DİZİ METOTLARI İLE İTERASYON

DİZİLERDE İTERASYON

- JS'de bir dizi içerisinde iterasyon yapmak için çok farklı yollar kullanılabilir.

- **Döngüler ile**
  - Klasik **for** döngüsü
  - **for in** döngüsü
  - **for of** döngüsü

- **Dizi iterasyon metotları ile (En çok kullanılanlar)**
  - Array.forEach()
  - Array.map()
  - Array.filter()
  - Array.reduce()

FOREACH METODU İLE DİZİ İTERASYONU

- **Array.forEach()** bir döngü deyimi değil bir dizi **İTERASYON** metodudur.

- Bu metot, bir fonksiyonu parametre olarak alır ve bu fonksiyona göre bir belirtilen dizi üzerinde **iterasyon** yapılabilir.

- Avantajı kullanımı kolaydır. Dezavantajı ise döngüyü kırmak ve atlamak mümkün değildir.

- Ayrıca **forEach** metodu orijinal diziyi **değiştirmez**.

FOREACH METODU

- **forEach()** metodu içerinde çağrılan ya da tanımlanan callback aslında **3 adet parametre** alabilmektedir.

```
Array.forEach ( function(şuankiDeğer, indis, dizi))
```

- **şuankiDeğer:** Seçilen dizi elemanının mevcut değerini göstermektedir. Kullanımı **ZORUNLUDUR**.
- **İndis:** Şu anki dizi elemanının sırasını (index) gösterir. **OPSİYONEL.**
- **Dizi:** Şu an ki elemanın ait olduğu dizi nesnesidir. **OPSİYONEL.**

NOT: Bu parametrelerin isimlerini kullanıcı belirler ancak sırası önemlidir. 1. parametre değer, 2. si index, 3.sü ise dizidir.

FOREACH METODU İLE DİZİ İTERASYONU

ÖRNEK: Bir dizideki elemanları her birini ayrı ayrı yazdıran uygulamayı **forEach** metodu ile yazınız.

FOREACH METODU İLE DİZİ İTERASYONU

ÖRNEK: Bir dizideki elemanların toplamını bularak sonucu konsola yazdıran uygulamayı FOREACH metodu ile yazınız.

FOREACH METODUNDA İNDİS KULLANIMI

ÖRNEK: Belirtilen dizinin her bir elamanının 5 katını alarak ayrı bir dizide saklayan uygulamayı **forEach()** metodu ile yazınız.

MAP() METODU

- **Array.map()** metodu, bir fonksiyonu parametre olarak alır ve orijinal dizinin kopyasını bu fonksiyona göre modifiye ederek döndürür.

- Yani bir diziyi **transformasyondan** geçirmek için **map()** metodu kullanılabilir.

- **map()** metodu orijinal diziyi **değiştirmez**. Yeni bir dizi oluşturarak döndürür.

MAP() METODU

ÖRNEK: Bir dizideki elemanların **5 katını** alarak yeni bir diziye kaydeden uygulamayı **map()** metodu ile yazınız.

```
const rakamlar = [3, 7, 17, 8, 9, 3, 0];
```

```
const katAlınmış = rakamlar.map((x) => x * 5);
console.log(katAlınmış);
```

- Bu uygulamayı **forEach** ile de yapmıştık ancak **map()** kullanmak çok daha basit.
- **map()** metodu, güncellenmiş diziyi doğrudan bir değişkene atmaya izin vermektedir.

MAP() METODU

ÖRNEK: Bir dizideki tüm isimleri **BÜYÜK** harfe dönüştüren uygulamayı yazınız.

```
const isimler = ["Mustafa", "Murat", "Ahmet", "Mustafa", "Ayşe", "canan"];
```

ÖRNEK: tlFiyatlar dizisindeki fiyatların Euro ve dolar karşılıklarını hesaplatarak yeni dizlere kaydediniz.

```
const euro = 9.68;
const dolar = 8.1;
const tlFiyatlar = [100, 150, 100, 50, 80];
```

MAP() METODUNDA İNDİS KULLANIMI

- **map()** metodu forEach() gibi **3 adet parametre** alabilmektedir.

```
Array.map( function(şuankiDeğer, indis, dizi))
```

- **şuankiDeğer:** Seçilen dizi elemanının mevcut değerini göstermektedir. Kullanımı **ZORUNLUDUR**.
- **İndis:** Şu anki dizi elemanının sırasını (index) gösterir. **OPSİYONEL.**
- **Dizi:** Şu an ki elemanın ait olduğu dizi nesnesidir. **OPSİYONEL.**

NOT: Bu parametrelerin isimlerini kullanıcı belirler ancak sırası önemlidir. 1. parametre değer, 2. si index, 3.sü ise dizidir.

MAP() METODU

ÖRNEK: tlFiyatlar dizidekisindeki ürünlere zam yapılmak isteniyor. Fiyatı 100 TL den fazla olanlara %10 zam, 100 TL den az olanlara ise %15 zam yapılmak isteniyor. Ayrıca, zamlı olan yeni değerleri örnekteki gibi diziye saklamak istiyoruz.

1.Ürün Zamlı Fiyatı:110

FILTER() METODU

- **Array.filter()** metodu, bir fonksiyonu parametre olarak alır ve orijinal dizinin kopyasını bu fonksiyona göre **filtreleyerek** döndürür.
- Yani bir dizideki istediğimiz elemanların seçmek için kullanılır.
- **filter()** metodu orijinal diziyi **değiştirmez**. Yeni bir dizi oluşturarak döndürür.

ÖRNEK: **Koordinatlar** dizisindeki negatif koordinatları alıp yeni bir diziye saklayan uygulamayı **filter()** ile yapınız.

```
const koordinatlar = [-100, 150, -32, 43, -20]
```

REDUCE() METODU

- **Array.reduce()** metodu, bir fonksiyonu parametre olarak alır ve orijinal diziyi bu fonksiyona göre işleyerek **tek bir değer** döndürür.

- Örneğin bir dizinin değerlerinin toplamını bulmak için **reduce()** metodu kullanılabilir.

- **reduce()** metodu orijinal diziyi **değiştirmez**. Sadece bir **değer** döndürür.

REDUCE() METODUNDA İNDİS KULLANIMI

- **reduce()** metodu içerisindeki fonksiyon **4 adet parametre** alabilmektedir.

```
Array.reduce( function(toplam, şuankiDeğer, indis, dizi))
```

  - **toplam:** Her iterasyonda ardışık olarak yapılan işlemlerin kümülatif toplamını gösterir. **ZORUNLUDUR**.
  - **şuankiDeğer:** Seçilen dizi elemanının mevcut değerini göstermektedir. Kullanımı **ZORUNLUDUR**.
  - **İndis:** Şu anki dizi elemanının sırasını (index) gösterir. **OPSİYONEL.**
  - **Dizi:** Şu an ki elemanın ait olduğu dizi nesnesidir. **OPSİYONEL.**

REDUCE() METODU

ÖRNEK: Koordinatlar dizisindeki değerlerin toplamını hesaplayarak konsola bastıran uygulamayı **reduce()** ile yazınız.

```
const toplam = koordinatlar.reduce(function (x, y) {
  return x + y;
});
console.log(toplam);
```

```
// Arrow fonskiyonu ile Daha kısa
const toplamıBul = koordinatlar.reduce((x, y) => x + y);
console.log("KOORDİNAT TOPLAMI:" + toplamıBul);
```

REDUCE() METODU

ÖRNEK: Koordinatlar dizisindeki değerlerin toplamını, ara değerleri de göstererek konsola bastıran uygulamayı **reduce()** ile yazınız.

```
const toplam = koordinatlar.reduce(function (x, y, i) {
  console.log(`iterasyon ${i} ${x}`);
  return x + y;
});
console.log(toplam);
```

x : toplam değer, **y:** anlık değer, **i:** indis

ÖRNEK: Koordinatlar dizisindeki değerlerin ortalamasını hesaplayarak konsola bastıran uygulamayı **reduce()** ile yazınız.

```
const ortalama = koordinatlar.reduce((x, y) => x + y) / koordinatlar.length;
console.log("Koordinatların Ortalaması:" + ortalama);
```

PIPELINE (HAT)

Dizi iterasyon metotları **ardı ardına** kullanılabilir. Böylelikle ardışık bir şekilde diziler işlenebilir.

ÖRNEK: Koordinatlar dizisindeki negatif koordinatları seçerek bunları pozitife çevirip alt alta konsola bastıran uygulamayı yazınız.

PIPELINE (HAT)

ÖRNEK:**Bireyler** dizisindeki kişilerden adı "**Belirtilen**" harf ile başlayanları seçerek ayrı bir diziye saklayan uygulamayı yazınız.

```
const bireyler = ["Mustafa", "Murat", "Ahmet", "mustafa", "Ayşe", "Canan"];
```

PIPELINE ÖRNEK

ÖRNEK: Bir Firma, **3000 TL** den **az** olan maaşlara **%10** zam yapmak istiyor ve zam yapılan bu kişilere **toplam** kaç TL ödeneceğini bilmek istiyor. İlgili programı yazınız.

```
const maaşlar = [3000, 2891, 3500, 4200, 7000, 2500];
```