri (state, side effect'ler, DOM referansları gibi) eklemeni sağlayan bir fonksiyondur:

```
import { useState } from "react";

// useState is a HOOK that React provides out of the box -- a function that
// starts with "use" and adds React features like state to function
// components.
function WhatIsAHookExample() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>useState is a hook: {count}</p>
      <button onClick={() => setCount(count + 1)}>+1</button>
    </div>
  );
}
```

`useState`, aslında React'in sana hazır verdiği bir hook. Bu kategoride göreceğimiz `useEffect`, `useRef`, `useMemo`, `useCallback` de birer hook -- hepsi aynı `use...` isimlendirme kalıbını izliyor.

Neden Hooks?

Hooks'tan önce, state ve diğer React özellikleri yalnızca "class component" denen, farklı bir söz dizimiyle yazılan component'lerde kullanılabiliyordu. Hooks, bu özellikleri normal function component'lere de taşıdı -- artık class component yazmaya hiç gerek yok. Bu kursta zaten baştan beri yalnızca function component kullandık; hooks, bunu mümkün kılan mekanizma.

Hooks Kuralları (Rules of Hooks)

Hook'ları kullanırken uyman gereken iki temel kural var:

```
import { useState } from "react";

function RulesOfHooksExample({ showExtra }) {
  // CORRECT: Hooks are ALWAYS called at the TOP level of the component.
  const [count, setCount] = useState(0);

  // WRONG: Calling a hook conditionally (never do this).
  // if (showExtra) {
  //   const [extra, setExtra] = useState(0); // Breaks the rules!
  // }
  //
  // Why? React tracks each hook by assuming they're called in the SAME
  // ORDER on EVERY RENDER. A conditional hook breaks this order by being
  // called on some renders and not others -- React gets confused about
  // which state belongs to which hook.

  return <p>Count: {count}</p>;
}
```

Birincisi: hook'lar her zaman component'in EN ÜST seviyesinde çağrılır -- bir `if`, `for` ya da başka bir fonksiyonun içine YERLEŞTİRİLMEZ. İkincisi: hook'lar yalnızca function component'lerin içinden ya da başka bir hook'un içinden çağrılır, normal bir fonksiyondan çağrılmaz. React, hook'ların her render'da aynı sırada çağrıldığını varsayarak state'i doğru component'e eşliyor -- bu kurallar bozulursa React karışır.

Function Component'ler ve Hooks

Hook'lar yalnızca function component'lerin içinde çalışır -- normal bir JavaScript fonksiyonunun (component olmayan) içinde `useState` çağırmak hataya yol açar. Bu yüzden bu kursta gördüğümüz her component (`App`, `Card`, `ProfileInfo` gibi) bir function component, ve hepsi hook kullanmaya uygun.

Özet ve Terimler Sözlüğü

Hook, `use` ile başlayan ve function component'lere React özellikleri ekleyen bir fonksiyondur. Hook'lar yalnızca component'in en üst seviyesinde ve yalnızca function component'ler (ya da başka hook'lar) içinden çağrılır -- koşullu ya da döngü içinde çağrılamaz.

Terimler Sözlüğü

Hook — `use` ile başlayan, function component'lere React özellikleri ekleyen fonksiyon.

Rules of Hooks (Hooks Kuralları) — Hook'ların yalnızca component'in en üst seviyesinde ve yalnızca function component/hook içinden çağrılabileceğini belirten iki kural.

Function Component — JavaScript fonksiyonu olarak yazılan React component'i; hook'lar yalnızca bunların içinde çalışır.