Mantıksal Operatörler

JavaScript dilinde üç tane mantıksal operatör bulunmaktadır: || (OR – VEYA), &&(AND – VE), ! (NOT – DEĞİL)

Mantıksal operatörler olarak adlandırılsalar bile, her tipteki değer için uygulanabilirler. Sadece boolean (doğru-yanlış) değerleri için değil. Sonuçta her tipte olabilir.

Detaylarına bakılacak olursa:

1. || (OR – VEYA)

"OR","VEYA" operatörü iki dikey çizgiden oluşur.

```
sonuc = a || b;
```

Klasik programlamada, mantıksal VEYA sadece boolean verileri değiştirme için kullanılır. Eğer iki değerden biri true ise sonuç true döner. Diğer türlü sonuç false döner.

JavaScript'te ise bu biraz daha karmaşık ve daha güçlü. Fakat önce boolean değerlere ne oluyor buna bakalım.

Dört farklı mantıksal kombinasyon bulunmakta:

```
alert( true || true );   // true

alert( false || true );  // true

alert( true || false );  // true

alert( false || false ); // false
```

Her iki tarafın da false olmadığı her durumda sonuç true olmakta.

Eğer operanda yani işleme giren değerler boolean değil ise boolean değere çevrilir.

Örneğin, sayı olan 1 true demek sayı olan 0 ise false demektir.

```
if (1 || 0) { // ( true || false ) ile aynı anlama gelir
  alert( 'Doğru!' );
}
```

Çoğu zaman, VEYA || if yapısı içerisinde kullanılır. *Herhangi biri* doğruysa yap anlamı taşımaktadır.

Örneğin:

```
let saat = 9;

if (saat < 10 || saat > 18) {
  alert( 'Ofis Kapalı' );
}
```

Birçok şart cümlesi ile if yapısını kurabilirsiniz.

```
let saat = 12;
```

```
let haftaSonu = true;

if (saat < 10 || saat > 18 || haftaSonu) {

  alert( 'Ofis Kapalı.' ); // Haftasonu

}
```

VEYA ilk doğru değeri arar

Yukarıda belirtilen mantık klasik mantıktır. JavaScript'in "ekstra" özelliklerine bakılacak olursa

Geliştirilmiş algoritma şu şekildedir.

Birden fazla VEYA ile yapılmış if yapısı:

```
sonuc = deger1 || deger2 || deger3;
```

VEYA "||" operatörü şunları yapar:

- Soldan sağa olacak şekilde operandları değerlendirir.
- Her operandın değerini boolean'a çevirir. Eğer sonuç doğru ise durur ve o operandın orjinal değerini döner.
- Eğer tüm operandlar kontrol edildi ve tamamı yanlış ise son operandı döner.

Eğer VEYA zincirinde bir tane doğru bulunursa o an dönülür. Eğer bulunamazsa sonuncusu döner.

Örneğin:

```
alert( 1 || 0 ); // 1 (1 doğru)

alert( true || 'önemsiz' ); // (true doğru)

alert( null || 1 ); // 1 (1 tek doğru veri)

alert( null || 0 || 1 ); // 1 (1 tek doğru veri)

alert( undefined || null || 0 ); // 0 (Hepsi yanlış sonuncusunu döner)
```

Bu klasik "boolean" VEYA tanımını aşarak ilginç kullanımlara neden olmaktadır.

1. Değişken veya ifadeler dizisinde ilk doğru(true) değeri bulmak için

Düşünün bir diziniz var ve içinde null/undefined değerler barındırmakta. Siz ilk veriyi bulduğunuzda döndürmek istiyorsunuz.

Bunun için || kullanabilirsiniz:

```
let simdikiKullanici = null;

let varsayilanKullanici = "Akif";

let isim = simdikiKullanici || varsayilanKullanici || "isimsiz";

alert( isim ); // "Akif" seçilir – ilk doğru değeri bulduğundan dolayı buradan dönülür ve ekrana "Akif" çıkar.
```

Eğer simdikiKullanici ve varsayilanKullanici yanlış(false) olsaydı "isimsiz" yazısı ekrana çıkacaktı.

2. Kısa devre değerlendirmesi

Operantlar sadece değer değil ifade de olabilir. VEYA testlerini soldan sağa doğru yapar. Doğru değer bulunduğunda döndürülür. Bu olaya kısa devre değerlendirmesi denir, çünkü soldan sağa en kısa yoldan gitmektedir.

Tabi bunun ifadelere yan etkisi olabilir. Örneğin değer atama

Aşağıdaki örnek çalıştığında x'e değer atanmayacak:

```
let x;

true || (x = 1);

alert(x); // tanımsız, çünkü (x = 1) ifadesi çalıştırılmadı
```

Eğer if yapısında ilk değer false ise bir sonrakine bakılır bu da şu şekilde sonuç verir:

```
let x;

false || (x = 1);

alert(x); // 1
```

Gördüğünüz gibi değer atandı. Böyle basit bir durumda yan etki görmezden gelinebilir.

Kısa yoldan if yapısında olduğu gibi ilk operand boolean'a çevrilir ve eğer yanlışsa ikinci değer çalıştırılır.

Çoğu zaman normal if yapısını kullanmanız daha iyidir çünkü kod daha anlaşılır olur. Fakat bazen kısa yoldan if yapmakta işinize yarayabilir.

2. && (AND – VE)

Ve operatörü iki tane & işaretiyle tanımlanmaktadır.

```
sonuc = a && b;
```

Klasik programlamaya göre eğer iki operandda doğru ise doğru, diğer türlü yanlış döner.

```
alert( true && true );   // true

alert( false && true );  // false

alert( true && false );  // false

alert( false && false ); // false
```

if ile bir örnek:

```
let saat = 12;

let dakika = 30;

if (saat == 12 && dakika == 30) {

  alert( 'Saat 12:30' );

}
```

VEYA'da olduğu gibi VE operatörü de her türlü değeri kabul eder.

```
if (1 && 0) { // true && false şeklinde değerlendirilmiştir.
  alert( "Çalışmaz çünkü sonuç `yanlış` " );
}
```

VE ilk yanlış değeri görür

Aşağıda 3 tane AND işlemine sokulmuş değer bulunmaktadır:

```
sonuc = deger1 && deger2 && deger3;
```

AND "&&" operatörü aşağıdaki gibi çalışır:

- Operandları soldan sağa doğru değerlendir.
- Her bir operandı boolean değere çevir. Eğer sonuç yanlış ise dur ve operatörün orijinal değerini döndür.
- Eğer diğer operandlara erişim sağlandıysa (hepsinin doğru olma durumu) sondaki operandı döndür.

Yukarıdaki kurallar VEYA kuralları ile benzerlik göstermektedir. Farklılık AND operatörünün ilk yanlış bulduğunda dönmesi. OR operatörü ise ilk doğru bulduğunda dönmekteydi.

Örnek:

```
// Eğer ilk operand doğru ise her halükarda ikincinin değeri dönecek.
alert( 1 && 0 ); // 0
alert( 1 && 5 ); // 5
// İlk operand yanlış ise ilk operandı döner ikinci operand pas geçilir.
alert( null && 5 ); // null
alert( 0 && "önemi yok" ); // 0
```

Birden fazla VE'yi if yapısıyla kullanmak mümkündür.

```
alert( 1 && 2 && null && 3 ); // null
```

Tüm değerler doğru ise sonuncu değer döner.

```
alert( 1 && 2 && 3 ); // 3,  sonuncu değer
```

VE && VEYA'dan || önce çalışır.

VE'nin && önceliği VEYA'ya || göre daha yüksektir. Bundan dolayı VEYA'dan önce çalışır.

Aşağıdaki örnekte 1 && 0 önce hesaplanır.

```
alert( 5 || 1 && 0 ); // 5
```

VEYA'da olduğu gibi VE'de de operatör bazen if yerine kullanılabilir.

Örneğin:

```
let x = 1;
(x > 0) && alert( 'Sıfırdan Büyük' );
```

Sağ taraftaki bildirim sadece değerlendirme oraya kadar gelebilirse çalışır. Bunun için de x>0'ın doğru dönmesi gerekmektedir.

Aslında aşağıdaki ile benzerdir:

```
let x = 1;

if (x > 0) {

  alert( 'Sıfırdan büyük!' );

}
```

&& ile yazılan çeşidi daha kısa gibi görünse de aslında if ile yazılanın daha okunabilir olduğu açıktır.

Bundan dolayı her yapıyı amacına göre kullanmanız önerilir. Eğer if kullanmak istiyorsanız if yazarak kullanın. Eğer VE kullanmak istiyorsnaız && yazarak kullanın.

3. ! (DEĞİL)

Boolean değil operatörü "!" ile tanımlanmıştır.

Yazımı çok kolaydır:

```
result = !value;
```

Operatör tek operanddan oluşur ve aşağıdaki şekilde çalışır:

1. Operand değerini boolean tipine çevir: true/false
2. Tersini geri döndür.

Örneğin:

```
alert( !true ); // false

alert( !0 ); // true
```

Çift DEĞİL işareti değeri boolean tipine çevirmeye yarar:

```
alert( !!"Boş olmayan karakter dizisi" ); // true

alert( !!null ); // false
```

Birinci DEĞİL değeri booleana çevirir ve tersini alır. İkincisi ise tersinin tersini alarak değeri orjinal halinin boolean haline çevirir.

Aynı şeyi Boolean fonksiyonu ile de yapmak mümkündür.

```
alert( Boolean("boş olmayan karakter dizisi") ); // true

alert( Boolean(null) ); // false
```

Görevler

VEYA'nın sonucu nedir?

önem: 1

Aşağıdaki kodun çıktısı nedir?

```
alert( null || 2 || undefined );
```

VEYA'landırılmış uyarıların çıktısı ne olur?

önem: 2

Aşağıdaki kodun çıktısı nedir?

```
alert( alert(1) || 2 || alert(3) );
```

VE'nin sonucu nedir?

önem: 3

Aşağıdaki kodun çıktısı nedir?

```
alert( 1 && null && 2 );
```

VE'lendirilmiş uyarıların çıktısı ne olur?

önem: 4

Aşağıdaki kodun çıktısı nedir?

```
alert( alert(1) && alert(2) );
```

VEYA ve VE'nin sonucu ne olur?

önem: 5

Aşağıdaki kodun çıktısı nedir?

```
alert( null || 2 && 3 || 4 );
```

Aralık kontrolü

önem: 6

yaş'ı 14 ile 90 arası olanları kontrol eden if koşulunu yazınız? Not: 14 ve 90 dahil.

Aralığın dışındaki değerleri yazınız.

önem: 7

yaş'ı 14 ile 90 arasında olmayanları bulan if koşulunu yazınız. Not: 14 ve 90 dahil.

Bunu ! kullanarak ve kullanmayarak iki şekilde yapın.

"if" hakkında bir soru.

önem: 8

Hangi alert çalışacak?

```
if(...) içerisindeki değerin sonucu ne olacak?
```

```
if (-1 || 0) alert( 'birinci' );

if (-1 && 0) alert( 'ikinci' );

if (null || -1 && 1) alert( 'üçüncü' );
```

Check the login

önem: 9

Write the code which asks for a login with prompt.

If the visitor enters "Admin", then prompt for a password, if the input is an empty line or Esc – show “Canceled.”, if it’s another string – then show “I don’t know you”.

The password is checked as follows:

- If it equals “TheMaster”, then show “Welcome!”,
- Another string – show “Wrong password”,
- For an empty string or cancelled input, show “Canceled.”

The schema:

Please use nested if blocks. Mind the overall readability of the code.

Hint: passing an empty input to a prompt returns an empty string ''. Pressing ESC during a prompt returns null.