# WEB TABANLI PROGRAMLAMA

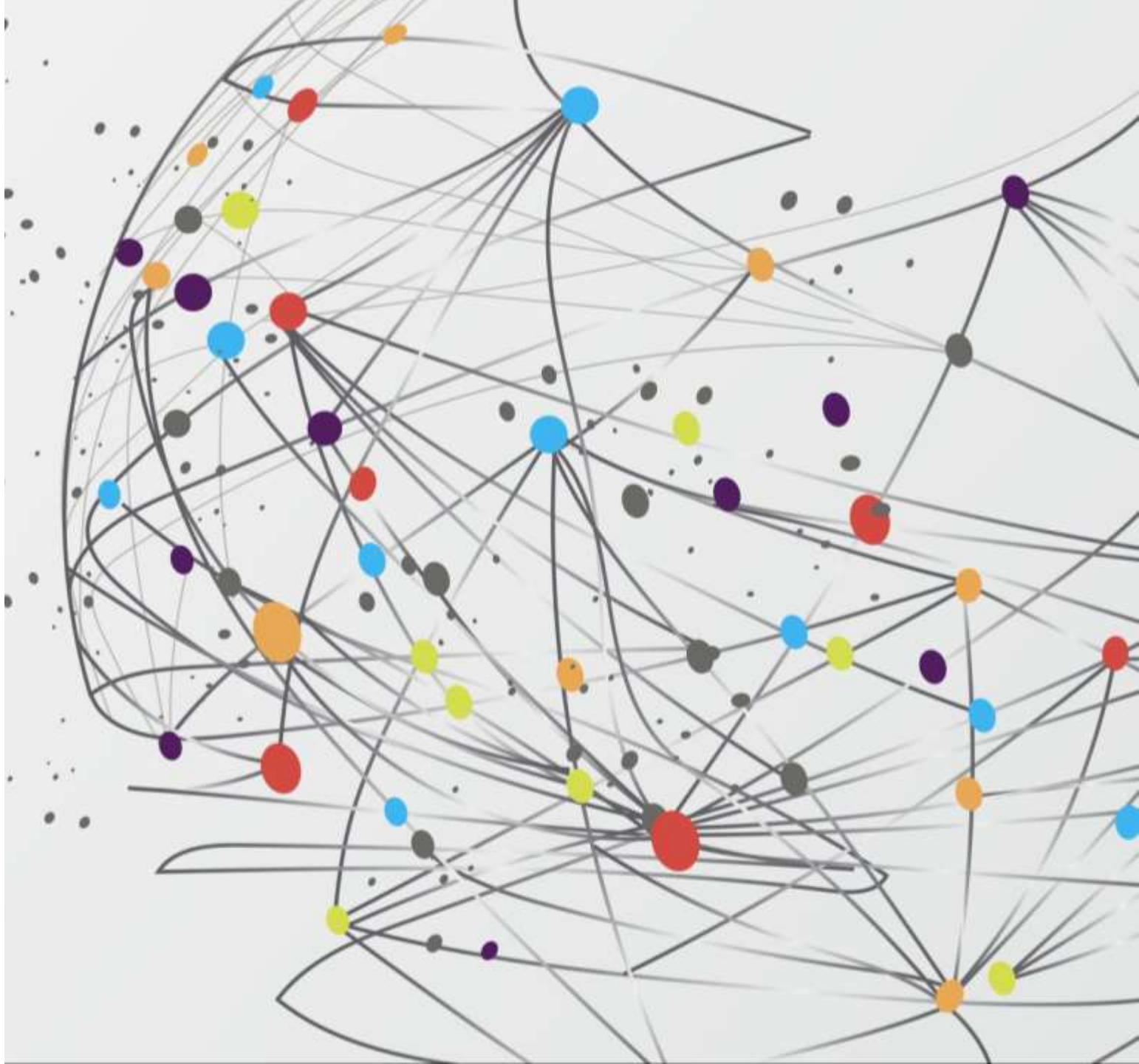
## BÖLÜM 7

JAVASCRIPT DİLİNE GİRİŞ

HTML İÇİNDE KULLANIM, DEĞİŞKENLER, OPERATÖRLER

Prof. Dr. Turgay Tugay Bilgin

turgay.bilgin@btu.edu.tr



# GENEL BAKIŞ…

- HTML dili metin içerik üzerinden birçok farklı biçimlendirmeler yapmaya, tablo oluşturmaya, form ve çerçeve oluşturmaya olanak sağlar. Fakat bunun yanında çok önemli bir dezavantajı vardır: Gerçek programlama dillerinin sağladığı değişken oluşturma, döngü oluşturma, koşul ifadeleri ve fonksiyon oluş-turma gibi birçok özelliği desteklemez. Bu sebeple HTML dili ile kişiye özel, kişiselleştirilmiş veya belirli bir koşul gerçekleştiğinde içeriği değiştirilebilen web sayfaları yapmak imkânsızdır.
- HTML dilinin bu eksikliği ilk olarak Netscape firması tarafından fark edilmiş ve buna uygun bir çözüm geliştirilmiştir. İlk olarak Aralık 1995 tarihinde Netscape firması tarafından geliştirilen Netscape Navigator 2.0 adlı web tarayıcı ile birlikte JavaScript dilinin ilk örneği piyasaya sürüldü. Bu dil, Eylül 1995 tarihine kadar resmi olarak LiveScript olarak isimlendirilmiştir [11]. Netscape, Sun Microsystems'ın geliştirdiği Java platformuna destek vermeye başlamasıy-la, o dönem yıldızı parlayan 'Java' dilinin popüleritesinden yararlanmak adına LiveScript dilinin adını Javascript olarak değiştirmiştir. Bu değişim sadece bir pazarlama stratejisiydi, bilinenin aksine Java ile JavaScript arasında isim benzerliği ve C diline benzer sözdizimleri dışında çalışma yapısı açısından bir benzerlik yoktur.
- 1996 yılının kasım ayında Netscape firması JavaScript'in endüstri standardı olarak belirlenmesi amacıyla Ecma International firmasına başvuruda bulundu. Bunun sonucunda standartlaşan sürüm ECMAScript olarak isimlendirildi ve Haziran 1998'de ISO/IEC-16262 standardına uyumlu hale getirildi [11].

# (7.1) JAVASCRIPT NEDİR ?

- JavaScript dili, CSS dili gibi HTML kodlarının içine yazılır. Yazılan kodun bir JavaScript kodu olduğu web tarayıcıya `<script>...</script>` etiketleri ile bildirilir. HTML ve CSS dilindeki gibi, JavaScript kodları da yorumlanmak için bir web tarayıcıya ihtiyaç duyarlar, **“.exe”** uzantılı, bağımsız olarak çalışabilecek bir dosya oluşmaz.

- JavaScript olay tabanlı (event driven) bir programlama dilidir. Olay, JavaScript açısından ziyaretçinin bir yere tıklaması, bir tuşa basması vs. olabilir. JavaScript sayesinde herhangi bir olayın gerçekleşmesi halinde bir web sayfasına belirli bir iş yaptırılabilir. Örneğin sayfada bir butona basıldığında sayfadaki bir metin kutusunun içine “merhaba” yazdırılabilir. Burada olay, ziyaretçinin butona tıklaması, iş ise metin kutusuna “merhaba” yazılmasıdır.


**Şekil 7.1 Resmi JavaScript Logosu**

# (7.2) JAVASCRİPT DİLİNİN AVANTAJLARI

- Javascript dilinin HTML ve CSS diline göre önemli avantajları bulunmaktadır. Bunlardan bazıları:
  - Web sunucu ile istemci arasında daha az bilgi trafiği: HTML formlar ile kullanıcıdan alınan bilgilerin istenilen ölçütlere uyumlu olup olmadığını denetlemek için JavaScript kullanılması durumunda istemci-sunucu arası git-gel trafiği azalacak, bu sayede internet bağlantı kapa-sitesi daha etkin kullanılabilecektir.
  - Ziyaretçilere anında geri dönüş: Ziyaretçiler bir HTML sayfanın bilgilerinin web sunucuya gidip oradan gelecek sonucun tekrar HTML olarak kendilerine ulaşmasını beklemek zorunda kalmadan anında geri dönüş (feedback) alırlar.
  - Daha fazla etkileşim: HTML diline göre çok daha fazla klavye, fare ve dokunma hareketleri ile etkileşim sağlanabilir.
  - Daha zengin kullanıcı arayüzü: JavaScript sayesinde kaydırma sürükleme bırakma gibi birçok farklı efekt oluşturularak masaüstü uygulama esnekliği web uygulamalarında sağlanabilmektedir.

# (7.3) JAVASCRİPT DİLİNİN SINIRLAMALARI

- JavaScript dili bir programlama dilinin birçok özelliğine sahip olsa da modern programlama dillerindeki bazı temel yetenekler "**güvenlik**" gerekçesiyle JavaScript diline dahil edilmemiştir. Bu özellikler:

→ JavaScript istemci taraflı yani sunucuda değil istemcinin web tarayıcıda çalışan bir dil olduğundan **istemcide dosya oluşturma, dosyaya yazma ve dosyadan okuma yeteneklerine sahip değildir**.

→ JavaScript dilinin **Ağ (network) bağlantısı oluşturma yeteneği yoktur**, istemci bilgisayardaki javascript kodu hiçbir şekilde ağ üzerinden başka bir makinaya TCP/UDP soket bağlantısı kuramaz.

→ JavaScript dilinin **multithreading** (çok işlemcik çalıştırma) veya **multiprocess** (çoklu süreç) desteği yoktur.

# (7.4) JAVASCRİPT KODU NASIL OLUŞTURULUR?

- JavaScript kodları HTML belgenin içinde yalnızca <script>...</script> bloğu arasındaki bölgede yazılır. <script>...</script> etiketlerini HTML belgenin sayfanın istediğiniz bir yerine konumlandırabilirsiniz fakat genellikle <head> … </head> bloğunda olması tercih edilir.
- <script>...</script> etiketleri arasındaki tüm içerik JavaScript dili kurallarına uygun olmalıdır, bu bölümde HTML veya CSS kodu yazılmamalıdır. Örnek bir script bloğu aşağıdaki gibidir:

```
<script language="javascript" type="text/javascript">
        // javascript kodları buraya yazılır.


</script>
```

- Script etiketi iki temel özelliğe sahiptir:

1. language: Bu bölüme script bloğunda hangi script dili kulanılacağı belirtilir. Geçmişte JavaScript dışında başka script dilleri de bulunmaktaydı (örneğin VBscript). Günümüzde bu alana sadece JavaScript yazılır. Bu niteliği kullanmazsanız da varsayılan değer olarak JavaScript kabul edilir.
2. type: Bu nitelik de benzer şekilde script dili tanımı yapmayı sağlar. Günümüzde yalnızca "text/javascript" değerini alır. Bu niteliği kullanmazsanız da kodunuz çalışır.

# (7.5) İLK JAVASCRIPT PROGRAMI

- Ekrana “Merhaba Dünya” yazan ilk JavaScript programımız aşağıdaki örnekte verilmiştir:

```
<html>
<body>
<script language="javascript" type="text/javascript">
   document.write("Merhaba Dünya!")
</script>
</body>
</html>
```

**Kod 7.1 İlk JavaScript programı**


**Şekil. 7.2 İlk JavaScript programı**

- Javascript dilinde ekrana bir metin yazdırmak için `document.write` fonksiyonu kullanılır. Bu fonksiyon C programlama dilindeki `printf` ve Microsoft C# dilinde `console.writeline()` komutları gibi ekrana bir metin yazmayı sağlar. Burada tek fark yazılacak ifadenin C/C++/C# dillerindeki gibi konsola değil, web tarayıcı penceresine yazdırmasıdır. Örnek kodun çıktısı Şekil 7.2’de verilmiştir.

**Hatırlatma** *Javascript dilinde kod yazarken unutulmaması gereken iki temel kural noktalı virgül kullanımı ve büyük-küçük harf duyarlılığıdır. Bunlar 7.5.1 ve 7.5.2 numaralı bölümlerde örneklerle açıklanmıştır.*

***7.5.1 Noktalı Virgül Kullanımı :*** JavaScript dilinde C/C++ dillerinde olduğu gibi her satır sonu noktalı virgül (;) işaretleriyle sonlandırılmalıdır. Buna rağmen, JavaScript bu konuda C/C++ dilleri kadar katı kurallara sahip değildir. *Eğer her bir satır yanyana değilde alt alta yazılacaksa noktalı virgül kullanmadığınızda da sorunsuz çalışacaktır*.

```
<script language="javascript" type="text/javascript">
   document.write("Merhaba")
   document.write("Ali")
</script>
```

**Kod 7.2 JavaScript'te noktalı virgül kullanımı**

- Fakat her bir komut tek satırda yan yana yazılacak ise mutlaka her komut sonrası noktalı virgül kullanılmalıdır:

```
<script language="javascript" type="text/javascript">
   document.write("Merhaba"); document.write("Ali");
</script>
```

**Kod 7.3 JavaScript'te noktalı virgül kullanımı**

- Yukarıdaki ifadede iki bağımsız satır yan yana yazıldığından aralarına noktalı virgül konması zorunludur.

**Hatırlatma** *Her durumda noktalı virgül kullanmayı alışkanlık haline getirmek iyi bir programlama alışkanlığıdır, bu sayede sonradan oluşacak satır birleştirmelerde kodunuzda problemler oluşmayacaktır.*

***7.5.2 Büyük-Küçük Harf Duyarlılığı :*** HTML dilinin aksine JavaScript dilinde büyük-küçük harf ayrımı vardır, özellikle değişken adlandırmalarda, fonksiyon isimlerinde büyük-küçük harf durumuna dikkat etmek gereklidir. JavaScript dilindeki fonksiyon ve komutların büyük bir kısmı küçük harfler ile yazılır. Örneğin Aşağıdaki kod ekrana merhaba yazmayacaktır çünkü `document.write` yerine `DOCUMENT.WRITE` yazılmıştır.

```
<script language="javascript"
type="text/javascript">
   DOCUMENT.WRITE("Merhaba");
</script>
```

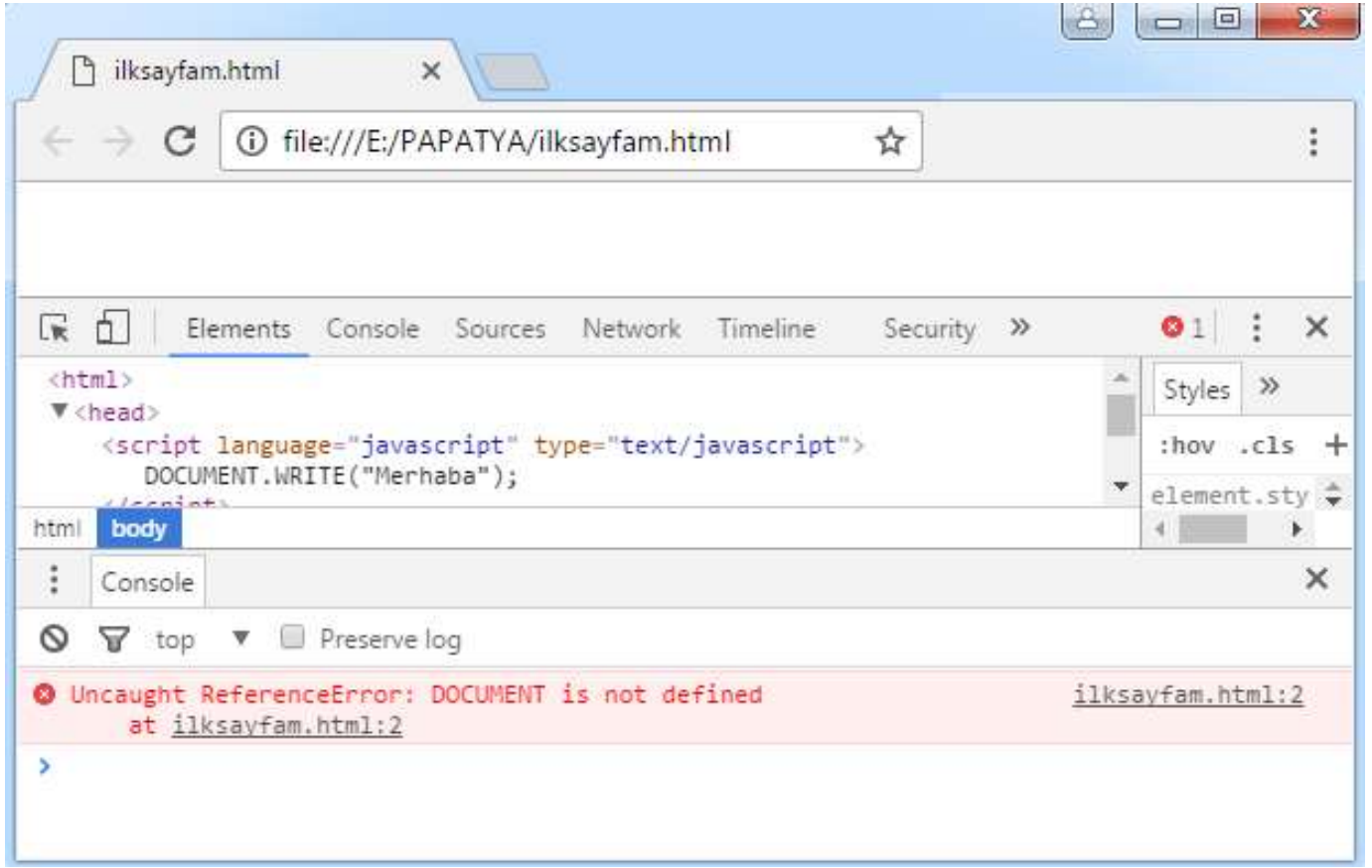
**Kod 7.4 JavaScript'te büyük küçük harf farkı**

- C/C++ dillerinin aksine, JavaScript dilinde hatalı yazılmış komutlar içeren kod çalışınca ekranda herhangi bir hata veya çökme mesajı görüntülenmez. *Yalnızca web tarayıcıda boş beyaz bir sayfa görüntülenir*.

**Hatırlatma** *Eğer C/C++ editörlerinde olduğu gibi hatanın ne olduğunu ve hangi konumda hata oluştuğunu öğrenmek isterseniz web tarayıcınızda sayfa açık durumdayken F12 tuşuna basarak debug modunu (hata ayıklama modu) başlata-bilirsiniz.*

- Kod 7.4'deki hatalı örneği Mozilla Firefox, Google Chrome veya Internet Explorer tarayıcısında çalıştırdığınızda F12 tuşuna basınca aşağıdaki gibi hatanın sebebi ve bulunduğu satırı, numarası ile birlikte görüntülenecektir.

- Şekil 7.3'de DOCUMENT adlı komutun tanımsız olduğunu belirten bir hata mesajı görüntülenmektedir.




**Şekil. 7.3 JavaScript hata ayıklama**

# (7.6) JAVASCRİPT DİLİNDE YORUMLAR

- JavaScript dilinde yorum satırları eklemek C/C++ dillerindeki ile birebir aynıdır: Şöyle ki:
  - ➢ tek satırlı yorumlar için satır başına iki adet sağa yatık bölü işareti ( // ) eklenir.
  - ➢ çok satırlı yorumlar ise "/*" ile "*/" ifadeleri arasına yazılır.
- Yorum satırları içeren bir kod çalıştırıldığında yorumlar ekranda görüntülenmez. Kullanıcı, web tarayıcısında ekrana sağ tıklayarak "*Sayfa kaynağını görüntüle*" seçeneği ile JavaScript kodlarını görüntülerse yorum satırlarını görebilir.

```
<script language="javascript" type="text/javascript">
/*
        bu satırda ekrana
        merhaba yazdırılıyor.
*/
document.write("Merhaba");
// kod burada bitiyor.
</script>
```

**Kod 7.5 JavaScript'te yorumlar**

# (7.7) HTML BELGE İÇİNDE JAVASCRIPT KULLANIMI

- HTML belge içerisinde birçok farklı konumda `<script>..</script>` etiketleri kullanarak JavaScript ekleyebilirsiniz. En çok tercih edilen konumlar şunlardır:

→ `<head>...</head>` bloğu içinde,

→ `<body>...</body>` bloğu içinde,

→ Hem `<body>...</body>` hem de `<head>...</head>` bloğu içinde,

→ `<head>...</head>` bloğu içinde `<script> ..</script>` etiketleri ile harici bir dosya ekleyerek kullanmak olarak ifade edilebilir.

***7.7.1 Harici Dosya Kullanarak Javascript Eklemek :*** `<script>` etiketinin src niteliğini kullanarak `<script>` ...`</script>` bloğu içine yazılması gereken javscript kodlarını harici bir metin dosyaya yazabilirsiniz.

**Hatırlatma** *Harici dosyanın uzantısı mutlaka ".js" olmalı ve içinde sadece javascript kodu bulunmalıdır.*

- Aşağıdaki gibi bir HTML dosya oluşturarak "**merhaba.html**" olarak adlandırınız.

```
<html>
<head>
<script type="text/javascript" src="kodlar.js">
</script>
</head>
<body>
</body>
</html>
```

**Kod 7.6 JavaScript'te harici dosya kullanımı**

Örnekte **kodlar.js** adlı bir metin dosya oluşturup içine

```
document.write("Merhaba");
```

yazdığınızda ve "**merhaba.html**" adlı belgeyi çalıştırdığınızda ekrana merhaba yazacaktır.

# (7.8) JAVASCRIPT DİLİNDE DEĞİŞKENLER

- C/C++ gibi programlama dillerinde olduğu gibi JavaScript dilinde de değişkenler kullanılmaya başlamadan önce tanımlanmalıdır. Fakat C/C++ dillerinden farklı olarak JavaScript dilinde değişken tanımlamak için sadece 1 adet anahtar kelime vardır. Değişkenler hangi türden olursa olsun `var` anahtar kelimesi ile tanımlanır.

```
<script type="text/javascript" >
var numara;
var adsoyad;
var sicaklik;
</script>
```

**Kod 7.7 JavaScript'te değişken kullanımı**

- JavaScript dili değişkenlerin hangi türde olduğunu içine değer ataması yapıldığı zaman algılar. Yukarıdaki örnekte **numara** değişkenine 145 değerini verirseniz JavaScript bu değişkenin tipini **int** (tamsayı) olarak, **numara=3.16** verirseniz tipini **double** (ondalıklı) olarak, **numara=”ondört”** verirseniz tipini **string** (metin) olarak algılayacaktır.

```
<script type="text/javascript" >
var numara=145;
var adsoyad=”Ali Yıldız”;
var sicaklik=29.8;
</script>
```

**Kod 7.8 JavaScript’te değişken kullanımı**

# (7.8) JAVASCRIPT DİLİNDE DEĞİŞKENLER
## LET

The `let` keyword was introduced in ES6 (2015).

### Cannot be Redeclared

Variables defined with `let` cannot be **redeclared**.

You cannot accidentally redeclare a variable.

With `let` you can not do this:

#### Example

```
let x = "John Doe";

let x = 0;

// SyntaxError: 'x' has already been declared
```

With `var` you can:

#### Example

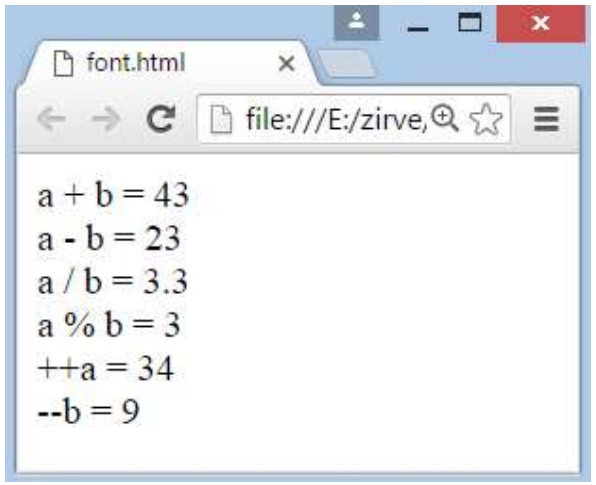
```
var x = "John Doe";

var x = 0;
```

# (7.8.1) JAVASCRİPT DEĞİŞKEN ADLANDIRMA KURALLARI

- JavaScript dilinde değişken adlandırma kuralları tamamen C/C++ dillerindeki ile aynıdır. Özetlemek gerekirse:
  - JavaScript dilinde anahtar kelimeler (komutlar) değişken adı olarak kullanılamaz, Örneğin while, if, var, for gibi kelimeler değişken ismi olamaz.
  - JavaScript dilinde değişkenler sayı ile başlayamazlar, bir harf veya alt çizgi (_) ile başlayabilirler. Örneğin 123deneme hatalı bir değişken adı, _123deneme ise doğru bir değişken adıdır.
  - JavaScript değişken adları büyük-küçük harf duyarlıdır.

# (7.9) ARİTMETİK OPERATÖRLER

- JavaScript dilinde aritmetik operatörlerin kullanımı C/C++/C#/Java dillerindeki ile aynıdır. +, -, *, /, %, ++, -- operatörlerine ait bir örnek aşağıda verilmiştir.
- Örnek kodun çıktısı Şekil 7.4'de verilmiştir.




**Şekil 7.4 Aritmetik operatörler örneğinin çıktısı**

```
<html><body>
<script type="text/javascript">
var a = 33;
var b = 10;
//toplama operatörü
            document.write("a + b = ");
            sonuc = a + b;
            document.write(sonuc + "<br />");
//çıkarma operatörü
            document.write("a - b = ");
            sonuc = a - b;
            document.write(sonuc + "<br />");
//bölme operatörü
            document.write("a / b = ");
            sonuc = a / b;
            document.write(sonuc + "<br />");
//mod alma operatörü
            document.write("a % b = ");
            sonuc = a % b;
            document.write(sonuc + "<br />");
//arttırma operatörü
            document.write("++a = ");
            sonuc = ++a;
            document.write(sonuc + "<br />");
//eksiltme operatörü
            document.write("--b = ");
            sonuc = --b;
            document.write(sonuc + "<br />");
</script>
</body></html>
```

**Kod 7.9 JavaScript'te aritmetik operatörler**

# (7.10) KARŞILAŞTIRMA OPERATÖRLERİ

- İki değerin karşılaştırılması sonucunda doğru (true) veya yanlış (false) değeri üreten ifadelerdir. İki değer veya ifade arasındaki büyüklük, küçüklük, eşitlik durumlarını test etmek için kullanılır. JavaScript dilinde karşılaştırma operatörlerin kullanımı C/C++/C#/Java dillerindeki ile aynıdır.

**Kod 7.10 Javascript'te Karşılaştırma operatörleri**

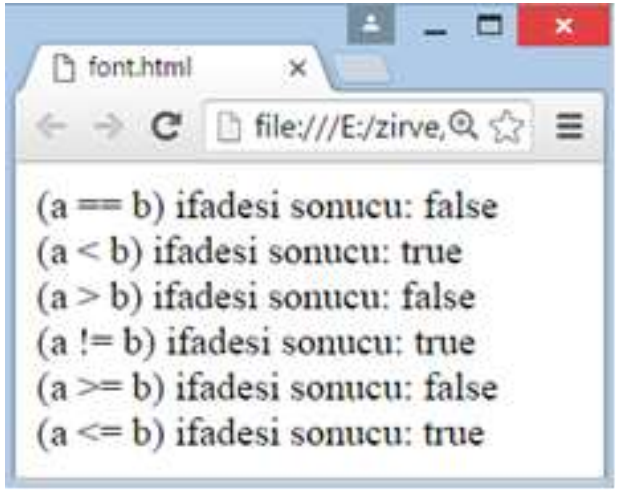
| operatör | anlamı |
|---|---|
| > | büyük |
| >= | büyük veya eşit |
| == | eşit |
| < | küçük |
| <= | küçük veya eşit |
| != | eşit değil |

- Bu operatörlerin her birine ait bir örnek aşağıda verilmiştir.

```
<html><body>
<script type="text/javascript">
var a = 10;
var b = 20;
// karşılaştırma operatörü
            document.write("(a == b) ifadesi sonucu: ");
            document.write(a == b);
            document.write("<br />");
//küçük operatörü
            document.write("(a < b) ifadesi sonucu: ");
            document.write(a < b);
            document.write("<br />");
//büyük operatörü
            document.write("(a > b) ifadesi sonucu: ");
            document.write(a > b);
            document.write("<br />");
// eşit değil operatörü
            document.write("(a != b) ifadesi sonucu: ");
            document.write(a != b);
            document.write("<br />");
//büyük eşit operatörü
            document.write("(a >= b) ifadesi sonucu: ");
            document.write(a >= b);
            document.write("<br />");
//küçük eşit operatörü
            document.write("(a <= b) ifadesi sonucu: ");
            document.write(a <= b);
            document.write("<br />");
</script>
</body></html>
```

**Kod 7.10 JavaScript'te Karşılaştırma operatörleri**

- Örnek kodun çıktısı Şekil 7.5'de verilmiştir.




**Şekil 7.5 Karşılaştırma operatörleri örneğinin çıktısı**

# (7.11) MANTIKSAL OPERATÖRLER

- Mantıksal operatörüler iki mantıksal ifade arasındaki ilişkiyi belirlemek için kullanılır. JavaScript dilinde mantıksal operatörlerin kullanımı C/C++/C#/Java dillerindeki ile aynıdır.

| operatör | anlamı |
|---|---|
| && | ve |
| \|\| | veya |
| ! | değil |

**Tablo 7.2. Mantıksal operatörler**

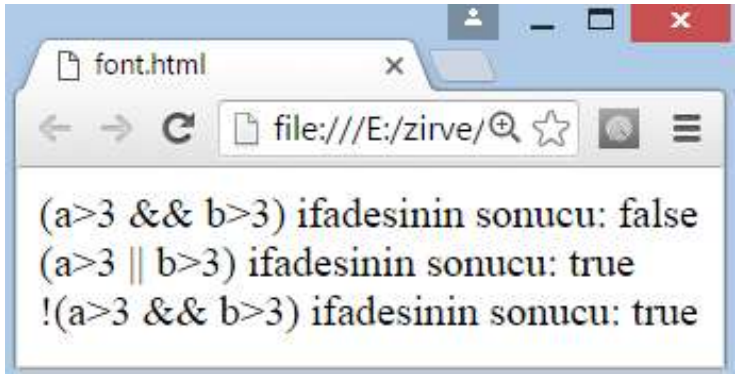
**Hatırlatma** *Mantıksal operatörlerin hem önündeki hem arkasındaki ifadeler doğru/yanlış (true/false) değer üretecek biçimde olmalıdır.*

- Bu operatörlerin her birine ait bir örnek aşağıda verilmiştir.

```
<html><body>
<script type="text/javascript">
var a = 5;
var b = 2;
// && (VE) operatörü
document.write("(a>3 &&  b>3) ifadesinin sonucu: ");
sonuc = (a>3 && b>3);
document.write(sonuc + "<br />");
// || (VEYA) operatörü
document.write("(a>3 || b>3) ifadesinin sonucu: ");
sonuc = (a>3 || b>3);
document.write(sonuc + "<br />");
// ! (DEĞİL) operatörü
document.write("!(a>3 &&  b>3) ifadesinin sonucu: ");
sonuc = (!(a>3 && b>3));
document.write(sonuc + "<br />");
</script>
</body></html>
```

**Kod 7.11 JavaScript'te mantıksal operatörler**

- Örnek kodun çıktısı Şekil 7.6'da verilmiştir.




**Şekil 7.6 Mantıksal operatörleri örneği çıktısı**

# (7.12) KOŞUL İFADELERİ: İF...ELSE

- JavaScript dilinde if..else koşul ifadelerinin kullanımı C/C++/C#/Java dillerindeki ile aynıdır. 3 farklı türde if yapısı kullanılabilmektedir:
  - → if yapısı
  - → if...else yapısı
  - → if...else if... yapısı
- Bunların kullanımına ait bir örnek aşağıda verilmiştir.

```
<html><body>
<script type="text/javascript">
var plaka = 34;

if( plaka == 17 ){
   document.write("<b>Çanakkale</b>");
}else if( plaka == 34 ){
   document.write("<b>İstanbul</b>");
}else if( plaka == 35 ){
   document.write("<b>İzmir</b>");
}else{
  document.write("<b>Bilinmeyen plaka!</b>");
}
</script>
</body></html>
```

**Kod 7.12 JavaScript'te Koşul İfadeleri**

- Örnek kod çalıştırıldığında, plaka değişkeni değeri 34 olduğundan ekranda kalın yazı tipi ile İstanbul yazısı görüntülenecektir.

> **Hatırlatma** ☞ *Örnekte görüleceği gibi JavaScript kodlarını ekrana yazdırırken HTML biçimlendirmeleri kullanmak gerektiğinde HTML kodları doğrudan JavaScript kodunun içine yazılamaz.*
>
> *Bunun yerine, örnekteki kalın yazma etiketi* `(<b>)` *gibi* `document.write` *ifadesi içinde web tarayıcı ekranına gönderilir. Web tarayıcı bu ifadeyi HTML komutu olarak alır ve gerekli biçimlendirmeyi gerçekleştirir.*

# (7.13) ÜÇ İŞLEÇLİ OPERATÖR (? :)

- `if..else` yapısının tek satırda yazılmasını sağlayan `(? : )` operatörü C/C++/C#/Java dillerinde olduğu gibi JavaScript dilinde de bulunur. Aynı problemin hem `if..else` yapısı ile hem `(? : )` operatörü ile çözümü aşağıdaki örnekte görülmektedir. Örnekte a ve b değişkenleri karşılaştırılmaktadır. Eğer a değişkeni b den büyük ise sonuc değişkenine a değeri, tam tersi ise sonuç değişkenine b değeri atamakta ve ekrana yazdırmaktadır. Her iki çözüm de birbirine eşdeğerdir.

**Üç işleçli operatörlü çözüm**

```
<html><body>
<script type="text/javascript">
var a = 10;
var b = 20;
sonuc = (a > b) ? 100 : 200;
document.write(sonuc);
</script>
</body></html>
```

**İf .. else yapısıyla çözüm**

```
<html><body>
<script type="text/javascript">
var a = 10;
var b = 20;
if( a > b ){
   sonuc=100;
}else{
   sonuc=200;
}
document.write(sonuc);
</script>
</body></html>
```

**Hatırlatma** *Üç işleçli operatör ile gerçekleştirilen çözüm daha az satır sayısına sahip olduğundan az da olsa yer tasarrufu sağlamaktadır.*

**Kod 7.13 JavaScript'te üç işleçli operatör**

# (7.14) switch..case YAPISI

- JavaScript dilinde `switch..case` yapısının kullanımı C/C++/C#/Java dillerindeki ile aynıdır. Aşağıdaki örnekte `switch..case` yapısının kullanımı görülmektedir.

- Örnek çalıştırıldığında, plaka değeri 34 olarak tanımlı olduğundan ekranda "İstanbul" yazacaktır.

```
<html><body>
<script type="text/javascript">
var plaka=34;
switch (plaka)
{
  case 17: document.write("Çanakkale");
           break;
  case 22: document.write("Edirne");
           break;
  case 34: document.write("İstanbul");
           break;
  case 35: document.write("İzmir");
           break;
  default:  document.write("Böyle bir plaka yok!")
}
</script>
</body></html>
```

**Kod 7.14 JavaScript'te switch..case Yapısı**

# (7.15) JAVASCRIPT DİLİNDE FOR DÖNGÜLERİ

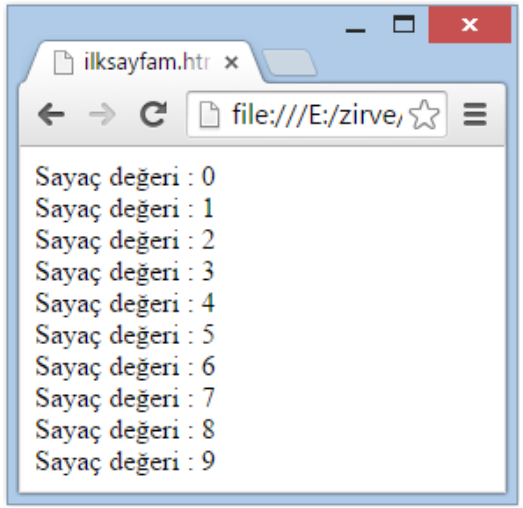
- JavaScript dilinde for döngüsünün kullanımı C/C++/C#/Java dillerindeki ile aynıdır. Aşağıdaki örnekte for döngüsünün kullanımı görülmektedir.

```
<html><body>
<script type="text/javascript">
var sayac;
 for(sayac = 0; sayac < 10; sayac++){
  document.write("Sayaç değeri : " + sayac + "<br />" );
 }
</script>
</body></html>
```

**Kod 7.15 JavaScript'te for döngüleri**

- Yukarıdaki örnek çalıştığında Şekil 7.7'deki görüntü elde edilecektir. Ekrana öncelikle `Sayaç değeri:` ifadesi, daha sonra döngüde o anki sayaç değişkeninin sayısal değeri ve en son olarak da `<br />` komutu ile alt satıra geçirme işlemi yapılmaktadır.

**Hatırlatma** *Javascript dilinde program çıktısı komut satırında değil web tarayıcısı penceresinde görüntülendiğinden C dilindeki "\n" veya C++ dilindeki "endl" gibi alt satıra geçirme ifadeleri işe yaramayacaktır. Bunların yerine HTML dilindeki alt satıra geçirme ifadesi olan <br /> kullanılmalıdır.*




**Şekil 7.7 JavaScript dilinde for döngüsü örneği**

# (7.16) JAVASCRİPT DİLİNDE WHİLE DÖNGÜLERİ

- JavaScript dilinde while döngüsünün kullanımı C/C++/C#/Java dillerindeki ile aynıdır. Aşağıdaki örnekte while döngüsünün kullanımı görülmektedir. Bölüm 7.15'deki for döngüsünün while benzeri aşağıdaki örnekte görülmektedir. Bu örnekte sayi adlı bir değişken tanımlanıyor. Bu değişkenin değeri ilk çalıştırmada 1 oldğundan ekrana `<h1>Merhaba</h1>` metni yazdırılmaktadır. Sonraki iterasyonda sayi değeri 2 olacağından ekrana `<h2>Merhaba</h2>` yazacaktır. Döngü 3,4,5 değerleri için aynı şekilde devam edecektir. Sayi değeri 6 olduğunda ekrana `<h6>Merhaba</h6>` yazarak program sonlanacaktır.

**Kod 7.16 Javascript'te while döngüleri**

```
<html><body>
<script type="text/javascript">
var sayi=1;
while (sayi < 7){
  document.write("<h" + sayi + ">Merhaba</h" + sayi + ">");
  sayi++;
}
</script>
</body></html>
```


**Şekil 7.8 JavaScript dilinde while döngüsü örneği**

- Örnek çalıştırıldığında Şekil 7.8'deki gibi giderek küçülen Merhaba ifadesi yazılacaktır

- Bu örnek, HTML etiketlerinin JavaScript içinde kullanımının daha kolay kavranabilmesi amacıyla verilmiştir. Buna benzer HTML kodları içeren başka örnekleri kendiniz de yazabilirsiniz. Böylece, az satır JavaScript kodu ile çok fazla HTML etiketi oluşturabilirsiniz.

# (7.17) ÖZET

- HTML dili, gerçek programlama dillerinin sağladığı değişken oluşturma, döngü oluşturma, koşul ifadeleri ve fonksiyon oluşturma gibi birçok özelliği desteklemez. HTML dilinin bu eksikliği ilk olarak Netscape firması tarafından fark edilmiş ve buna uygun bir çözüm geliştirilmiştir. JavaScript dili, CSS dili gibi HTML kodlarının içine yazılır. Yazılan kodun bir JavaScript kodu olduğu web tarayıcıya `<script>...</script>` etiketleri ile bildirilir. JavaScript olay tabanlı (event driven) bir programlama dili olup bazı sınırlamaları bulunmaktadır. Bunlardan en önemlisi istemcide dosya oluşturma, dosyaya yazma ve dosyadan okuma yeteneklerine sahip olmamasıdır. HTML dilinin aksine Javascript dilinde büyük-küçük harf ayrımı vardır. Javascript dilinde değişken tanımlamak için sadece bir anahtar kelime vardır. Değişkenler hangi türden olursa olsun `var` anahtar kelimesi ile tanımlanır. JavaScript dilinin operatörleri, koşul ifadeleri ve döngüleri gibi birçok temel öğesi C/C++ dillerindeki gibi tanımlanır.

# KAYNAKÇA


- [11] https://w3techs.com/technologies/overview/programming_language/all, Erişim: Mart 2017