

JAVASCRIPT DİLİ VE KOMUTLARI



İÇİNDEKİLER

- Javascript ve HTML Etiketleri ilişkisi
- Değişkenler
- Operatörler
- Koşul Komutları ve Koşul Operatörü



HEDEFLER

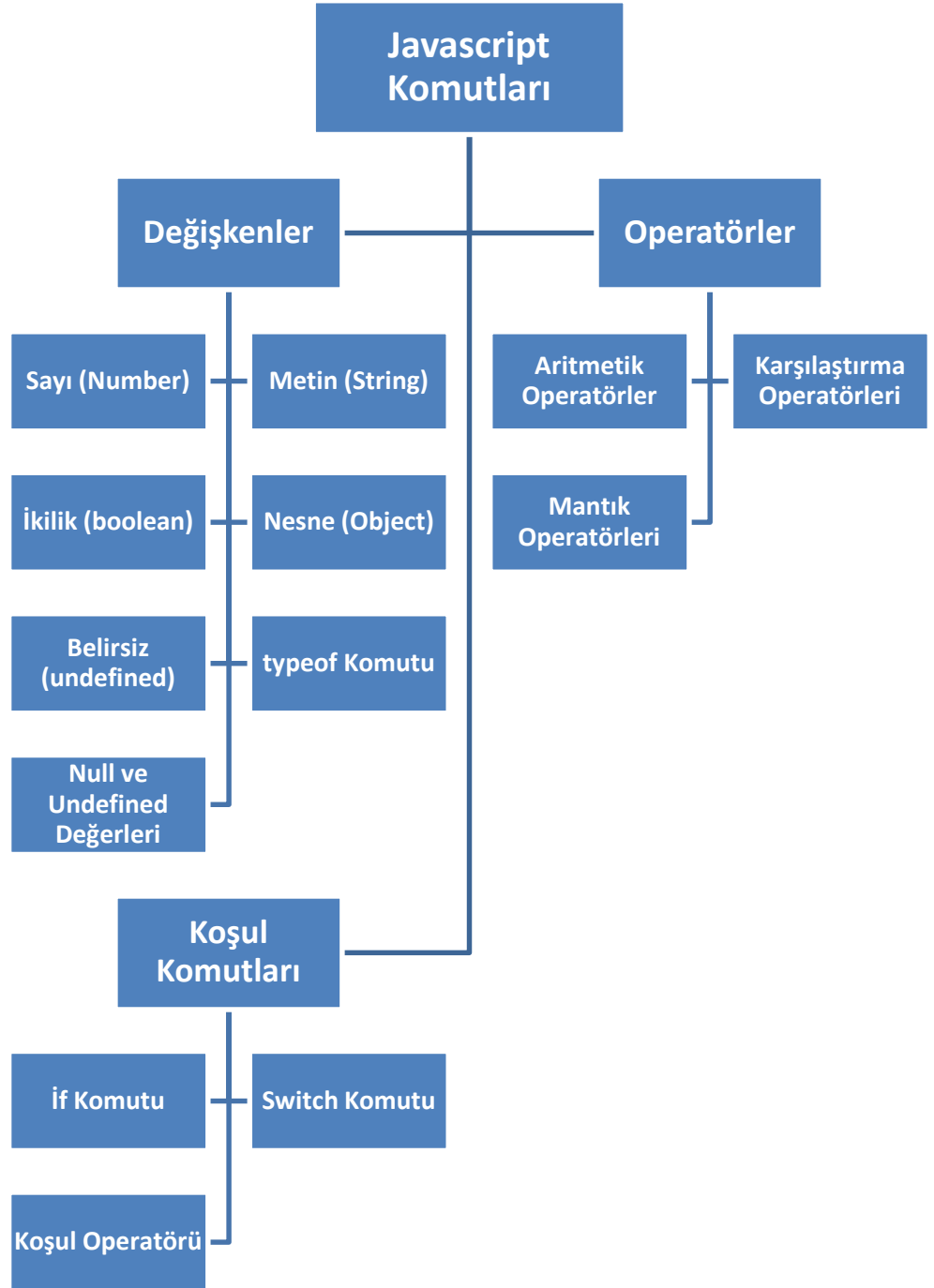
- Bu üniteyi çalıştıktan sonra;
 - Javascript kodları ile HTML etiketlerine ve sayfalarına çıktı yazabilecek,
 - Değişken tanımlayabilecek ve uygun türü seçebilecek,
 - Değişken ve/veya değerler ile işlem yapmak için gerekli operatörleri kullanabilecek,
 - Bir koşula bağlı olarak programın akışını yönlendirebileceksiniz.



Atatürk Üniversitesi
Açıköğretim Fakültesi

**İNTERNET
PROGRAMCILIĞI I**
**Dr. Öğr. Üyesi Halil
ERSOY**

**ÜNİTE
10**



GİRİŞ

Javascript, dil özellikleri bakımında C/C++ programlama dillerine benzer. Bu benzerliğin ilk akla gelen örneği komutların noktalı virgül ile sonlanması gerektiği ve büyük/küçük harf ayrımının var olmasıdır. Öte taraftan komutların gruplanması gerektiğinde küme parantezlerinin ({ }) kullanılması zorunluluğu da aynı benzerliktedir.

Javascript kodlarını var olan sözdizim kurallarına uygun biçimde yazma işlemi başta zor olabilir. Javascript'i ilk defa öğrenenler, HTML etiketlerini yazarken yaşadıkları esnekliğin Javascript'te olmadığını fark edeceklerdir. Örneğin HTML etiketlerinde kapatılması gereken bir etiket kapatılmadığında tarayıcılar bunu yorumlama aşamasında tolere ederek sanki kapanış etiketi varmış gibi davranabilirler. Ayrıca HTML etiketleri büyük veya küçük harfe duyarlı değildir. Bu nedenle çoğu geliştirici, HTML dilini yazarken Javascript ile kodlamaya geçtiklerinde sözdizim hataları ile çok fazla uğraşmak zorunda kalırlar.



Sözdizim, yani yazım, hataları, çoğu editör tarafından kodu yazar yazmaz algılanabilir.

Sözdizim hatalarını en aza indirmek için size tavsiyemiz, öncelikle çeşitli kaynaklardan gördüğünüz kodları bire bir alarak çalıştırmayı denemenizdir. Ardından çalışan kodlarda değişiklik yaparak testler yapmanızdır. Bu sayede olası bir hatanın tespiti kolay olacaktır.

Yazım hatalarından kaçınmak için bir başka tavsiye ise Javascript diline destek veren editör kullanmaktır. Önceki ünite de belirtilen editörlerin tümü bu konuda size yardımcı olacak özelliklere sahiptir. Bu özellikler;

- Komutları hatırlatma,
- Yanlış yazılan komutların altını çizme,
- Kapatılması unutulmuş parantezler olduğunda uyarı verme,
- İsimlendirme kurallarına uymayan durumlarda uyarı verme,
- Nesne tabanlı programlamada nesne fonksiyon ve üyelerini hatırlatma gibi sıralanabilir.

Bu ünite de, Javascript ile kodlamaya başlayacağız. Bunun için dili oluşturan değişkenler, operatörler ve koşul ifadelerini ele alacağız. Her programlama dilinde ana yapı taşları olarak kullanılan bu öğeler, hiç programlama bilmeyenler için belki zor gelebilir, ancak komutları olabildiğince HTML sayfalarına hitap edecek biçimde sunmaya çalışacağız. Bu noktada tavsiyemiz, gösterilen konuların mutlaka seçilecek bir editör ile bire bir test edilmesi ve ondan sonra geliştirilmeye çalışılmasıdır.


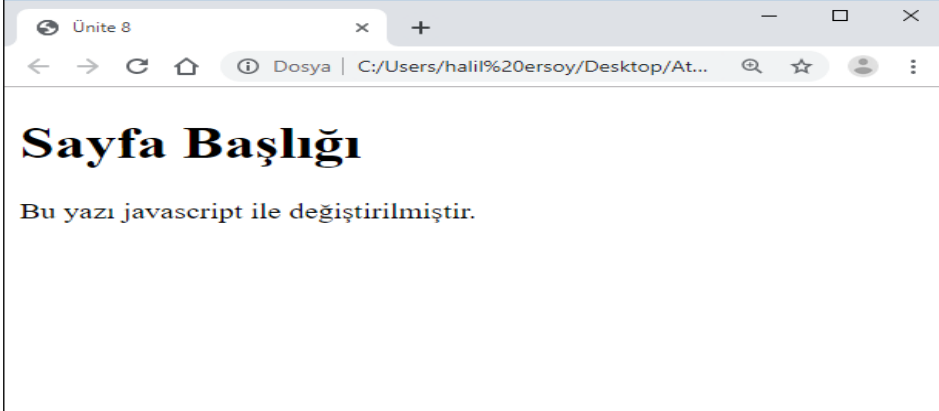
JAVASCRIPT VE HTML ETİKETLERİ İLİŞKİSİ

Ünite de ele alınacak konuların HTML sayfalarının içerisindeki diğer içerik veya etiketler ile iletişim hâlinde olması için, Javascript ile HTML etiketleri arasındaki bağlantıyı gösteren basit komutlar ile başlayacağız.

HTML Etiketlerine Javascript ile Erişmek

Javascript dili ile HTML sayfalarındaki tüm etiketlerin özelliğini veya içeriğini değiştirmek mümkündür. Bunun için `document.getElementById()` komutu kullanılır.

Tablo 10.1. `document.getElementById()` fonksiyonu

HTML ve Javascript Komutları	
 <p><code>document.getElementById()</code> komutundaki harflerin büyük veya küçük olması önemlidir, kodlama yaparken buna dikkat etmelisiniz.</p>	<pre><!DOCTYPE html> <html lang="tr"> <head> <meta charset="UTF-8"> <title>Ünite 8</title> </head> <body> <h1>Sayfa Başlığı</h1> <p id="yazi1">Bu bir paragraftır.</p> <script> document.getElementById("yazi1").innerText="Bu yazı javascript ile değiştirilmiştir."; </script> </body> </html></pre>
Web Sayfası Görüntüsü	
	

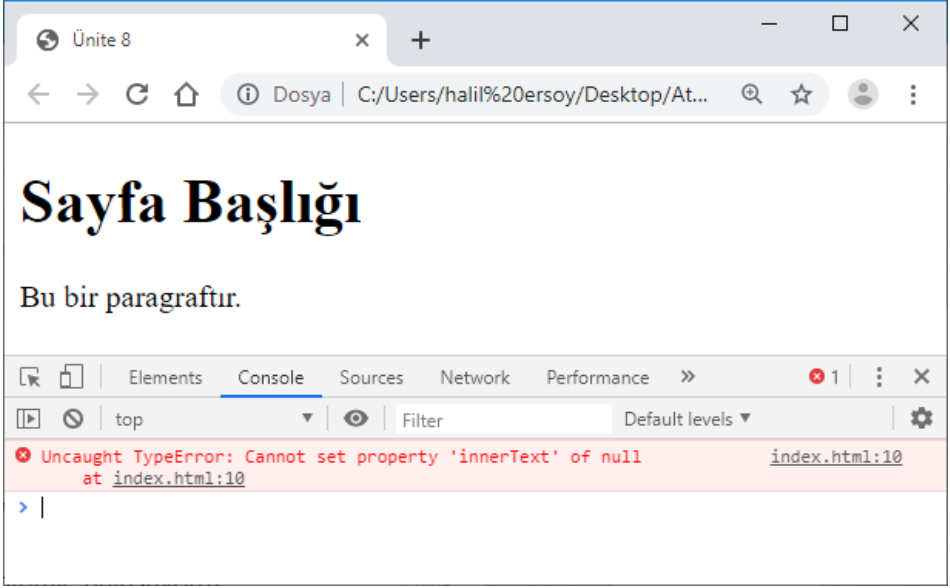
Yukarıdaki kod ve çıktısını incelediğimizde, Javascript komutları ile bir paragrafın *içerisinde* yazan yazının değiştirildiğini göreceksiniz. Bu komutun doğru şekilde çalışması için aşağıdaki koşulların sağlanmış olduğuna dikkat ediniz:

- HTML etiketlerine Javascript komutları ile erişmek için “id” özelliği ile *benzersiz* bir isim verilmelidir. Bu isim *Javascript isimlendirme kurallarına* uymalıdır. Aynı HTML sayfası içerisinde *aynı id’ye* sahip başka eleman *olmamalıdır*.
- `document.getElementById()` fonksiyonunun parantezleri içerisine bu “id” bilgisi *çift tırnak/tek tırnak* içerisinde yazılmalıdır.
- `document.getElementById()` komutu, *etiket yüklendikten sonra çalışabilir*, öncesinde çalışmaz.

Özellikle yukarıdaki son koşulu iyi anlamak gerekir. Etiketlere ulaşmak için kullanılacak Javascript komutları, *etiketler yüklendikten sonra* bu erişimi

sağlayabilir. Aksi durumda etiket bulunamaz ve *çalışma hatası* verilir. Bu olası hatalı durumu göstermek için aşağıdaki kodları ve çıktığı inceleyelim.

Tablo 10.2. Mantık Hatası

HTML ve Javascript Komutları	
<pre><!DOCTYPE html> <html lang="tr"> <head> <meta charset="UTF-8"> <title>Ünite 8</title> </head> <body> <h1>Sayfa Başlığı</h1> <script> document.getElementById("yazi1").innerText="Bu yazı javascript ile değiştirilmiştir."; </script> <p id="yazi1">Bu bir paragraftır.</p> </body> </html></pre>	
Web Sayfası Görüntüsü	
	



Çoğu hata mesajı size anlamsız gelebilir, ancak hatanın olduğu satırın numarasını görmeniz, hatayı anlamanızı kolaylaştırır.

Yukarıdaki örnekte Javascript kodu, paragraf etiketinin içeriğini değiştirememiştir. Bunun nedeni, Javascript kodunun *çalıştığı sırada* henüz ilgili paragraf etiketi *yüklenmemiştir*. Bu nedenle tarayıcıda görülen hata mesajı ortaya çıkmıştır (Tarayıcının tespit ettiği hataları göstermesi için **F12** tuşuna basarak *geliştirici seçeneklerini* ve *Console* sekmesini açmalısınız).

Örnekteki “innerText” özelliği, p, h1 (tüm başlık etiketleri), div, span, td gibi *içerisinde metin barındıran tüm etiketler* için kullanılabilir.

Basit Mesajlar Yazdırma

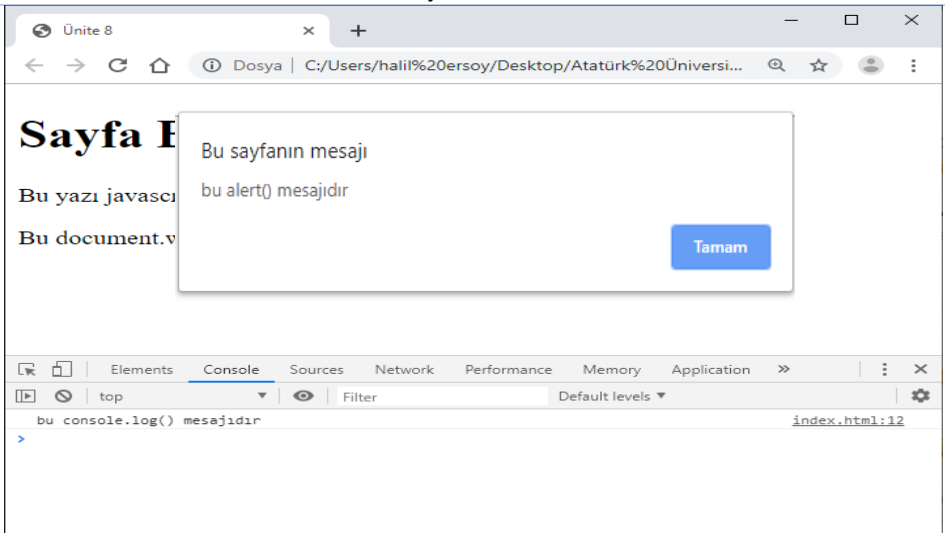
Ünitenin devamındaki konularda birçok işlem sonucunda ortaya çıkacak çıktıların görülebilmesi için aşağıdaki 3 basit komut ile *yazdırma* işlemi yapılacaktır. Bunlar;

- alert();
- document.write();
- console.log();

komutlarıdır.

Yukarıdaki 3 komutun çalışmasını gösteren örnek uygulamayı aşağıdaki kodlar ve çıktıda görebilirsiniz.

Tablo 10.3. Mesaj Veren 3 Basit Komut Ve Çıktıları

HTML ve Javascript Komutları	
<pre> <body> <h1>Sayfa Başlığı</h1> <p id="yazi1">Bu bir paragraftır.</p> <script> document.getElementById("yazi1").innerText="Bu yazı javascript ile değiştirilmiştir"; console.log("bu console.log() mesajıdır"); document.write("Bu document.write() mesajıdır"); alert("bu alert() mesajıdır"); </script> </body> </pre>	
Web Sayfası Görüntüsü	
 <p>The screenshot shows a web browser window with a single tab titled 'Ünite 8'. The address bar shows the file path 'C:/Users/halil%20ersoy/Desktop/Atatürk%20Üniversi...'. The page content includes a heading 'Sayfa I', a paragraph 'Bu yazı javasc...', and another paragraph 'Bu document.v...'. A small alert box is displayed in the center of the page with the text 'Bu sayfanın mesajı bu alert() mesajıdır' and a 'Tamam' button. The browser's developer tools are open, showing the 'Console' tab with the message 'bu console.log() mesajıdır'.</p>	

Alert() komutunun sonucunda açılacak küçük pencere, farklı tarayıcılarda farklı görünecektir, bu normaldir.

Kodlardaki **alert()** fonksiyonu, parantezi içine yazdığınız metin ifadelerini veya değişkenlerin değerlerini tarayıcı ekranına açılacak yeni küçük bir pencere içerisine yazar. Bu pencere, üzerindeki “Tamam” butonuna basılıncaya kadar bekler ve kapanmadığı sürece sayfada başka bir işlem yapılmasına **izin vermez**.

Komutlardan **document.write()**, parantezleri içerisine yazacağınız açık metinleri veya değişkenlerin değerlerini **çalıştığı yere** HTML içeriği olarak yazar. Benzer bir komut da **document.writeln()** komutudur. Önceki komuttan tek farkı HTML sayfasının içerisine eklenen yazının sonuna **satır sonu karakteri** eklemek ve kendisinden sonra gelen yazıların **kodda** alt satırdan başlamasını sağlamaktır. Ancak unutulmamalıdır ki HTML **kodlarında** alt alta yazılmış ifadeler, tarayıcı tarafından **ekrana** alt alta yazılmaz! Bunun için HTML etiketlerinden **
** ifadesinin eklenmesi gerekir.

Son olarak **console.log()** komutu, ekrana değil, tarayıcının (eğer açıksa) **geliştirici seçeneklerindeki konsol ekranına** çıktı yazmak üzere kullanılır. Parantezleri içerisine metin ya da değişken yazılarak konsol ekranına çıktı gönderilebilir. Bu komut daha çok geliştiricilerin birtakım değerleri veya işlemleri **takip etmek** için tercih ettikleri yoldur. Bu sayede sonradan silinecek birtakım çıktıların sayfanın içeriğinde yer alarak unutulması riskini yok etmiş olurlar.



Örnek

- Alert, document.write ve console.log komutları içerisinde birden fazla değer ve değişkeni birlikte yazabilirsiniz. Bunun için yazıları tırnak içersinde, sayısal değerleri olduğu gibi ve değişkenleri de olduğu gibi aralarına + işaretini koyarak birleşik şekilde kullanabilirsiniz.
- alert("Sonuç = " + 2 + " olur.");
- ÇIKTI: Sonuç 2 olur.

Açıklama Satırları

Geliştiriciler, kodlama işlerini yaparken kendileri veya başka geliştiricilere hatırlatma yapmak, bilgi vermek ya da geçici olarak bazı komutları devre dışına almak için kodlarına **açıklama** satırları eklerler. Bunu;

- Eğer *tek bir satır* açıklama satırı eklenecek ise, başına **//** sembollerini yazarak;
- Eğer *birden fazla satır* toplu hâlde açıklama olarak eklenecek ise başına **/*** ve sonuna ***/** sembolleri koyarak yapabilirler.

Açıklama satırları **çalıştırılmaz** ancak kodlarla birlikte kaydedilebilir. Aşağıdaki örnekteki açıklama satırlarını ve programın çıktısını inceleyiniz.

Tablo 10.4. Açıklama Satırları Ve Çıktısı

Javascript Komutları
<pre>document.write("<p>satır 0</p>"); // aşağıdaki bazı satırlar açıklama olarak işaretlenmiştir // ve çalışmazlar // document.write("<p>satır 1</p>"); document.write("<p>satır 2</p>"); /* document.write("<p>satır 3</p>"); document.write("<p>satır 4</p>"); */ document.write("<p>satır 5</p>");</pre>



Geçici olarak çalışmasını askıya almak istediğiniz komutları, satır başına **//** işareti koyarak açıklama satırı hâline getirebilirsiniz.

satır 0

satır 2

satır 5

DEĞİŞKENLER

Her programlama dilinde olduğu gibi, Javascript'te de değişken tanımlama ve kullanma önemli bir gerekliliktir. *Değişken*, programın çalışması sırasında birtakım değerleri (sayı, yazı, tarih, vb.) bilgisayarın hafızasında (RAM) saklamaya yarayan hafıza alanlarına verilen isimlerdir. *Değişken adı* kullanıcı tarafından belirlenir. Değişken tanımlamak için *var* komutu kullanılır.



Değişken isimlendirme kuralları bir önceki ünite de anlatılmıştır.

Tablo 10.5. Değişken Tanımlama

Javascript Komutları
<pre> <script> var a; var b=50; var c=3.14; var d="merhaba"; var e="dünya"; f=80; document.write("a=" + a + "
"); document.write("b=" + b + "
"); document.write("c=" + c + "
"); document.write("d=" + d + "
"); document.write("e=" + e + "
"); document.write("f=" + f + "
"); </script> </pre>
Web Sayfası Görüntüsü
<pre> a=undefined b=50 c=3.14 d=merhaba e=dünya f=80 </pre>

Yukarıdaki örnekte toplam 6 adet değişken tanımlanmıştır. Ardından değişkenlerin değerleri ekrana yazılmıştır. Yukarıdaki tanımlamalar ve çıktılar incelendiğinde aşağıdaki değişken tanımlama kuralları anlaşılabilir:

Tablo 10.6. Değişken Tanımlama Kuralları

Kural	Örnek ve Çıktı
1. Değişkenler var komutu ile tanımlanır, istenirse tanımlama sırasında ilk değer verilebilir.	<pre> var b; b=50; YA DA var b=50; </pre>
2. İlk değeri verilmeyen değişkenlerin değeri "undefined" olur.	<pre> var a; document.write(a); → undefined </pre>
3. Aynı var ifadesi ile birden çok	<pre> Var a,b,c=30; document.write(a); → undefined </pre>

değişken tanımlanabilir. Yandaki örnekte sadece c değişkeninin değeri 50 olur, diğerlerinin "undefined" olur.	document.write(b); → undefined document.write(c); → 50
4. Var komutu ile aynı isimde bir başka değişken tekrar tanımlanabilir, bu durumda önceki değişken yok edilmiş olur.	var a=20; document.write(a); → 20 var a = 75; document.write(a); → 75
5. Aynı satırda birden fazla değişkene aynı ilk değer verilebilir.	var a=b=c=40; document.write(a); → 40 document.write(b); → 40 document.write(c); → 40
6. Bir <script> blokunda tanımlanan değişken, bu blok kapandıktan sonra açılacak bir sonraki script blokunda da kullanılabilir.	<scrip> var a=30; </script> <p>yazı</p> <scrip> document.write(a); → 30 </script>



Değişkenleri kullanacakları fonksiyon ya da blok içerisinde ilk sırada tanımlamak iyi bir alışkanlıktır.

Görüldüğü üzere, değişkenler konusunda Javascript oldukça esnekler. Hatta bir değişken *var* kelimesi kullanılmadan da ilk değer ataması ile tanımlanmış olur. Yukarıda Tablo 10.5'teki "f" değişkeni buna bir örnektir. Ancak bu durumda "f" değişkeni *global* bir değişken olacaktır, bu nedenle bu şekilde kullanım önerilmez. Değişkenlerin *global* veya *lokal* olması, ileriki bölümlerde *Fonksiyonlar* başlığında açıklanacaktır.

Bu esnekliğe rağmen, geliştiricilerin sayfada kullanacakları değişkenleri diğer Javascript komutlarından önce *var* komutu ile tanımlamaları önerilir.

Değişkenler tanımlanmadan, herhangi bir işlemde kullanılamazlar. Aşağıdaki örnekte *a* ve *b* değişkenleri tanımlanmış ancak *c* değişkeni tanımlanmadığı için *yazım hatası* oluşacaktır.

Tablo 10.7. Tanımlanmamış Değişken Kullanılamaz.

Javascript Komutları
<pre> <script> var a=50; document.write(a); → 50 b=40; document.write(b); → 40 document.write(c); → HATA !! </script> </pre>

Değişken ve Veri Türleri

Değişkenlerin içerisinde saklanacak veri çeşidine göre *türü* vardır. Farklı türdeki değişkenleri tanımlamak için yine *var* ifadesi kullanılır, ancak bundan sonra değişkene atanacak *ilk değer*in *türüne göre*, değişken türü belirlenmiş olur.

Tablo 10.8. Farklı Veri Türleri

Javascript Komutları
<pre> <script> var a; // tür : belirsiz (undefined) a=50; // tür: sayı (number) var b=3.14; // tür: sayı (number) var c="merhaba"; // tür: metin (string) var d= 123e5; // tür: bilimsel sayı (123 x 10⁵) (number) var e='t'; // tür: metin (string) </script> </pre>

Yukarıdaki Tablo 10.8.'de görüleceği üzere farklı biçimlerde değerleri tutan değişkenlerin *number* veya *string* türünde olduklarını gördük. Örnekteki "a" değişkeni, ilk değer olarak 50 değerini alana kadar türü *belirsiz (undefined)* olur.

Tür Dönüşümü

Değişkenlere farklı türde değer vererek o değişkenin *türünü değiştirmiş* oluruz. Aşağıdaki örnekte aynı değişkene atanan farklı değerlerin tür değişikliği etkisini görebilmek için, *typeof()* komutu ile değişkenin türünü ekrana yazdırıyoruz.

Tablo 10.9. Farklı değerler ile tür dönüşümleri ve *typeof* komutu

Javascript Komutları
<pre> var a=3.14; document.write(typeof(a)); → number a = "merhaba"; document.write(typeof(a)); → string a=123e5; document.write(typeof(a)); → number a='r'; document.write(typeof(a)); → string a=30; document.write(typeof(a)); → number </pre>



True ve false kelimeleri Javascript'in anahtar kelimelerindendir, metin türü ile karıştırılmamalıdır.

Diğer Veri Türleri

Javascript'te *number* (sayı) ve *string* (metin) türleri temel veri türleridir. Sayı ve metin türündeki değerlerin grup hâlinde kullanılmasıyla farklı veri türleri ortaya çıkmıştır. Bunlar *boolean* (ikilik) ve *object* (nesne) türleridir.

Boolean veri türü, değer olarak *true* ve *false* ifadeleri tutan değişkenler için kullanılır. Bu tür değişkenler, herhangi bir kıyaslama işleminin sonucunu *doğru (true)* ya da *yanlış (false)* olarak tutabilirler.

Tablo 10.10. Boolean Veri Türü

Javascript Komutları
<pre> var a= 4; var b= 8; var c= b > a; document.write("c = " + c + " tür: "+ typeof(c)); </pre>
Çıktı
c = true tür: boolean

Object veri türü, aynı veya farklı türdeki birden çok değişkenin gruplanması gerektiğinde başvuru bir türdür. Bu türe yönelik ilk örnek **dizilerdir (array)**. Aşağıdaki örnekte dizi tanımlaması ve kullanımı gösterilmiştir. Ancak daha detaylı dizi kullanımı bir sonraki ünite **Diziler** başlığı altında anlatılmıştır.

Tablo 10.11. Dizilerin Türü Object'tir.

JavaScript Komutları
<pre>var a=[10,20,30,40,50]; var c= a[0] + a[1]; // 20 ve 30 toplanıyor document.write("c = " + c + " tür: "+ typeof(c)); document.write("a[0] = " + a[0] + " tür: "+ typeof(a[0])); document.write("a dizisi türü: "+ typeof(a));</pre>
Çıktı
<pre>c = 50 tür: number a[0] = 10 tür: number a dizisi tür: object</pre>

Farklı türleri gruplayarak da **object** türünde değişkenler (**nesneler**) yaratılabilir.

Tablo 10.12. Farklı Türlerden Oluşan Nesne Türü

JavaScript Komutları
<pre>var insan = {ad: "ali", soyad:"tok",yas:50, gozRengi:"mavi"}; document.write("insan.ad=" + insan.ad + " tür:" + typeof(insan.ad)); document.write("insan tür:" + typeof(insan));</pre>
Çıktı
<pre>insan.isim=ali tür: string insan tür: object</pre>

Yukarıdaki son iki örnekte görüleceği üzere, **nesne** türündeki değişkenlerin (a ve insan) kendi türleri **object** iken, nesnenin parçalarının her biri kendi temel türündedir (a[0] elemanı **number**, insan.isim elemanı **string** türündedir).

Nesne (object) veri türü detayları bu kitabın kapsamına alınmamıştır.

Değişken Türlerinin Önemi

Yukarıda tanıtılan değişken türlerinden uygun olanı seçmemiz gerekir. Bunun iki sebebi vardır:

- Bazı veri türleri hafızada (kullanıcının bilgisayarının RAM belleğinde) diğerlerine göre daha fazla yer işgal eder. Gereksiz yere büyük veri türleri seçmek, kullanıcının bilgisayarını gereksiz yere yoracaktır.
- Verinin türüne göre, veri üzerinde yapılacak işlemler farklılık gösterir, uygun olmayan türlerde yapılan işlemler yanlış sonuçlara neden olur.

Özellikle ikinci sebebi aşağıdaki örnek ile gösterelim.



Diziler Javascript'te çok kullanıma rağmen tür olarak nesne (object) sınıfındadır.

Tablo 10.13 Toplama İşleminin Farklı Türlerdeki Etkisi

JavaScript Komutları
<pre>var a=10, b=20, c; var x="10", y="20", z; c= a + b; z = x + y; document.write("c=" + c + "
"); document.write("z=" + z + "
");</pre>
ÇIKTI
<pre>c=30 c=1020</pre>



Toplama ve bitleştirme operatörü olan + sembolü, sonucu doğru vermesi için doğru türdeki değişken ya da değerler ile çalıştırılmalıdır.

Toplama işlemi için kullanılan artı operatörü (+), işlem yapmadan önce kendisinin sağındaki ve solundaki değer ya da değişkenlerin türlerine bakar ve türe göre işlem yapar. Eğer her iki taraf da *sayı* ise *matematiksel toplama* yapar (10 + 20 = 30). Eğer taraflardan en az bir tanesi *metin* (string) ise toplama yerine *bitleştirme* yapar ("10" + " 20" = "1020"). Bu tür işlemlerin beklendiği gibi çalışması için işleme giren değişken ya da değerlerin türleri doğru oluşturduğumuzda emin olmalıyız.

Undefined ve Null Veri Değerleri

Değişkenler henüz ilk değerini almadıklarında *tür* ve *değer* olarak *undefined* şeklinde tanımlanır. Herhangi bir değişken açık biçimde *undefined* değerine atanarak *hem içerisindeki değer* silinmiş olur, *hem de türü undefined* olarak sıfırlanmış olur.

Tablo 10.14. Undefined Değeri

JavaScript Komutları
<pre>var m; document.write("m=" + m + " tür:" + typeof(m) + "
"); m=100; document.write("m=" + m + " tür:" + typeof(m) + "
"); m=undefined; document.write("m=" + m + " tür:" + typeof(m) + "
"); m="merhaba"; document.write("m=" + m + " tür:" + typeof(m) + "
");</pre>
ÇIKTI
<pre>m=undefined tür: undefined m=100 tür: number m=undefined tür: undefined m=merhaba tür: string</pre>

Javascript'te *null* değeri ise nesnelerin değerini yok etmek için kullanılır. *Null* değerini alan nesnenin içindeki *değer silinirken*, türü yine *object* olarak kalır. Eğer nesnenin hem değerini hem de türünü sıfırlamak istenirse *undefined* değerine atanmalıdır.

Tablo 10.15. Undefined Ve Null Değerleri

JavaScript Komutları
<pre>var m=[10,20,30,40]; document.write("m=" + m + " tür:" + typeof(m) + "
"); m=null;</pre>

```
document.write("m=" + m + " tür:" + typeof(m) + "<br><br>");

var m=[10,20,30,40];
document.write("m=" + m + " tür:" + typeof(m) + "<br>");
m=undefined;
document.write("m=" + m + " tür:" + typeof(m) + "<br>");
```

ÇIKTI

```
m=10,20,30,40   tür: object
m=null           tür: object

m=10,20,30,40   tür: object
m=undefined      tür: undefined
```

OPERATÖRLER

Değişkenlerin değerlerini değiştirmeye veya değerlerini kontrol etmeye yarayan komutlara operatör denir. Operatörler genellikle sembollerden oluşur. Yaptıkları işler bakımından operatörleri 3 sınıfta görelim:

- Aritmetik operatörleri,
- Karşılaştırma operatörleri,
- Mantık operatörleri.



Aşağıdaki 3 komut eşittir:
a=a+1;
a+= 1;
a++;

Aritmetik Operatörleri

Sayısal değişkenler üzerinde yapılabilecek aritmetik işlemlerini gerçekleştiren aritmetik operatörleri aşağıdaki tabloda verilmiştir. Tablodaki işlemler *a*, *b* ve *c* değişkenlerinin ilk satırdaki tanımlamalarına göre yapılmıştır.

Tablo 10.16. Aritmetik Operatörleri

Operatör	Anlamı	Örnek var a=10, b=20, c;	Sonuç c
+	Toplama	c = a + b;	30
-	Çıkarma	c = a - b;	-10
*	Çarpma	c = a * b;	200
/	Bölme	c = a / b;	0.5
%	Bölümünden kalan (mod)	c = a % b;	10
++	Sayıyı bir arttırmak	c = 5; c++; veya ++c;	6
--	Sayıyı bir azaltmak	c = 5; c--; veya --c;	4

Yukarıdaki operatörlerden *artırma (++)* ve *azaltma (--)* operatörleri, kullanım şekline göre artırma ve ya eksiltme işlemini, atamadan *önce* ya da *sonra* yapar. Aşağıdaki şekli inceleyelim.

Tablo 10.17. Artırma Ve Azaltma Operatörünün Farklı Kullanımı

İşlemler	Çıktı
var a=5, b=1; var sonuc; sonuc = a + b++;	

document.write("sonuc=" + sonuc + "</br>"); document.write("a=" + a + "</br>"); document.write("b=" + b + "</br>");	sonuc=6 a=5 b=2
var a=5, b=1; var sonuc; sonuc = a + ++b; document.write("sonuc=" + sonuc + "</br>"); document.write("a=" + a + "</br>"); document.write("b=" + b + "</br>");	sonuc=7 a=5 b=2

Yukarıdaki örnekte **sonuc** değişkeninin değerinin farklı olmasının nedeni **artırma** (++) operatörünün değişkenin **solunda** ya da **sağında** yazılmasından kaynaklanmaktadır. Eğer ilk örnekteki gibi değişkenin **sağına yazılırsa** (b++), toplama işlemine değişkenin artmamış değeri (1) girer ve sonuç 6 değerine ulaşır. Ardından **b** değişkeni 1 artar ve 2 olur. İkinci örnekte ise artırma operatörü değişkenin soluna (a++) yazılmıştır. Bu durumda toplama işleminden önce artırma yapılır (**b** değişkeni 2 olur) ve sonuç 7 olarak hesaplanır.

Bazı aritmetik işlemler, **değişkenin kendisini kullanarak** işlem yapıyorsa, bu işlemleri aşağıdaki tablodaki gibi **kısaltarak** kullanmak mümkündür.

Tablo 10.18. Aritmetik İşlemlerin Kısaltılmış Hâli

Operatör	Anlamı	Örnek ve Alternatifi var a=10, b=20;	Sonuç
+=	Kendisiyle bir değeri topla	a += b; // a=a+b;	a = 30
-=	Kendisinden bir değeri çıkar	a -= b; // a=a-b;	a = -10
*=	Kendisiyle bir değeri çarp	a *= b; // a=a*b;	a = 200
/=	Kendisini bir değere böl	a /= b; // a=a/b;	a = 0,5
%=	Kendisinin bir değere bölümünden kalanı (mod) bul	a %= b; // a=a%b;	a = 10

Karşılaştırma Operatörleri

Değişken ya da değerlerin birbiri ile kıyaslanması için kullanılan operatörlere **karşılaştırma operatörleri** denir. Bu operatörler belirli noktalarda karar vermeye ve gerekirse programın akışını değiştirmek için, az sonra ayrı bir başlık ile anlatılacak olan, koşul yapıları için gereklidir. Karşılaştırma operatörlerinin sonucunda **true (doğru)** ya da **false (yanlış)** değerleri üretilir. Bu değerler metin ya da sayı değildir, türü **boolean**'dir. Javascript'teki karşılaştırma operatörlerini aşağıdaki tabloda görebilirsiniz.

Tablo 10.19. Karşılaştırma Operatörleri

Operatör	Anlamı	Örnek var a=10;	Sonuç
==	Değerleri birbirine eşitse	a == 2	false
		a == 10	true
!=	Değerleri eşit değilse	a != 9	true
===	Değerleri ve türleri eşitse	a === "10"	false
		a === 10	true



Kısaltma operatörlerini yazarken, iki sembol arasına boşluk bırakmak hata olacaktır.
a += 3; // doğru
a + = 3; // hatalı



En çok yapılan hata, atama operatörü (=) ile eşitlik kontrolü operatörünün (==) karıştırılmasıdır.

!=	Değerleri ve türleri eşit değilse	a != "10"	true
		a != 10	false
>	Değeri büyükse	a > 8	true
<	Değeri küçükse	a < 8	false
>=	Değeri büyük veya eşitse	a >= 10	true
<=	Değeri küçük veya eşitse	a <= 9	false

Yukarıdaki operatörlerden === ve !== operatörleri, sağındaki ve solundaki değişken veya değerlerin *hem değerine hem de türüne göre* karar verir. Buna göre değerleri aynı, ancak türleri farklı değişkenlerin karşılaştırılmasındaki sonuçları aşağıda görebilirsiniz.

Tablo 10.20. Değer Karşılaştırma Ve Tür Karşılaştırma Operatörleri

Javascript Komutları			
<pre>var a=50; // tür: number var b=50.0; // tür: number var c="50"; // tür: string</pre>			
Örnek ve Çıktılar			
Karşılaştırma	Sonuç	Türler	Karşılaştırma Ölçütü
a == b	true	number ve number	Sadece değer eşit mi?
a == c	true	number ve string	
b == c	true	number ve string	
a === b	true	number ve number	Hem değer hem tür eşit mi?
a === c	false	number ve string	
b === c	false	number ve string	
a != b	false	number ve number	Sadece değer farklı mı?
a != c	false	number ve string	
b != c	false	number ve string	
a !== b	false	number ve number	Hem değer hem tür farklı mı?
a !== c	true	number ve string	
b !== c	true	number ve string	

Mantık Operatörleri

Birden fazla karşılaştırma işleminin birlikte değerlendirilmesi ve tek bir doğru (*true*) ya da yanlış (*false*) üretilebilmesi için mantık operatörleri kullanılır. Bu operatörler tıpkı karşılaştırma operatörleri gibi, koşul ve döngü yapılarında sıklıkla kullanılırlar. Mantık operatörleri aşağıdaki tabloda verilmiştir.

Tablo 10.21. Mantık Operatörleri

Operatör	Anlamı	Örnek var a=10, b=20;	Sonuç
&&	ve	a <= 10 && b < 100	true
	veya	a == 5 b == 5	false
!	değil	!(a > b)	false



&& ve ||

operatörlerinin tek sembolik hâlleri de vardır (& ve |). Ancak onlar farklı işlevleri yerine getirmektedirler ve bu kitabın kapsamında değildir.

Mantık operatörleri iki ya da daha fazla karşılaştırma ifadesini karşılaştırabilir. Buna göre;

- *Ve (&&)* operatörünün *her iki yanında true* değeri varsa sonuç *true*, bunun dışındaki tüm durumlarda sonuç *false* olur.
- *Veya (||)* operatörünün yanındaki ifadelerden *en az bir tanesi true* ise sonuç *true*, tüm ifadeler *false* ise sonuç *false* olur.
- *Değil (!)* operatörü sağ tarafına *tek ifade* alır ve bu ifadenin *boolean* cinsinden *tersini* (*true* ise *false*, *false* ise *true*) üretir.

Operatörlerde İşlem Önceliği

Yukarıdaki operatörler kendi başlarına iki ya da tek değişken ile işlem yaparlar. Buna karşın bazı işlemler *birden fazla operatörün* sırayla kullanılmasını gerektirebilir. Bu durumda farklı operatörlerin *yan yana* yazılarak sonucun üretilmesi gerekebilir. Böyle durumlarda yan yana yazılmış operatörlerin hangisinin *öncelikle* çalışacağı bilinmelidir. Aşağıdaki tabloda *operatörlerin işlem önceliği* belirtilmiştir.

Tablo 10.22 Operatör İşlem Önceliği Ve Yönü

Operatör	İşlevi	İşlem Önceliği	İşlem Yönü
++	Değeri 1 artırma	Birinci	Sağdan sola (←)
--	Değeri 1 azaltma	Birinci	Sağdan sola (←)
-	Eksi değer	Birinci	Sağdan sola (←)
!	Değil	Birinci	Sağdan sola (←)
*, /, %	Çarpma, bölme, mod alma	İkinci	Soldan Sağa (→)
+, -	Toplama, çıkarma	Üçüncü	Soldan Sağa (→)
+	Bitiştirme	Üçüncü	Soldan Sağa (→)
<, <=	Küçük, küçük veya eşit	Dördüncü	Soldan Sağa (→)
>, >=	Büyük, büyük veya eşit	Dördüncü	Soldan Sağa (→)
==	Eşit	Beşinci	Soldan Sağa (→)
!=	Farklı	Beşinci	Soldan Sağa (→)
===	Tür ve değer aynı	Beşinci	Soldan Sağa (→)
!==	Tür ve değer farklı	Beşinci	Soldan Sağa (→)
&&	Ve	Altıncı	Soldan Sağa (→)
	Veya	Yedinci	Soldan Sağa (→)
=	Atama	Sekizinci	Sağdan sola (←)
+=, -=, *=, /=, %=	Aritmetik işlem ve atama	Sekizinci	Sağdan sola (←)

Yukarıdaki tabloda operatörler işlem önceliğine göre sıralanmıştır. Farklı operatörler yan yana yazıldığında yukarıdaki *önceliğe* göre işlem yapmaya başlanır. Eğer önceliği eşit olan operatörler yan yana yazılırsa, o zaman *işlem yönüne göre* öncelik verilir.



İşlem önceliği aynı düzeyde olan operatörlerin işlem yönüne bakılmalıdır.

İşlem önceliği değiştirmek istenirse, öncelik verilmek istenen operatör ve o operatör için gerekli değişken ve değerler *parantez ()* içerisine alınır.

Tablo 10.23. İşlem Önceliği Örneği

Javascript Komutları ve Örnek Çıktılar			
var a=5, b=1, c=4;			
var sonuc;			
sonuc = a + b / c;	→ 5.25	// öncelik bölmede	
sonuc = (a + b) / c;	→ 1.5	// öncelik toplamada	
sonuc = a / c / 2;	→ 0.625	// öncelik soldan sağa	

Mantık ve karşılaştırma operatörleri, eğer aritmetik işlemler ile birlikte kullanılacak ise, *öncelik* aritmetik işlemlerdir.

Tablo 10.24 İşlem Önceliği Ve Yönü Örneği

Javascript Komutları ve Örnek Çıktılar			
var a=5, b=1, c=4;			
var sonuc;			
sonuc = a > b && a > c;	→ true	//öncelik karşılaştırmada (>)	
sonuc = a % 2 == 0;	→ false	//öncelik mod almada (%)	
sonuc = a >= 2 - 1 && b >= 0 && c >= 0;	→ true		
// öncelik çıkarmada (2 - 1) , sonra karşılaştırmalarda (>=),			
// ardından soldan sağa mantık operatörleri (&&)			



İşlem önceliğini kontrol etmek için istenirse birden fazla parantez iç içe yazılabilir.
x=((a+1)/b)*c;



İşlem önceliğini kontrol etmek için sadece düz parantezler kullanılır. Farklı parantez türleri hataya neden olur.
x=[(a+1)/b]; // HATA
x=((a+1)/b); // Doğru



Bireysel Etkinlik

- Aşağıdaki komutların sonucunu önce kağıt üzerinde hesaplayınız. Ardından bilgisayarda kodlayıp sonuçları karşılaştırınız.
- sonuc = a * a + b * b;
- sonuc = a * (a + b) * b;
- sonuc = a++ * b;
- sonuc = a + 1 % 10;

KOŞUL KOMUTLARI VE KOŞUL OPERATÖRÜ

Tüm programlama dillerinde komutların çalışma sırası yukarıdan aşağıyadır. Buna programın akışı denilebilir. Bu akış bazı durumlarda değişmek zorundadır. Akışı belirli koşullara göre değiştiren komutlara *koşul komutları* denir. Javascript'te iki adet koşul komutu vardır:

- if
- switch

Bu iki komuta ek olarak, koşula göre değer aktarma işlemini değiştiren bir koşul operatörü vardır:

- ?:

If Komutu

Bu komut, kendisi içerisinde yazılan kıyaslama işleminin sonucunda *true* değeri üretilirse devamındaki komutları çalıştırır. Sonuç *false* ise devamındaki komutları çalıştırmaz.

Tablo 10.25. If Komutu Ve Çıktısı

JavaScript Komutları
<pre>var a=5, b=6; if(a%2==0) // koşul false'dur document.write("a değişkeni çift sayıdır"); // çalışmaz if(b%2==0) // koşul true'dur document.write("b değişkeni çift sayıdır"); // çalışır</pre>
ÇIKTI
b değişkeni çift sayıdır

Yukarıdaki örnekte, *if* komutundan sonraki parantez içerisine bir *koşul yazmak* mecburidir. Bu koşul ifadesi birden çok karşılaştırma ve mantık operatörünü içerebilir. Parantez içerisindeki tüm kıyaslamalar, varsa aritmetik işlemler ve mantık karşılaştırmaları tamamlandıktan sonra ortaya çıkacak olan *true/false* değerine göre işlem yapılır.

If komutunun koşulu *doğru (true)* ise kendisinden sonraki bir komut çalıştırılır, *yanlış (false)* ise yine kendisinden sonraki bir komut çalıştırılmaz. Ancak bazı durumlarda bu koşula bağlı olarak birden çok komut veya komut satırı benzer şekilde çalışmalı ya da çalışmamalıdır. Bu durumda *if* komutunun etkisini birden çok komuta taşımak için, ilgili komutlar *küme parantezleri { }* içerisine yazılmalıdır.

Tablo 10.26. Küme Parantezleri İle Blok Yaratmak

JavaScript Komutları
<pre>var pi=3.1417, a=20; if(pi > 3.0 && a%2==0){ // koşul true'dur document.write("pi değeri 3ten büyüktür"); // çalışır document.write("a değişkeni de çift sayıdır"); // çalışır }</pre>
ÇIKTI
Pi değeri 3ten büyüktür a değişkeni çift sayıdır

If komutunun 3 farklı farklı biçimi vardır. Bunlar aşağıdaki gibidir.

Tablo 10.27. If Komutunu 3 Farklı Biçimi Ve Açıklaması

If Komutu Çeşitleri		
if(koşul) komutlar;	if(koşul) komutlar1; else komutlar2;	if(koşul1) komutlar2; else if(koşul2) komutlar2; else if(koşul3) komutlar3; else komutlar4;



if...else if komutunda, "else" ile if'i bitişik yazmak yanlıştır. else if(koşul)//doğru else if(koşul) //hata

Koşul <i>true</i> ise, <i>komutlar</i> çalıştırılır	Koşul <i>true</i> ise sadece <i>komutlar1</i> çalıştırılır, <i>false</i> ise sadece <i>komutlar2</i> çalıştırılır.	Sırası ile ilk koşuldan itibaren kontrol edilir, sonucu <i>true</i> olan <i>koşul</i> bulunursa, sadece onun altındaki komutlar çalıştırılır ve devamındaki koşullara <i>bakılmadan</i> if komutu sonlandırılır. Eğer hiçbir koşul <i>true</i> değil ise, o zaman <i>else</i> komutunun altındaki komutlar çalıştırılır.
---	--	--

Yukarıdaki tabloda belirtilen “*komutlar*” ifadesi, eğer birden fazla komut çalıştırılmak isteniyorsa, *küme parantezi* içerisine alınmış komut blokları ile değiştirilmelidir. Eğer tek komut çalıştırılacak ise küme parantezine almaya gerek yoktur, ancak alınırsa da hata olmaz.

Tablo 10.28. İf.. Else İf...Else Örneği

JavaScript Komutları
<pre>var a=5, b=10, c=4; if(a%2==0) // false document.write("a değişkeni çift sayıdır");// çalışmaz else if(b%2==0) // true document.write("b değişkeni çift sayıdır");// çalışır else if(c%2==0) // kontrol edilmez document.write("c değişkeni çift sayıdır");// çalışmaz else document.write("hiç biri çift değildir");// çalışmaz</pre>
ÇIKTI
b değişkeni çift sayıdır

Yukarıdaki örnekte, ikinci koşul (b%2==0) *true* olduğu için, ikinci komut çalışır. Bu noktadan sonra artık *kontrol yapılmaz* ve *if* komutu sonlanır. Dikkatlice incelendiğinde, aslında üçüncü komutun da *true* olduğu görülebilir, ancak *if...else if...else* yapısında, herhangi bir koşul doğru ise sadece ona bağlı komutlar çalıştırılır ve diğer koşullar kontrol edilmez.

Bu biçimde kullanılan *if* komutunda istenilen sayıda *else if()* *koşulu* ve komutları eklenebilir. İstenirse *else* ifadesiyle, önceki tüm koşulların doğru olmadığı durumlarda çalışması için komut veya komutlar eklenebilir, ancak *else* ifadesi mutlaka *en sonda* ve *sadece bir defa* yazılmalıdır. Eğer istenirse, *else* ifadesi hiç yazılmayabilir.

Aşağıdaki örnekte *birbirinden bağımsız üç if komutu* görülmektedir. Bu şekilde bağımsız olan *if* komutlarında, tüm koşullar kontrol edilir ve doğru olan tüm koşulları için ilgili komutlar çalıştırılır.

Tablo 10.29. Bağımsız Üç İf Komutu

JavaScript Komutları
<pre>var a=5, b=10, c=4; if(a%2==0) // false document.write("a değişkeni çift sayıdır");// çalışmaz if(b%2==0) // true document.write("b değişkeni çift sayıdır");// çalışır</pre>



If komutu, içerisinde *elseif* ve *else* içermek zorunda değildir.

```
if(c%2==0) // true
    document.write("c değışkeni çift sayıdır");// çalışır
```

ÇIKTI

b değışkeni çift sayıdır
c değışkeni çift sayıdır

If komutundaki *komutlar* eğer istenirse, *if* satırının devamına da yazılabilir. Ancak bu *okunabilirliği* zorlaştırabilir, o nedenle yukarıdaki örneklerdeki gibi komutların if satırından sonra *alt satırda* yazılması ve satır başından *girinti* yapılarak hizalanması daha doğru olacaktır.

Tablo 10.30. Okunabilirlik Açısından Karmaşık Ve Düzenli Yazım Biçimleri

Karmaşık yazım şekli	Önerilen hizalı yazım şekli
<pre>var a=5, b=10, c=4; if(a % 2 == 0) b = 2 * a + c; else b = a / 2 +c;</pre>	<pre>var a=5, b=10, c=4; if(a % 2 == 0) b = 2 * a + c; else b = a / 2 +c;</pre>

Switch Komutu

Program akışının bir koşula göre değışmesi için kullanılan bir diğerkomut ise *switch* komutudur. Komutun yazılış şekli aşağıdaki şekilde gibidir.

Tablo 10.31. Switch Komutu Örneğı

Javascript Komutları
<pre>Var gun=3; var gunAdi; switch(gun) { case 1: gunAdi = "Pazartesi"; break; case 2: gunAdi = "Salı"; break; case 3: gunAdi = "Çarşamba"; break; case 4: gunAdi = "Perşembe"; break; case 5: gunAdi = "Cuma"; break; case 6: gunAdi = "Cumartesi"; break; case 7: gunAdi = "Pazar"; break; default: gunAdi = "belirsiz"; break; }</pre>



Her case yanına tek bir değerk yazılabilir.

```
document.write("Gün adı = " + gunAdi);
```

ÇIKTI

Gün adı = Çarşamba

Yukarıdaki *switch* komutunda, parantez içerisindeki değişkenin (*gun*) değeri (3) ile yukarıdan aşağıya doğru *case* ifadelerinin sağındaki değerler (1, 2,... vb.) karşılaştırılır. Eğer bu değer ile *case* değeri eşit ise, o *case* ifadesinden sonraki komutlar çalıştırılır. Case ifadesinden sonra, *switch* parantezi içerisindeki değişkenin değeri olabilecek değerler yazılmalıdır. Bunun için *istenilen sayıda case* ifadesi ve değeri yazılabilir.

Case ifadesinden sonra yazılacak komutların en sonuna ve bir sonraki case'den önce *break* komutu eklenmelidir. Break komutu ile devamındaki *case* komutlarının kontrol edilmesi ve çalıştırılması engellenir ve switch komutu sonlandırılır. Eğer *break* komutu yazılmaz ise değeri eşleşen *case* ifadesinin komutları çalıştırılır *ve devamındaki case* ifadesinin komutları da, *değeri eşleşmese bile*, çalıştırılır.

En sondaki *default* ifadesini yazmak şart değildir, ancak yazılacak ise *bir defa* yer almalıdır. *Default* ifadesi, var olan *case* değerlerinden hiç biri ile eşleşme olmuyorsa çalıştırmak istenilen komutlar için eklenir. Genellikle *default* ifadesi tüm *case* ifadelerinden sonra yazılır ancak bu şart değildir. Eğer araya yazılacak ise *break* komutu ile sonlandırılması gerekir.

Tablo 10.32. Switch Komutundaki Default İfadesi

Javascript Komutları

```
var gun=10;    // eşleşme olmazsa, default kısmı çalışır
var gunAdi;
switch(gun) {
  case 1:
    gunAdi = "Pazartesi";
    break;
  case 2:
    gunAdi = "Salı";
    break;
  case 3:
    gunAdi = "Çarşamba";
    break;
  case 4:
    gunAdi = "Perşembe";
    break;
  case 5:
    gunAdi = "Cuma";
    break;
  case 6:
    gunAdi = "Cumartesi";
    break;
  case 7:
    gunAdi = "Pazar";
    break;
  default:
    gunAdi = "belirsiz";
    break;
```



Switch komutu içerisindeki break komutu yazılmazsa söz dizim hatası olmaz, ancak mantık hatası olabilir.

```
}
document.write("Gün adı = " + gunAdi);
```

ÇIKTI

Gün adı = belirsiz

Eğer farklı değerler için aynı komutlar çalıştırılmak istenirse, *her değer* için bir *case* yazılır ancak *break* yazılmadan aşağıdaki gibi boş bırakılır.

Tablo 10.33. Birden Fazla Değerin Eşleşmesinin Sağlanması

Javascript Komutları
<pre>var gun=3; var gunAdi; switch(gun) { case 1: case 2: case 3: case 4: case 5: gunAdi = "hafta içi"; break; case 6: case 7: gunAdi = "hafta sonu"; break; default: gunAdi = "belirsiz"; } document.write("Gün adı = " + gunAdi);</pre>
ÇIKTI
Gün adı = hafta içi

Koşul Operatörü (?:)

Bir koşula bağlı olarak, iki farklı değerden birini seçen operatör, koşul operatörüdür (? :). Kullanımı aşağıdaki gibidir.

- Değişken = koşul ? değer1 : değer2;

Öncelikle *koşulun* sonucuna bakılır, sonuç *true* ise *değer1* olarak ifade edilen değer *Değişken'e* aktarılır. Eğer sonuç *false* ise *değer2* olarak ifade edilen değer *Değişken'e* aktarılır.

Tablo 10.34. Koşul Operatörünün Örneği

Javascript Komutları
<pre>var a = 6, sonuc; sonuc = a % 2 == 0 ? "çift" : "tek"; document.write("a değişkeni " + sonuc + " sayıdır");</pre>
ÇIKTI
a değişkeni çift sayıdır

Koşul operatörü ile yapılan işlem, *if..else* komutu ile yapılabilir, ancak daha kısa olduğu için tercih edilebilir.



?: operatörünün sonundaki noktalı virgül (;) unutulmamalıdır.

Tablo 10.35. Koşul Operatörü Ve İf...Else Komutunun Benzerliği

Koşul Operatörü ile Çözüm
<pre>var a = 6, sonuc; sonuc = a % 2 == 0 ? "çift" : "tek"; document.write("a değişkeni " + sonuc + " sayıdır");</pre>
If Else Komutu ile Çözüm
<pre>var a = 6, sonuc; if(a % 2 == 0) sonuc = "çift"; else sonuc = "tek"; document.write("a değişkeni " + sonuc + " sayıdır");</pre>



Özet

• Giriş

• Javascript, dil özellikleri bakımında C/C++ programlama dillerine benzer. Komutlar noktalı virgül (;) ile biter, bloklar küme parantezleri { } ile açılır ve kapanır. Eğer istenirse tüm komutlar aynı satıra yan yana yazılabilir. Ancak kodların anlaşılır olması için komutların alt alta, her satırda tek komut olacak şekilde yazılması ve satır başı girintilerinin uygulanması tavsiye edilir. Bu sayede kodlama aşamasında birçok olası hata fark edilebilir. Bu ünite, Javascript'te değişkenler, operatörler ve koşul ifadelerini ele alacağız.

• JAVASCRIPT VE HTML ETİKETLERİ İLİŞKİSİ

• Javascript dili ile HTML sayfalarındaki tüm etiketlerin özellikleri ve/veya içerikleri değiştirilebilir. Bunun için Javascript komutları içerisinde ilgili etikete document.getElementById() komutu ile erişilebilir. Komuttaki parantezler içerisinde yazılacak isim (id), etikete verilmiş "id" ismi olmalıdır.

• Üç komut ile farklı yerlere çıktı yazdırma işlemi yapılabilir.

• alert();

• document.write();

• console.log();

• Açıklama satırları // sembolleri ile başlar. Eğer birden fazla satır açıklama satırı yapılacak ise satırlar /* ve */ ifadeleri arasına yazılmalıdır.

• Değişkenler

• Javascript'te değişkenler mutlaka önceden tanımlanmalıdır. Tanımlama için var kelimesi kullanılır.

• Değişken isimlerinde bir önceki ünite, belirtilen isimlendirme kuralları geçerlidir.

• Değişken Türleri ve Değerleri

• Değişkenlere tanımlandıkları yerde ilk değer vermek tavsiye edilir. Bu sayede hem türü belirlenmiş olur hem de beklenmedik sonuçlar ortaya çıkmaz. Değişkenleri türü ilk alacakları değer türü ile belirlenir. Javascript'teki veri türleri sayı (number), metin (string), belirsiz (undefined), nesne (object) ve ikiliktir (boolean).

• Değişkenlerin veya açık değerlerin türlerini kod içerisinde anlamak için typeof() komutu kullanılır.

• Değişkenler henüz ilk değerini almadıklarında tür ve değer olarak undefined şeklinde tanımlanır. Herhangi bir değişken açık biçimde undefined değerine atanarak hem içerisindeki değer silinmiş olur, hem de türü undefined olarak sıfırlanmış olur. Bu değişkene daha sonra farklı türde bir değer verilerek türü değiştirilebilir.

• Null değeri ise nesnelerin (object türündeki değişkenlerin) değerini yok etmek için kullanılır. Null değerini alan nesnenin içindeki değer silinirken, türü yine object olarak kalır. Eğer nesnenin hem değerini hem de türünü sıfırlamak istenirse undefined değerine atanmalıdır.



Özet (devamı)

- **Operatörler**
- Değişkenlerin değerlerini değiştirmeye veya değerlerini kontrol etmeye yarayan komutlara operatör denir. Operatörler genellikle sembollerden oluşur. Yaptıkları işler bakımından operatörler 3 sınıfa ayrılabilir:
- **Aritmetik Operatörleri**
- Sayısal türdeki değişkenler ve / veya değerler ile yapılacak aritmetiksel işlemleri yapan operatörlerdir (+, -, *, /, %, ++, -- vb).
- **Karşılaştırma Operatörleri**
- Değişken ve/veya değerleri birbirine göre kıyaslayan ve sonuç olarak doğru (true) ya da yanlış (false) değerlerini üreten operatörlerdir (<, >, ==, <=, >=, != vb).
- **Mantık Operatörleri**
- Birden fazla karşılaştırma işleminin sonrasında tek bir doğru ya da yanlış kararı verebilmek için kullanılan operatörlerdir (&&, ||, !).
- **Operatörlerde İşlem Önceliği ve Yönü**
- Aynı satırda yan yana yazılan operatörler belirli sırada çalışırlar. Bu çalışma sırasına işlem önceliği denir. Eğer işlem önceliği aynı olan operatörler varsa o zamanda işlem yönüne göre (soldan sağa ya da tam tersi) işlem yapılır.
- İşlem önceliğini değiştirerek belirli işlemlere öncelik verilmek istenirse o işlem parantez içerisinde yazılmalıdır.
- **Koşul Komutları**
- Javascript'te iki adet koşul komutu vardır:
- **if Komutu**
- Bir koşula göre kimi komutların çalışmasını veya çalışmamasını sağlar. Operatörler ile yapılan işlemlerin sonucundaki doğru (true) ya da yanlış (false) sonucuna göre akış yönlendirilir.
- **Switch Komutu**
- Bir değişkenin olası değerlerine göre akış yönlendirilir. Case ifadesi ile olası değerler ve olası komutlar yazılır. Eğer herhangi bir case değeri ile eşleşme olursa, devamındaki komutlar çalıştırılır.
- **Koşul Operatörü**
- Koşul operatörü ?: sembolleridir.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi id'si "aciklama" olan bir paragraf etiketinin içerisindeki yazıyı değiştirir?
 - a) document.getElementById.aciklama.innetText = "dikkat";
 - b) document.getElement(aciklama).innetText = "dikkat";
 - c) document.getElementById("aciklama").innetText = "dikkat";
 - d) document.aciklama.innetText = "dikkat";
 - e) document.getElement("aciklama").innetText = "dikkat";
2. Aşağıdaki komutlardan hangisi web tarayıcısına yeni pencere açarak çıktı verir?
 - a) document.getElementById.body.Text("dikkat");
 - b) alert("dikkat");
 - c) document.alert("dikkat");
 - d) alert.open("dikkat");
 - e) concole.log("dikkat");
3. Aşağıdaki sembollerden hangisi açıklama satırlarının başına konulmalıdır?
 - a) *
 - b) /*
 - c) **
 - d) !/
 - e) //
4. Aşağıdaki değişkenlerden hangisi boolean türündedir?
 - a) var a=1;
 - b) var a="1";
 - c) var a=null;
 - d) var a=true;
 - e) var a=undefined;
5. Aşağıdaki tanımlamadan sonra, typeof(a) ifadesinin çıktısı nedir?
var a;
 - a) number
 - b) string
 - c) object
 - d) boolean
 - e) undefined

6. Aşağıdaki tanımlamadan sonra, a değişkeninin değeri nedir?
var a="20 + 40";
a) 2040
b) "2040"
c) "20 + 40"
d) 60
e) "60"
7. Aşağıdaki işlemin sonunda sonuc değişkeninin değeri nedir?
var sonuc = 20 / 5 + 5 % 2;
a) 5
b) 1
c) 0
d) 4
e) 3.82
8. Aşağıdaki işlemin sonunda sonuc değişkeninin değeri nedir?
var a=3, b=4, c=5;
var sonuc = a > b || c % b == 0;
a) true
b) false
c) 0
d) 8
e) 11
9. Aşağıdaki işlemin sonucunda a değişkeninin değeri ne olur?
var a=10, b=5;
a += b++;
a) 15
b) 5
c) 6
d) 16
e) 105
10. Aşağıdaki işlemin sonucunda t değişkeninin değeri ne olur?
var t=1, y=5;
if(t != y % 2)
 t *= 5;
else
 t += y;
a) false
b) true
c) 6
d) 5
e) 15

Cevap Anahtarı

1.c, 2.b, 3.e, 4.d, 5.e, 6.c, 7.a, 8.b, 9.a, 10.c

YARARLANILAN KAYNAKLAR

JavaScript. 18 Temmuz 2019 tarihinde <https://tr.wikibooks.org/wiki/JavaScript> adresinden erişildi.

JavaScript Operator Precedence. 19 Temmuz 2019 tarihinde <https://www.dummies.com/web-design-development/javascript-operator-precedence/> adresinden erişildi.

Javascript Tutorial. 18 Temmuz 2019 tarihinde <https://www.w3schools.com/js/> adresinden erişildi.

Javascript Dersleri. 7 Temmuz 2019 tarihinde <http://javascript.sitesi.web.tr/> adresinden erişildi.