State

State nedir, useState ile nasıl tanımlanır, nasıl güncellenir, bir önceki state'e göre güncelleme ve state immutability -- basit örneklerle.

Başlangıç 12 dk

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

State

Events dersinde kullanıcının eylemlerini yakalamayı öğrendik, ama handler'larımız şimdiye kadar yalnızca `console.log` yazdı -- ekranda hiçbir şey değişmedi. Bu ders, React'te ekranı GERÇEKTEN değiştirmenin yolunu anlatıyor: **state**.

State Nedir?

State, bir component'in "hatırladığı" ve zamanla değişebilen veridir. State değiştiğinde, React o component'i otomatik olarak yeniden render eder -- yani ekranı güncel state'e göre yeniden çizer. Bir sayaç, bir input'un o anki değeri, bir menünün açık mı kapalı mı olduğu -- bunların hepsi state'e örnek.

useState ile Bir State Tanımlamak

State tanımlamak için `useState` fonksiyonunu kullanırsın:

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

function UseStateBasicExample() {
  // useState(0) sets up state with an initial value of 0.
  // count: to read the current value. setCount: to update the value.
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

`useState(0)`, state'i `0` başlangıç değeriyle kurar ve iki şey döndürür: mevcut değeri (`count`) ve onu güncellemek için bir fonksiyon (`setCount`). `const [count, setCount] = ...` yazımı, bu iki değeri tek satırda ayrı değişkenlere ayıran bir JavaScript özelliği (array destructuring) -- Props dersindeki nesne destructuring'ine benzer, ama dizi için.

State Güncellemek

State'i değiştirmek için doğrudan `count = count + 1` **yazamazsın** -- bunun yerine `useState` 'in verdiği fonksiyonu (`setCount`) çağırman gerekir:

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

function UpdatingStateExample() {
  const [count, setCount] = useState(0);

  function increment() {
    setCount(count + 1);
  }

  function reset() {
    setCount(0);
  }

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>+1</button>
      <button onClick={reset}>Reset</button>
    </div>
  );
}
```

`setCount(...)` çağrıldığında React iki şey yapar: state'in yeni değerini kaydeder, ve component'i o yeni değerle yeniden render eder. Doğrudan bir değişkene atama yapmak React'e "bir şey değişti" demediği için ekranı güncellemez -- bunu State vs Normal Değişken bölümünde göreceğiz.

Bir Önceki State'e Göre Güncelleme

Yeni state'i hesaplarken bir önceki state'e ihtiyacın varsa, `setCount` 'a doğrudan bir değer yerine bir FONKSİYON vermelisin:

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

function PreviousStateExample() {
  const [count, setCount] = useState(0);

  function incrementTwice() {
    // WRONG: count still shows the OLD value here, and both lines use
    // the same "old value + 1" -- so the result only increases by 1,
    // not 2.
    // setCount(count + 1);
    // setCount(count + 1);

    // RIGHT: the function form (prevCount => ...) always uses the most
    // up-to-date value from the previous update.
    setCount((prevCount) => prevCount + 1);
    setCount((prevCount) => prevCount + 1);
  }

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={incrementTwice}>Increment by 2</button>
    </div>
  );
}
```

`setCount(count + 1)` yazmak, aynı render içinde birden fazla kez çağrılırsa güvenilir değildir -- `count` değişkeni, o render sırasında sabit kalır. `setCount((prevCount) => prevCount + 1)` yazmak ise React'e "her ne olursa olsun, EN GÜNCEL değere göre hesapla" der.

State vs Normal Değişken

Neden `let` ile normal bir değişken kullanmıyoruz? Çünkü normal bir değişkeni değiştirmek ekranı güncellemez:

```
function StateVsVariableExample() {
  let plainCount = 0; // A plain variable -- NOT state.

  function handleClick() {
    plainCount = plainCount + 1;
    console.log("plainCount is now:", plainCount);
    // The value actually changed, but nothing updates on screen --
    // because React has no idea a plain variable changed.
  }

  return (
    <div>
      <p>The value on screen always stays the same: {plainCount}</p>
      <button onClick={handleClick}>Increment (Screen Won't Change)</button>
    </div>
  );
}
```

`plainCount` gerçekten değişiyor (console'da görebilirsin), ama React bunu bilmediği için component'i yeniden render etmiyor -- ekranda hep aynı değer kalıyor. `useState` özel çünkü React'e "bu değer değiştiğinde beni haberdar et" demiş oluyorsun.

State Immutability (Değişmezlik)

State bir nesne ya da diziyse, onu doğrudan değiştirmek (mutate etmek) yerine her zaman YENİ bir nesne/dizi oluşturup `set` fonksiyonuna vermelisin:

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

function StateImmutabilityExample() {
  const [user, setUser] = useState({ name: "Emma", age: 25 });

  function haveBirthday() {
    // WRONG: mutating the existing object directly.
    // React might not notice the change, and the screen might not update.
    // user.age = user.age + 1;
    // setUser(user);

    // RIGHT: create a NEW object that copies the old values via spread
    // (...) and pass that to the state.
    setUser({ ...user, age: user.age + 1 });
  }

  return (
    <div>
      <p>
        {user.name} is {user.age} years old.
      </p>
      <button onClick={haveBirthday}>Birthday</button>
    </div>
  );
}
```

`user.age = user.age + 1` yazıp aynı nesneyi geri vermek, React'in değişikliği fark etmesini garanti etmez -- React, state'in "aynı nesne mi farklı bir nesne mi" olduğuna bakarak yeniden render edip etmeyeceğine karar verir. Spread operatörü (`{ ...user, age: ... }`), eski nesnenin tüm alanlarını kopyalayıp yalnızca belirttiğin alanı değiştiren yeni bir nesne oluşturur.

Özet ve Terimler Sözlüğü

State, bir component'in zamanla değişebilen ve değiştiğinde ekranı otomatik güncelleyen verisidir. `useState` ile tanımlanır, yalnızca `set` fonksiyonuyla güncellenir -- doğrudan atama işe yaramaz. Bir önceki state'e göre güncelleme yaparken fonksiyon formu kullanılır. Nesne /dizi state'ler asla doğrudan değiştirilmez (mutate edilmez), her zaman yeni bir kopya oluşturulur.

Terimler Sözlüğü

State — Bir component'in zamanla değişebilen, değiştiğinde component'in yeniden render edilmesini tetikleyen verisi.

`useState` — Bir component'e state eklemeni sağlayan React fonksiyonu; mevcut değeri ve onu güncelleyen bir fonksiyonu döndürür.

Re-render (Yeniden Render) — React'in, state ya da props değiştiğinde component'i yeniden çalıştırıp ekranı güncellemesi.

Immutability (Değişmezlik) — Mevcut bir nesneyi/diziyi doğrudan değiştirmek yerine, değişikliği içeren yeni bir kopya oluşturma ilkesi.