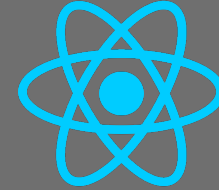


FRONT-END



JAVASCRIPT



DÖNGÜLER

TEKRARLANAN DURUMLAR (ITERASYON)

- Programlarda tekrarlanan işlemleri nasıl yapabiliriz.
 - Örneğin Ekranı adımızı **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 döngüyü desteklemektedir.
 - **FOR, FOR IN, FOR OF, FOREACH, MAP, FILTER**
 - **WHILE, DO-WHILE**

FOR DÖNGÜSÜ

- **For** döngüsünün Syntax'ı

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

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

Eğer, koşul baştan **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 10 kadar olan sayıların toplamını yazdıran program:

```
let toplam=0;
for (let t = 1; t <= 10; t++) {
  toplam += t;
}
console.log(toplam);
```

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

```
const n = prompt("Hangi Sayıya kadar toplamak istiyorsunuz:");
let toplam=0;
for (let t = 1; t <= n; t++) {
  toplam += t;
}
console.log(toplam);
```

FOR DÖNGÜSÜ (ASAL SAYI)

ÖRNEK: Dışarıdan girilen sayının **Asal** olup olmadığını tespit ederek sonucu yazdıran programı yazınız. **NOT:** 1 ve kendisinden başka böleni olmayan sayılar ASAL sayılardır.

```
const sayı = prompt("Pozitif Bir Sayıyı Giriniz:");
const asalMı = (sayı) => {
  let asal = true;
  for(let i = 2 ; i<sayı; i++){
    if (sayı%i == 0) {
      asal = false;
      break;
    }
  }
  return asal ? "ASAL" : "ASAL DEĞİL";
}
console.log(`${sayı} ${asalMı(sayı)}`);
```

NOT: Eğer girilen sayı, herhangi bir sayıya tam bölünüyorsa diğer sayıları kontrol etmeye gerek yoktur. **ASAL DEĞİL** diyebiliriz.

Bu durumda döngünün devam etmesi bize zaman kaybettirecektir.

break deyimi ile döngünün bitmesi sağlanır.

FOR DÖNGÜSÜ (ASAL SAYI)

ÖRNEK: Dışarıdan girilen sayının **Asal** olup olmadığını tespit ederek sonucu yazdıran programı yazınız. **NOT:** 1 ve kendisinden başka böleni olmayan sayılar ASAL sayılardır.

```
const sayı = prompt("Pozitif Bir Sayıyı Giriniz:");
const asalMı = (sayı) => {
  let asal = true;
  for(let i = 2 ; i<sayı; i++){
    if (sayı%i == 0) {
      asal = false;
      break;
    }
  }
  return asal ? "ASAL" : "ASAL DEĞİL";
}
console.log(`${sayı} ${asalMı(sayı)}`);
```

NOT: Eğer girilen sayı, herhangi bir sayıya tam bölünüyorsa diğer sayıları kontrol etmeye gerek yoktur. **ASAL DEĞİL** diyebiliriz.

Bu durumda döngünün devam etmesi bize zaman kaybettirecektir.

break deyimi ile döngünün bitmesi sağlanır.

WHILE DÖNGÜSÜ

- **While** döngüsü yapı olarak for döngüsünden farklı olsa da benzer işleri yapmak içindir.
- Döngü koşulu doğru olduğu müddetçe döngü tekrar ettirilir. Yanlış olur olmaz döngüden çıkılır.

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

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.

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: Dışarıdan 0-100 (not) dışında bir değer girilmesini not engelleyen kodu yazınız.

```
let not;  
do {  
    not = prompt("0-100 arasında bir not giriniz:");  
} while (not < 0 || not > 100);  
console.log("Girdiğiniz not 0-100 arasındadır");
```