Lists & Keys

map() ile liste render etmek, key prop'u nedir, neden önemlidir, ve en yaygın hata olan index'i key olarak kullanmak -- basit örneklerle.

Başlangıç 10 dk

Son güncelleme: 18 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/lists-and-keys

---

Lists & Keys

Şimdiye kadarki örneklerde hep tek bir şeyi gösterdik -- tek bir sayaç, tek bir kullanıcı. Ama gerçek uygulamalarda çoğu zaman bir LİSTE gösterirsin: bir ürün listesi, bir yorum listesi, bir görev listesi. Bu ders, React'te listeleri render etmeyi ve `key` prop'unun neden önemli olduğunu anlatıyor.

map() ile Liste Render Etmek

Bir diziyi ekrana JSX elementleri olarak dökmek için JavaScript'in `map()` fonksiyonunu kullanırsın:

```
function MapRenderListExample() {
  const fruits = ["Apple", "Pear", "Banana"];

  // map() converts each item in an array into a JSX element; the result
  // is again an array (this time made up of JSX elements).
  return (
    <ul>
      {fruits.map((fruit) => (
        <li key={fruit}>{fruit}</li>
      ))}
    </ul>
  );
}
```

`map()`, bir dizideki her elemanı BAŞKA bir şeye çevirip yeni bir dizi döndürür -- burada her meyve string'ini bir `<li>` elementine çeviriyoruz. Sonuç, JSX elementlerinden oluşan bir dizi; React bu diziyi doğrudan render edebilir.

key Prop'u Nedir?

`map()` ile bir liste render ederken, her elemana bir `key` prop'u vermen gerekir:

```
function KeyPropExample() {
  const users = [
    { id: 1, name: "Alice" },
    { id: 2, name: "Bob" },
    { id: 3, name: "Clara" },
  ];

  // The key should, whenever possible, be a PERMANENT and UNIQUE identifier
  // specific to that item -- not its name or index -- such as an id
  // coming from the database, as here.
  return (
    <ul>
      {users.map((user) => (
        <li key={user.id}>{user.name}</li>
      ))}
    </ul>
  );
}
```

`key`, o listedeki elemanı BENZERSİZ şekilde tanımlayan bir string ya da sayı. Mümkünse elemanın ismi ya da index'i değil, veritabanından gelen `id` gibi kalıcı bir kimlik kullanılmalı.

key Neden Önemli?

`key`, React'e bir sonraki render'da hangi liste elemanının HANGİSİYLE aynı olduğunu söyler:

```
function WhyKeysMatterExample() {
  const tasks = [
    { id: 1, text: "Buy milk" },
    { id: 2, text: "Buy bread" },
  ];

  // The key tells React "which item from the previous render this list
  // item is the same as". When an item is added/removed or the order
  // changes, React uses this to update the correct item -- without a key
  // (or with the wrong key), React might update the wrong item, causing
  // unexpected visual bugs (e.g. an input shifting to the wrong row).
  return (
    <ul>
      {tasks.map((task) => (
        <li key={task.id}>{task.text}</li>
      ))}
    </ul>
  );
}
```

Liste değişmeden kaldığı sürece `key` fark etmez gibi görünür. Ama bir eleman eklenip /silindiğinde ya da sıralama değiştiğinde, React `key`'i kullanarak hangi elemanın "aynı kaldığını", hangisinin yeni olduğunu anlar -- bu sayede yalnızca gerçekten değişen kısmı günceller, listenin tamamını yeniden oluşturmaz.

Yaygın Hatalar

En yaygın hata, `key` olarak dizinin index'ini kullanmaktır:

```
function CommonKeyMistakeExample() {
  const items = ["Apple", "Pear", "Banana"];

  return (
    <ul>
      {/* COMMON MISTAKE: using the index as the key. It appears to work
          as long as the list never changes, but when an item is
          added/removed or the order changes, every item's index shifts --
          React may now think the wrong item is "the same item". */}
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}
```

Liste hiç değişmiyorsa bu çalışıyor gibi görünür. Ama bir eleman eklendiğinde/silindiğinde ya da sıralama değiştiğinde her elemanın index'i kayar -- React, index'e bakarak "aynı eleman" sandığı şeyin aslında farklı bir eleman olduğunu anlayamaz. Bu, özellikle input içeren listelerde (bir görev listesindeki checkbox'lar gibi) yanlış elemanın güncellenmesi gibi görsel hatalara yol açabilir. Mümkün olduğunca `id` gibi kalıcı bir kimlik kullan; elinde gerçekten hiç kalıcı bir kimlik yoksa ve liste asla yeniden sıralanmıyor/değişmiyorsa, index bir son çare olabilir.

Özet ve Terimler Sözlüğü

Bir diziyi ekrana dökmek için `map()` kullanılır; her elemana, React'in listedeki değişiklikleri doğru takip edebilmesi için benzersiz bir `key` verilir. `key` olarak mümkünse kalıcı bir `id` kullanılmalı, dizinin index'i değil -- index kullanmak, liste değiştiğinde yanlış elemanların güncellenmesine yol açabilir.

Terimler Sözlüğü

`map()` — Bir dizideki her elemanı başka bir şeye çevirip yeni bir dizi döndüren JavaScript dizi metodu; React'te listeleri JSX elementlerine çevirmek için kullanılır.

`key` — Bir listedeki her elemana verilen, React'in render'lar arasında elemanları takip etmesini sağlayan benzersiz kimlik.

Reconciliation (Uzlaştırma) — React'in, bir önceki render ile yeni render arasındaki farkı bulup yalnızca gerçekten değişen kısımları güncellediği süreç; `key`, bu süreçte listelerin doğru eşleştirilmesini sağlar.

Pratik Proje

Bu kategoride (Events, State, Conditional Rendering, Lists & Keys) öğrendiğimiz kavramları bir arada kullanan, gerçek ve çalıştırılabilir bir örnek proje var: **State & Events Demo** -- basit bir görev listesi (task list) uygulaması.

Proje; event handler'ları (`onClick` / `onChange` / `onSubmit`), `useState`'i, önceki state'e göre güncellemeyi, state immutability'yi, ternary/`&&` ile conditional rendering'i ve `map()` + `key` ile liste render etmeyi 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/state-events
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/state-events` yapıp `npm run dev` demen yeterli.