20 Mayıs 2022

WEB PROGRAMLAMA - BIL104

Öğr. Gör. Buse Yaren TEKİN

İçerikler

JavaScript Karşılaştırma Operatörleri
JavaScript Mantıksal Operatörler
Ekrana Çıktı Ve Klavyeden Bilgi Giriş

Karşılaştırma Operatörleri

Bölüm 1

Karşılaştırma Operatörleri

Değişkenlerin birbirleri ile karşılaştırılmak istendiğinde kullanılan operatörlerdir.

Bu operatörler ise;

- == operatörü iki değişkenin birbirine eşitliğini kontrol eder.
- != operatörü iki değişkenin birbirine eşit olmadığı durumlarda kullanılır.
- < operatörü bilindiği üzere küçüktür operatörüdür. Soldaki değişkenin sağdakinde küçüklüğünü kontrol eder.

Karşılaştırma Operatörleri

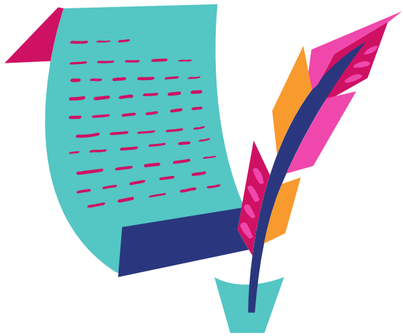
Değişkenlerin birbirleri ile karşılaştırılmak istendiğinde kullanılan operatörlerdir.
Bu operatörler ise;

- <= soldaki değişkenin sağdaki değişkene küçük eşitliğini kontrol eder.
- > soldaki değişkenin sağdaki değişkene göre büyük olup olmadığını kontrol eder.
- >= soldaki değişkenin sağdaki değişkene büyük eşitliğini kontrol eder.

Mantıksal Operatörler

İki değişkene bağlı karşılaştırılmaların yapılmak istendiği durumlarda kullanılır.
Operatörler **&& , || , !** operatörleridir.

- **&& And (ve)** operatörü iki değişkenin de değeri doğru olması istendiğinde kullanılır.
- **|| Or (veya)** operatörü iki değişkenden en az birinin doğru olması durumu istediğinde kullanılır.
- **! Not (değil)** operatörü değişkenin değeri doğru ise yanlış , yanlış ise doğru olması istendiği durumlarda kullanılır.



Özel Karşılaştırma Operatörü

Bu operatör iki değişken arasında karşılaştırma yapmanın en sade ve kısa yoludur. Operatörün kullanım biçimi :

- **değişken_1 [karşılaştırma operatörü] değişken_2 ? değişken_3 : değişken_4**

Bunu bir örnekle açıklayalım. Değişkenleri var ile tanımladığımızı farz ediyorum. Buna göre ;

- `a < b ? c : d`

Yukarıdaki satırda yapılması istenen işlem; a değişkeninin b değişkeninden küçük olup olmadığı karşılaştırılıyor. Buna göre cevap doğruysa işlemin sonucu c değişkeninin değeri değilse d değişkeni oluyor.

Özel Karşılaştırma Operatörü

```
var i=1; var j=2;
var k=3; var m=4;
var n=5;
var p=6;
var q=7;
i=+j;
j++;
k--;
```

```
m=m+k;
n=*j;
i < j ? 3 : 1 ;
k >= n ? 0 : 1 ;
k=2 && j=5 ? p : q ;
i=2 || j=3 ? m : n ;
p!=2 ? k : 10 ;
```

Özel Karşılaştırma Operatörü

İlk yedi satırda değişkenlerimizi hem tanımladık hem de değer atadık. Böyle bir yazımı yapabileceğimizi değişkenleri anlatmaya başlarken söylemiştik.
İşlem satırlarına geçtiğimizde ise;

```
i+=j;
```

Bu işlem daha da önce gördüğümüz gibi bize i=i+j işlemini yapmamızı söyler.
Buna göre i değişkeninin değeri 3 olacaktır.

Özel Karşılaştırma Operatörü

Hemen altındaki satırda bulunan j++ işlemi dolayısıyla da j değişkeni 3 değerini alacaktır. Diğer işlem satırında k--işlemi ile de k değişkeni 2 değerini alır. Bir diğer satırdaki m=m+k işlemi ile m(m=4) değişkeni k(k=2) değişkeni toplanarak 6 değerini alır. n=*j işlemi ile de n(n=5) değişkeni 3*5=15 değerini alacaktır.

```
j++;

k--;

m=m+k;

n=*j;
```

Özel Karşılaştırma Operatörü

Şimdi diğer karşılaştırma işlemlerine geçmeden önce değişkenlerimizin işlem sonrası aldığı değerleri yazalım.

```
i=3 , j=3 , k=2 , m=6 , n=15 , p=6 , q=7 ; i < j ? 3: 1 ;
```

Bu satırın 3 < 3 işleminin cevabı doğru ise 3 değilse 1 değeri alacağını görebiliyoruz. Tabi ki üç üçten küçük olmadığı için cevabımız 1 olacaktır.

Özel Karşılaştırma Operatörü

```
k>=n ? 0 : 1 ;
```

Bu satırda ise 2>=15 işlemi gerçekleşir ki bunun cevabı da yanlıştır ve ikinci değer işlem satırının cevabıdır yani 1 dir. Şimdiki karşılaştırma işlemimiz ise mantıksal operatörlerle ilgili. Buna göre; k=2 && j=5 ? p : q ; İşlem bize ne söylüyor ? K değişkeni ve j değişkeninin kesin olarak bir değere eşit olup olmadığını karşılaştırmamızı söylüyor. Bu iki değer de doğruysa çünkü &&(and) mantıksal operatörünün anlamı bu işlem doğrudur değilse yanlıştır. Buna göre k=2 'dir. Fakat buna karşılık j'nin değeri 5 değildir. Bu yüzden karşılaştırmanın cevabı yanlıştır. Dolayısıyla işlem q yani 7 değerini alır.

Özel Karşılaştırma Operatörü

p!=2 ? k : 10 ;

İşlemde istenilen p değişkeninin değerinin ikiden farklı olması durumdur. Yani 6!=2 bunun anlamı doğrudur. Gerçektende 6=2 değildir. Bizde bu satırda bunu istiyorduk. O halde cevap doğrudur. Böylelikle işlem k yani 2 değerini alır. Şimdi biz bu yaptıklarımızla sadece javascript'te bir şeyler hesap etmesini ve karşılaştırmasını söyledik. Tarayıcı da bu işlemleri yapar ve hafızasında tutar. Daha sonra öğreneceğimiz komutlarla bunları istersek tarayıcıya yazdırabilir. Başka bir yerde kullanılmasını söyleyebiliriz.

JavaScript Veri Tipleri

JavaScript değişkenleri farklı veri türlerini tutabilir: sayılar, dizeler, nesneler ve daha fazlası:

```
let length = 16;                               // Number
let lastName = "Johnson";                      // String
let x = {firstName:"John", lastName:"Doe"};    // Object
```

Veri Türleri Kavramı

Programlamada veri türleri önemli bir kavramdır.

Değişkenler üzerinde işlem yapabilmek için tür hakkında bir şeyler bilmek önemlidir.

Veri türleri olmadan bir bilgisayar bunu güvenli bir şekilde çözemez:

```
let x = 16 + "Volvo";
```

Veri Türleri Kavramı

On altıya "Volvo" eklemek bir anlam ifade ediyor mu? Bir hata mı üretecek yoksa bir sonuç mu üretecek?

Veri Türleri Kavramı

```
<body>

<h2>JavaScript</h2>

<p>When adding a number and a string,
JavaScript will treat the number as a string.
</p>

<p id="demo"></p>

<script>
let x = 16 + "Volvo";
document.getElementById("demo").innerHTML = x;
</script>

</body>
```

JavaScript

When adding a number and a string, JavaScript will treat the nu as a string.

16Volvo

Veri Türleri Kavramı

JavaScript, ifadeleri soldan sağa doğru değerlendirir. Farklı diziler farklı sonuçlar üretebilir:

```
let x = 16 + 4 + "Volvo";
```

20Volvo

Veri Türleri Kavramı

JavaScript, ifadeleri soldan sağa doğru değerlendirir. Farklı diziler farklı sonuçlar üretebilir:

```
let x = "Volvo" + 16 + 4;
```

Volvo164

JavaScript Türleri

JavaScript'in dinamik türleri vardır. Bu, aynı değişkenin farklı veri türlerini tutmak için kullanılabileceği anlamına gelir:

```
<body>

<h2>JavaScript Data Types</h2>

<p>JavaScript has dynamic types. This means that the same variable can be used to
hold different data types:</p>

<p id="demo"></p>

<script>
let x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";     // Now x is a String

document.getElementById("demo").innerHTML = x;
</script>

</body>
```

JavaScript Türleri

```
<body>

<h2>JavaScript Strings</h2>

<p>Strings are written with quotes. You can use single or double quotes:</p>

<p id="demo"></p>

<script>
let carName1 = "Volvo XC60";
let carName2 = 'Volvo XC60';

document.getElementById("demo").innerHTML =
carName1 + "<br>" +
carName2;
</script>

</body>
```

Çıktı?

JavaScript Türleri

JavaScript Strings

Strings are written with quotes. You can use single or double quotes:

Volvo XC60
Volvo XC60

JavaScript Türleri

```
<body>

<h2>JavaScript Strings</h2>

<p>You can use quotes inside a string, as long as they don't match the q
surrounding the string:</p>

<p id="demo"></p>

<script>
let answer1 = "It's alright";
let answer2 = "He is called 'Johnny'";
let answer3 = 'He is called "Johnny"';

document.getElementById("demo").innerHTML =
answer1 + "<br>" +
answer2 + "<br>" +
answer3;
</script>

</body>
```

JavaScript Strings

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

It's alright
He is called 'Johnny'
He is called "Johnny"

JavaScript Booleans

Boolean'ların yalnızca iki değeri olabilir: **true** veya **false**. Boole değerleri genellikle koşullu testlerde kullanılır.

```
<h2>JavaScript Booleans</h2>

<p>Booleans can have two values: true or false:</p>

<p id="demo"></p>
<script>
let x = 5;
let y = 5;
let z = 6;
document.getElementById("demo").innerHTML =
(x == y) + "<br>" + (x == z);
</script>

</body>
```

JavaScript Booleans

Booleans can have two values: true or false:

true
false

JavaScript Diziler

JavaScript dizileri köşeli parantezlerle yazılır. Dizi öğeleri virgülle ayrılır.

Aşağıdaki kod, üç öğe (araba adları) içeren arabalar adlı bir dizi bildirir (oluşturur):

```
<h2>JavaScript Arrays</h2>

<p>Array indexes are zero-based, which means the first item is [0].</p>

<p id="demo"></p>

<script>
const cars = ["Saab","Volvo","BMW"];

document.getElementById("demo").innerHTML = cars[0];
</script>
```

Dizi dizinleri sıfır tabanlıdır; bu, ilk öğenin [0], ikinci öğenin [1] olduğu vb. anlamına gelir.

JavaScript Diziler

JavaScript dizileri köşeli parantezlerle yazılır. Dizi öğeleri virgülle ayrılır.

Aşağıdaki kod, üç öğe (araba adları) içeren arabalar adlı bir dizi bildirir (oluşturur):

```
<h2>JavaScript Arrays</h2>

<p>Array indexes are zero-based, which means the first item is [0].</p>

<p id="demo"></p>

<script>
const cars = ["Saab","Volvo","BMW"];

document.getElementById("demo").innerHTML = cars[0];
</script>
```

KAYNAKLAR

1. Wikipedia, Kavramlar.
2. Anadolu Üniversitesi, Web Tabanlı Kodlama.
3. W3schools, https://www.w3schools.com/default.asp adresinden alınmıştır.