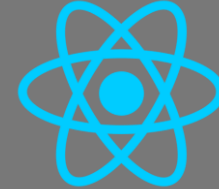


# FRONT - END



**JAVASCRIPT**



**DÖNGÜLER**

# TEKRARLANAN DURUMLAR (ITERASYON)

- Programlarda tekrarlanan işlemleri nasıl yapabiliriz.
  - Örneğin Ekran **100 kere adımızı** yazdırmak istiyoruz.
  - Girilen **1000 adet** sayıyı toplamak istiyoruz.
- Tek tek yazmak çok uzun kod demektir ve çoğu zaman mümkün değildir.
  - Çözüm **Döngü** kullanmak.
- **Döngüler** her programlama dilinde bulunmaktadır ve tekrarlanan işlemleri çok daha **az kod** kullanarak gerçekleştirmemize olanak sağlar.
- **Javascript** dili bir çok döngü deyimini ve iterasyon metodunu desteklemektedir.
  - **FOR**, FOR IN, FOR OF, **WHILE**, **DO-WHILE** (Döngü deyimleri)
  - FOREACH, MAP, FILTER, REDUCE (iterasyon metotları)

# FOR DÖNGÜSÜ

- **For** döngüsünün Syntax'ı

```
for (başlangıç; koşul; güncellemeMiktarı) {  
    // Döngü içi  
}
```

**NOT:** Koşul **doğru** olduğu müddetçe döngüye devam edilir.

Eğer, koşul **yanlış** ise bu döngüye girilmemiş olur.

**ÖRNEK:** Konsola **10 kere** bir ifade yazdıran program:

```
for(let i = 1 ; i<= 10 ; i++){  
    console.log("Merhaba");  
}
```

```
for (let i = 1; i <= 100; i++) {  
    console.log(`${i}-Merhaba`);  
}
```

# FOR DÖNGÜSÜ

**ÖRNEK:** 1 den n kadar olan sayıların toplamını yazdıran program.  
NOT: n prompt ile dışarıdan girilebilir.

# FOR DÖNGÜSÜ

**ÖRNEK:** 0-100 arasında **10 adet** rasgele sayı üreten kodu **for** döngüsü ile yazınız.

# FOR DÖNGÜSÜ (ASAL SAYI)

**ÖRNEK:** Dışarıdan girilen sayının **Asal** olup olmadığını tespit ederek sonucu yazdıran programı **for** döngüleri ile yazınız.

## NOT:

- 1 ve kendisinden başka böleni olmayan sayılar ASAL sayılardır.
- Eğer girilen sayı, herhangi bir sayıya tam bölünüyorsa diğer sayıları kontrol etmeye gerek yoktur. **ASAL DEĞİL** diyebiliriz.
- **break** deyimi ile döngü bir koşul gerçekleştiğinde kırılabilir.

# WHILE DÖNGÜSÜ

- **While** döngüsü yapı olarak for döngüsünden farklı olsa da benzer işleri yapmak içindir.

```
while ( koşul ) {  
    // Döngü içi  
}
```

- Döngü koşulu doğru olduğu müddetçe döngü tekrar ettirilir. Yanlış olur olmaz döngüden çıkılır.

**ÖRNEK:** 10 kere konsola **Merhaba** yazdıran uygulamayı **while** döngüsü ile yazınız.

```
let i = 1;  
while (i <= 10) {  
    console.log(i + " Merhaba");  
    i++;  
}
```

Eğer döngü koşulu baştan yanlış olsaydı bu döngüye hiç girilmemiş olurdu.

# WHILE DÖNGÜSÜ

**ÖRNEK:** 10 kere konsola **Merhaba** yazdıran uygulamayı **while** döngüsü ile yazınız.



# WHILE DÖNGÜSÜ

**ÖRNEK:** Kullanıcıdan 0-100 arasında bir not isteyen ve girilen not 0-100'den farklı ise Kullanıcıyı uyararak yeniden 0-100 arasında not girmeye zorlayan kodu **while** döngüsü ile yazınız.

# DO-WHILE DÖNGÜSÜ

- **Do-While** döngüsü yapı olarak **while** döngüsüne çok benzer. Tek farkı döngü koşulunun en sonda kontrol edilmesidir.
- Bu yüzden bir **do-while** döngüsü en az bir kere çalışır.

```
do{  
    // Döngü içi  
} while ( koşul ) ;
```

**ÖRNEK:** 10 kere konsola **Merhaba** yazdıran uygulamayı **do-while** döngüsü ile yazınız.

```
let sayac = 1;  
do {  
    console.log("Merhaba -", sayac);  
    sayac++;  
} while (sayac <= 10);
```

# DO-WHILE DÖNGÜSÜ

**ÖRNEK:** Kullanıcıdan 0-100 arasında bir not isteyen ve girilen not 0-100'den farklı ise Kullanıcıyı uyararak yeniden 0-100 arasında not girmeye zorlayan kodu **do-while** döngüsü ile yazınız.

# DO-WHILE DÖNGÜSÜ

**ÖDEV:** Klavyeden Q veya q karakteri girilene kadar not girişi yapan ve bu karakterlerden birisi girildiğinde O ana kadar girilen tüm notların ortalamasını hesaplayarak konsola ortalamayı bastıran uygulamayı yazınız.

# DO-WHILE DÖNGÜSÜ

## **ÖDEV:** Tahmin Oyunu

- Program 0-100 arasında rasgele bir sayı tutmalı ve kullanıcının bu sayıyı 5 kerede (hak) bilmesini istemelidir.
- Her yanlışta hakkını bir düşürmeli ve ARTTIR/AZALT diyerek kullanıcıyı yönlendirmelidir.
- Sonuç olarak kullanıcının hakkı 0 olursa "Üzgünüz bilemediniz" eğer bildi ise "Tebrikler ... denemede bildiniz" diye bir bilgi mesajı yazdırmalıdır.