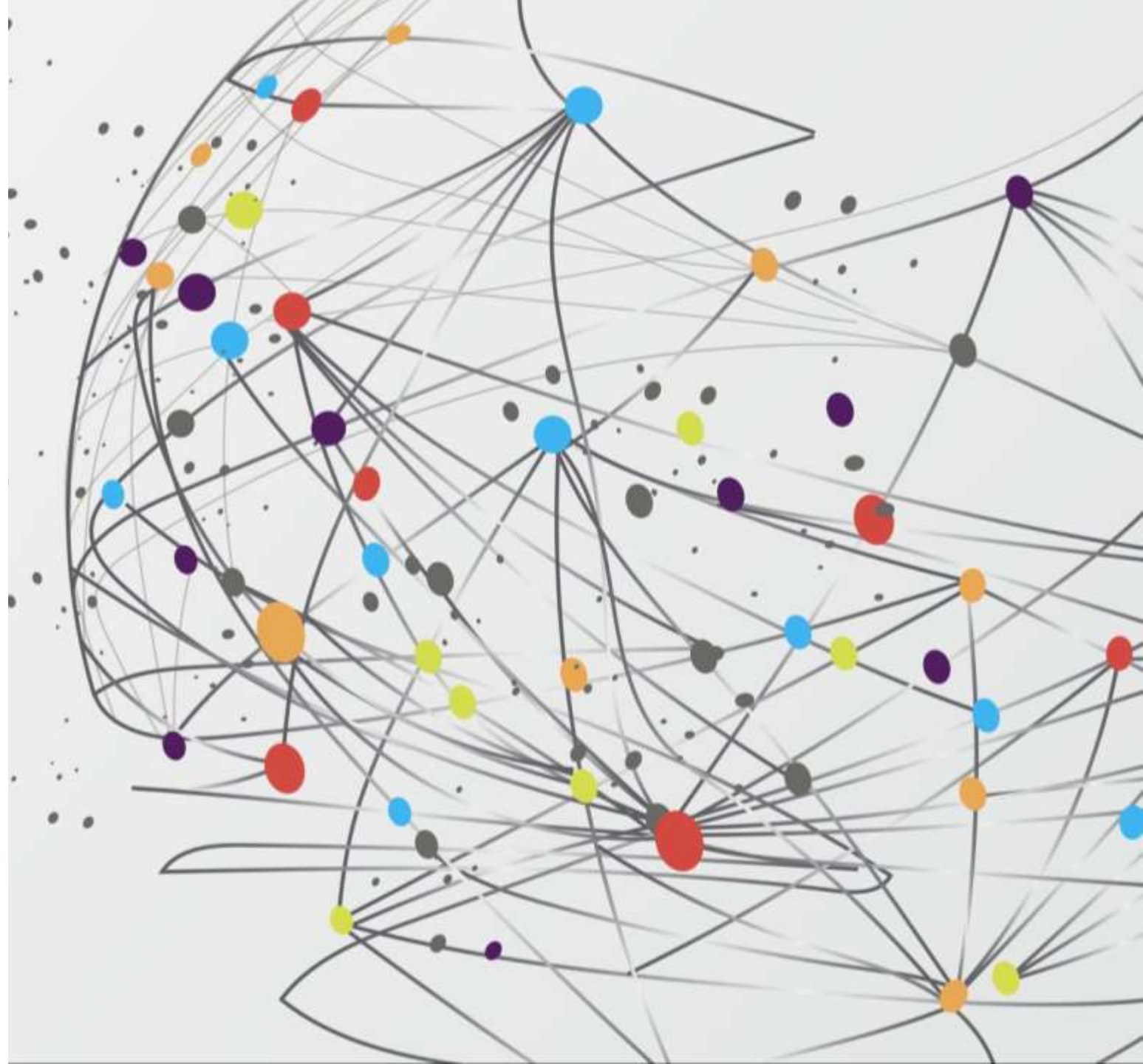

WEB TABANLI PROGRAMLAMA

BÖLÜM 8

FONKSİYON, DİZİ VE NESNELER
JAVASCRIPT'TE FONKSİYON, DİZİ VE
NESNELER

Prof. Dr. Turgay Tugay Bilgin

turgay.bilgin@btu.edu.tr



GENEL BAKIŞ...

8.1) Fonksiyon Tanımlama

8.2) Fonksiyon Çağırma

8.3) Parametre Alan Fonksiyonlar

8.4) Değer Döndüren Fonksiyonlar (*return* İfadesi)

8.5) JavaScript Nesneleri: Dizgi Nesnesi (String Object)

8.5.1) Dizgi Uzunluğunu Bulmak

8.5.2) Dizginin Belirli Bir Karakterine Erişim: `charAt()` Metodu

8.5.3) Dizgileri Arka Arkaya Birleştirmek: `concat()`

8.5.4) Dizgileri Belirli Bir Karaktere Göre Parçalamak: `split()`

8.5.5) Dizgiden Alt Dizgi Türetmek : `substring()`

8.5.6) Dizginin Tümünü Küçük Veya Büyük Harfe Çevirme

8.6) JavaScript'te Diziler

8.6.1) Dizilerin Uzunluğu

8.6.2) Dizilerin Sıralanması: `sort()` Metodu

8.7) JavaScript'te Tarih Nesnesi

8.8) Tarih Nesnesi Metotları

8.8.1) `getTime()` Metodu

8.8.2) `getFullYear()` Metodu

8.8.3) `getDay()` Metodu

8.8.4) Diğer Tarih Metotları

8.9) Özet

(8.1) FONKSİYON TANIMLAMA

- Modern programlama dillerinin çoğunda bulunan dizi yapıları ve fonksiyonlar Javascript dilinde de mevcuttur. Bunların yanı sıra C, C++, Java ve C# dilleri gibi JavaScript dili de nesneye yönelik bir programlama dilidir.
- Programlama dillerinde fonksiyon yapıları sayesinde gereksiz kod tekrarları engellenir. Örneğin bir ürüne ait KDV hesaplama işlemi gerçekleştirilecekse bunu her seferinde kodlamak yerine bunu hesaplayan bir fonksiyon yazılarak gerekli olduğu her yerde çağırılabilir. Javascript dilinde basit bir fonksiyon tanımlama örneği aşağıda verilmiştir. Örnekte parametre almayan bir fonksiyon oluşturulmuştur.

```
<html><head>  
<script type="text/javascript">  
  function Selamla()  
  {  
    alert("Merhaba!");  
  }  
</script>  
</head></html>
```

Kod 8.1 Fonksiyon tanımlama

Hatırlatma 🌀 Javascript dilinde “**alert**” komutu, kullanıcıya bilgi mesajı içeren ve sadece “Tamam” (eğer web tarayıcı İngilizce ise “OK”) butonu görüntüleyen bir pencere oluşturur. Bu pencerede “Tamam” (veya “OK”) butonu tıklanmadığı sürece web tarayıcıda başka hiçbir işlem yapılamaz.

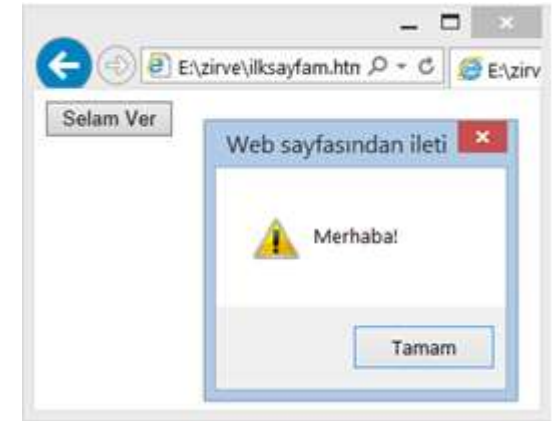
(8.2) FONKSİYON ÇAĞIRMA

- JavaScript dilinde fonksiyonlar çoğunlukla buton, seçenek kutusu, metin kutusu, açılan kutu gibi HTML form elemanlarına tanımlanan tıklama olayları ile çağırılır. Fakat, JavaScript kodlarının kendi içerisinde de JavaScript fonksiyonları çağırılabilir.
- Aşağıdaki örnekte bir HTML butona tıklanınca ekrana “Merhaba” yazan Bölüm 8.1’deki örnek fonksiyon çağırılmıştır.

Kod 8.2 Fonksiyon çağırma

```
<html><head>
<script type="text/javascript">
  function Selamla ()
  {
    alert("Merhaba!");
  }
</script>
</head><body>
<form>
<input type="button" onclick="Selamla()" value="Selam
Ver">
</form>
</body></html>
```

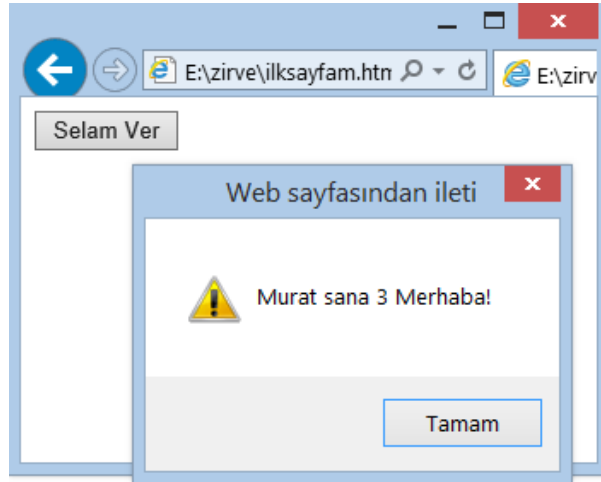
- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda Şekil 8.1'deki görüntü elde edilir. Bu uygulamada “Selam ver” butonu tıklanınca Şekil 8.1'deki gibi bir pencere içerisinde “Merhaba” yazısı görünecektir.



Şekil 8.1. JavaScript dilinde fonksiyon kullanımı

(8.3) PARAMETRE ALAN FONKSİYONLAR

- Diğer tüm programlama dillerinde olduğu gibi JavaScript fonksiyonları da parametre alabilirler. Parametreler aralarına virgül yerleştirilerek kullanılır ve parametre sayısında bir sınırlama yoktur. Aşağıdaki örnekte selamla fonksiyonu isim olarak Murat ve adet olarak 3 değerini parametre olarak almaktadır.



Şekil 8.2. JavaScript dilinde parametre alan fonksiyon örneği

```
<html><head>
<script type="text/javascript">
function Selamla(isim, adet)
{
    alert(isim + " sana " + adet + " Merhaba!");
}
</script>
</head><body>
<form>
<input type="button" onclick="Selamla('Murat',3)"
value="Selam Ver">
</form>
</body></html>
```

Kod 8.3 Parametre alan fonksiyonlar

- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda Şekil 8.2'deki görüntü elde edilir.

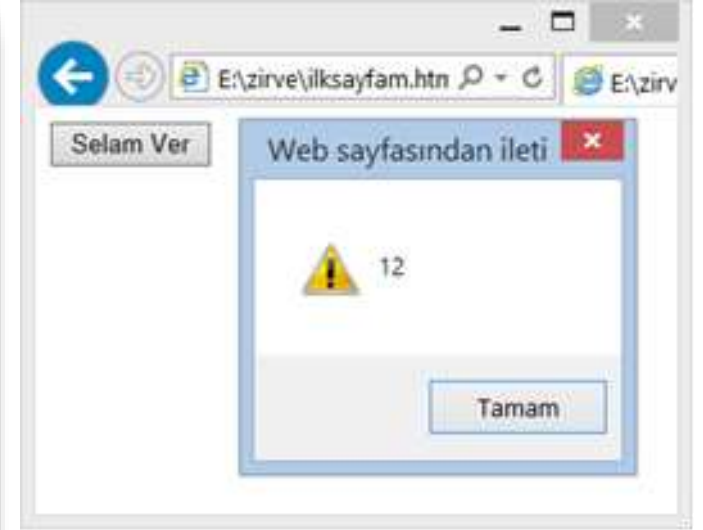
(8.4) DEĞER DÖNDÜREN FONKSİYONLAR (RETURN İFADESİ)

- JavaScript dilinde C/C++/C#/Java dillerinde olduğu gibi fonksiyonlar çağırıldıkları konuma değer döndürebilirler. Döndürülecek değer return ifadesi ile belirtilir.

Hatırlatma ~ “return” ifadesi içeren satır fonksiyon bloğunun en son satırı olmalıdır.

Kod 8.4 Değer döndüren fonksiyonlar

```
<html><head>
<script type="text/javascript">
function topla(sayi1, sayi2)
{
    return sayi1 + sayi2;
}
function hesapla()
{
    var sonuc;
    sonuc = topla(3, 9);
    alert(sonuc);
}
</script>
</head><body>
<form>
<input type="button" onclick="hesapla()" value="Selam Ver">
</form>
</body></html>
```



Şekil 8.3. JavaScript dilinde değer döndüren fonksiyon örneği

- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda Şekil 8.4'teki görüntü elde edilir. Örnekte buton tıklandığında `hesapla()` fonksiyonu çağırılıyor, o da kendi içinde `topla()` fonksiyonunu 3 ve 9 parametreleri ile çağırarak toplama işlemi yaptırıyor. `topla()` fonksiyonu iki sayıyı toplayıp sonuç değeri `hesapla()` fonksiyonuna geri döndürmektedir. Örnek çalıştırıldığında ekranda 12 sayısı görüntülenir.


(8.6) JAVASCRIPT NESNELERİ: DİZGİ NESNESİ (STRING OBJECT)

- JavaScript dili C++/C# ve Java gibi nesneye yönelik bir yapıda geliştirilmiştir. JavaScript dilinde nesneler `new` anahtar kelimesi kullanılarak oluşturulur. Bir dizgi (string) nesnesi oluşturmak için aşağıdaki yazım biçimi kullanılır:

```
var a = new String("merhaba");
```

- Yukarıda dizgi türünden bir `a` nesnesi oluşturulmaktadır. Bu nesnenin içinde “merhaba” dizgisi depolanmaktadır.

8.5.1 Dizgi Uzunluğunu Bulmak : Bir dizgi nesnesinin uzunluğu `length` ifadesi ile öğrenilebilir.

Hatırlatma  `length` ifadesi bir metot olmadığından sonunda parantezler kullanılmaz.

```
<html>
<body>
<script type="text/javascript">
    var bilmece = new String("Bu köşe kış köşesi");
    document.write("Bilmece uzunluğu:" + bilmece.length);
</script>
</body>
</html>
```

Kod 8.5 Dizgi uzunluğunu bulmak

Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

```
Bilmece uzunluğu:18
```

8.5.2 Dizginin Belirli Bir Karakterine Erişim: `charAt()` Metodu : Bir dizgi nesnesi bir dizi gibi düşünülebilir. Bizim sıranumarasını belirttiğimiz herhangi bir karaktere erişmek için `charAt()` metodu kullanılır. Örneğin dizginin 3. Karakterine `charAt(3)` şeklinde erişilir.

- Yandaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

```
0.karakter: B
1.karakter: u
2.karakter:
3.karakter: k
4.karakter: ö
```

Hatırlatma 🌀 Diğer tüm programlama dillerinde olduğu gibi dizi indislerinin sıfırdan başladığı unutulmamalıdır.

```
<html>
<body>
<script type="text/javascript">
    var str = new String( "Bu köşe kış
    köşesi" );
    document.write("<br />0. karakter: " +
    str.charAt(0));
    document.write("<br />1. karakter: " +
    str.charAt(1));
    document.write("<br />2. karakter: " +
    str.charAt(2));
    document.write("<br />3. karakter: " +
    str.charAt(3));
    document.write("<br />4. karakter: " +
    str.charAt(4));
</script>
</body>
</html>
```

Kod 8.6 `charAt()` metodu

8.5.3 Dizgileri Arka Arkaya Birleştirmek: `concat()` : Javascript dilinde birinci dizgi nesnesinin arkasına ikinciyi ekleyerek yeni bir nesne oluşturmak için `concat()` metodu kullanılır.

- Aşağıdaki örneklerde `dizgi1` ve `dizgi2` şeklinde iki dizgi birleştirilerek `dizgi3` oluşturuluyor ve ekrana yazdırılıyor.

```
<html>
<body>
<script type="text/javascript">
var dizgi1 = new String( "Ankara" );
var dizgi2 = new String( "Havası" );
var dizgi3 = dizgi1.concat( dizgi2 );
document.write(dizgi3);
</script>
</body>
</html>
```

Kod 8.7 Dizgileri arka arkaya birleştirmek

```
<html>
<body>
<script type="text/javascript">
var dizgi1 = new String( "Ankara" );
var dizgi2 = new String( "Havası" );
var dizgi3 = dizgi1 + dizgi2;
document.write(dizgi3);
</script>
</body>
</html>
```

Kod 8.8 Dizgileri arka arkaya toplamak

Aynı örnek `concat()` yerine toplama operatörü ile de yazılabilir. Sonuç değişmeyecektir.

- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:



AnkaraHavası

8.5.4 Dizgileri Belirli Bir Karaktere Göre Parçalamak: `split()`: Dizgi nesneleri belirli bir karaktere göre parçalara ayrılabilir, bu durumda her bir parça bağımsız bir dizgi nesnesi olacaktır. Bunun için `split()` metodu kullanılır:

```
dizgi.split(ayıraç, alınacak parça sayısı);
```

- `split` metodu 2 parametre alır:
 - **ayıraç:** dizgi hangi karakterler görüldüğünde parçalanacaksa o karakter ayıraç olarak verilir.
 - **alınacak parça sayısı:** parçalanmış dizginin kaç adet elemanının alınacağını bildirir.

```
<html>
<body>
<script type="text/javascript">
var a = "Üç tunç tas has hoş hoşaf.";
var parcalar = a.split(" ", 3);
document.write( parcalar );
</script>
</body>
</html>
```

Kod 8.9 Dizgileri belirli bir karaktere göre parçalamak

- Yukarıdaki örnekte dizgi boşluk karakteri görülen her konumda parçalanarak yeni bir dizi elemanı elde edilmekte ve elde edilen dizinin ilk 3 elemanı alınarak “parçalar” isimli dizi değişkeninde depolanmaktadır. Örnek çalıştırıldığında şu çıktı elde edilir:

```
Üç, tunç, tas
```

8.5.5 Dizgiden Alt Dizgi Türetmek : substring() : substring() metodu bir dizgi nesnesinin sizin verdiğiniz bir başlangıç numarasından yine sizin verdiğiniz bir bitiş numarasına kadar olan (*bitişin kendisi dahil değil*) kısmını seçerek kullanabilmenize olanak sağlar. Genel kullanım şekli şöyledir:

dizgiAdi.substring(başlangıç, bitiş)

- Başlangıç değeri sıfırdan (0) başlar ve dizginin gerçek boyutunun 1 eksiğine kadar değer alabilir. Bitiş değeri de sıfırdan başlar ve dizginin gerçek boyutunun 1 eksiğine kadar değer alabilir. Bitiş değeri opsiyoneldir, verilmediği takdirde dizginin sonuna kadar tüm değerler alınır.

```
<html>
<body>
<script type="text/javascript">
var dizgi = "İkiBinOnYedi Ağustos";
document.write(dizgi.substring(1,2));
document.write("<br />" + dizgi.substring(0, 10));
document.write("<br />" + dizgi.substring(5));
</script>
</body>
</html>
```

Kod 8.10 Dizgiden alt dizgi türetmek

- Yandaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

```
k
İkiBinOnYe
nOnYedi Ağustos
```

8.5.6 Dizginin Tümünü Küçük Veya Büyük Harfe Çevirme :

- Bir dizgi nesnesinin;
 - tüm karakterlerini küçük harfe çevirmek için `toLowerCase()` Metodu,
 - tüm karakterlerini büyük harfe çevirmek için `toUpperCase()` Metodu kullanılır.

```
<html>
<body>
<script type="text/javascript">
var dizgi = "İkiBinOnYedi Ağustos";
document.write(dizgi.toLowerCase());
document.write("<br />");
document.write(dizgi.toUpperCase());
</script>
</body>
</html>
```

Kod 8.11 Dizginin tümünü küçük veya büyük harfe çevirme

- Yandaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

```
ikibinonyedi ağustos
İKIBINONYEDI AĞUSTOS
```

Hatırlatma ~ `toLowerCase()` ve `toUpperCase()` metotlarını Türkçe metinlerde kullanırken dikkatli olunmalıdır. Çünkü Türkçemizdeki küçük i harfi `toUpperCase()` metodu tarafından büyük I harfine dönüştürülmektedir. Bunun sebebi ingiliz alfabesinde küçük i harfinin büyüğünün büyük I harfi olmasıdır. Yukarıdaki örnekte bu problem açıkça görülmektedir.

(8.6) JAVASCRIPT'TE DİZİLER

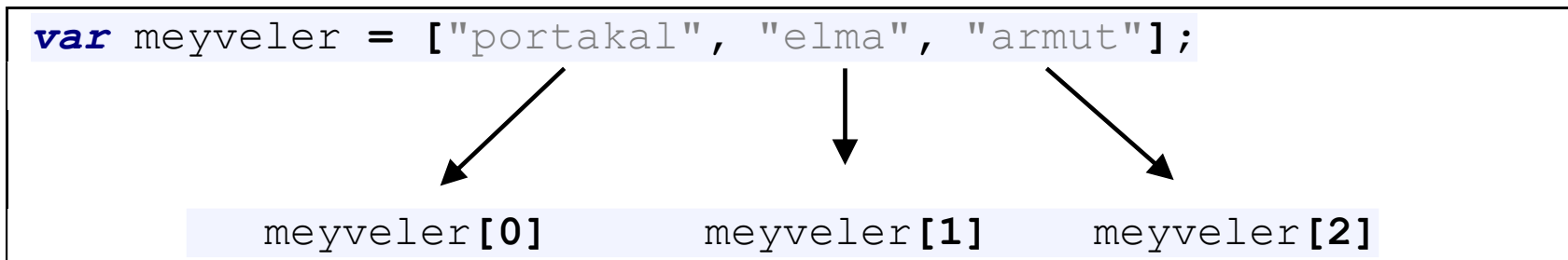
- JavaScript dilinde diziler nesnelerdir. Bu sebeple dizi oluşturulurken `new Array` anahtar kelimesi kullanılır. Diziler hem sayısal değerlerden oluşabildiği gibi hem de string (dizgi) türünden değerlerden oluşabilir. Aşağıda her ikisine de bir örnek verilmiştir:

```
var meyveler = new Array( "portakal", "elma", "armut" );  
var fiyatlar = new Array( 12, 9, 21, 34, 17 );
```

- Bunların dışında `new Array()` kullanmadan doğrudan değişken adına köşeli parantezler içerisinde değerler girilerek de dizi tanımlanabilir. Aşağıda her iki-sine de bir örnek verilmiştir:

```
var meyveler = ["portakal", "elma", "armut"];  
var fiyatlar = [ 12, 9, 21, 34, 17 ];
```


-
- Diğer tüm programlama dillerinde olduğu gibi dizilerin indis değerleri sıfırdan başladığından elemanlara erişim aşağıdaki gibidir:



(8.6.1) DİZİLERİN UZUNLUĞU

- Diziler bir nesne olduğundan dizilerin eleman sayısı yani uzunluğunu bulmak nesnelerdeki gibi length anahtar kelimesi ile gerçekleştirilir.

Kod 8.12 Dizilerin uzunluğu

```
<html>
<body>
<script type="text/javascript">
  var meyveler = new Array( "portakal", "elma", "armut" );
  document.write("Meyveler uzunluğu:" + meyveler.length);
</script>
</body>
</html>
```

- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

Meyveler uzunluğu:3

(8.6.2) DİZİLERİN SIRALANMASI: SORT() METODU

- Dizilerin sıralanması sort() metodu aracılığıyla gerçekleştirilir. Aşağıda sort() metodu kullanımı için bir örnek verilmiştir:

Kod 8.13 Dizilerin sıralanması

```
<html>
<body>
<script type="text/javascript">
  var meyveler = new Array( "portakal", "elma", "armut" );
  document.write("<b>Meyveler dizisi :</b>" + meyveler);
  var sirali = meyveler.sort();
  document.write("<br /><b>Sıralı Meyveler:</b>" + sirali);
</script>
</body>
</html>
```

- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

Meyveler dizisi :portakal,elma,armut
Sıralı Meyveler :armut,elma,portakal

Hatırlatma ✍ `sort()` metodu sadece alfabetik sıralama yapar, sayısal sıralama gerçekleştirmez, sayısal sıralama yapmak için diğer programlama dillerinde olduğu gibi sıralama işlemi yapan kod yazmanız gerekir.

(8.7) JAVASCRIPT'TE TARİH NESNESİ

- JavaScript dilinde tarih tipinden verileri depolamak için tarih nesnesi kullanılır. Tarih nesnesi diğer tüm nesne tiplerinde olduğu gibi new anahtar kelimesi ve Date() ifadesi ile oluşturulur. Tarih nesnesi;

→ Yıl,

→ Ay,

→ Gün,

→ Saat,

→ Dakika,

→ Saniye,

→ Milisaniye

bileşenleri içerir. Tarih nesnesi oluşturulurken hiçbir parametre verilmeden new Date() şeklinde kullanıldığında o anki tarihi içeren bir nesne oluşturulmuş olur.

```
<html>
<body>
<script type="text/javascript">
  var tarih = new Date();
  document.write(tarih);
</script>
</body>
</html>
```

Kod 8.14 JavaScript'te tarih nesnesi



Mon Feb 13 2017 22:17:31 GMT+0300 (Türkiye Yaz Saati)

Hatırlatma ⚡ Date() nesnesi kullanıldığı andaki tarih değerini alır ve sadece onu görüntüler, herhangi bir güncelleme işlemi yapmaz. Yukarıdaki örneği çalıştırdığınızda web tarayıcınızda **Yenile (Refresh)** butonuna bastığınızda farklı bir tarih alındığını göreceksiniz.

- `Date()` nesnesine parametre verilerek istenilen bir tarihi depolayan bir nesne oluşturulabilir. Bunun için aşağıdaki gibi bir sıralama ile tarih bileşenleri içeren bir dizgi parametre olarak verilmelidir.

```
new Date(yıl, ay, gün, saat, dakika, saniye)
```

```
<html>
<body>
<script type="text/javascript">
var tarih = new Date(2013, 09, 12, 13, 14, 00);
document.write(tarih);
</script>
</body>
</html>
```

Kod 8.15 JavaScript'te tarih nesnesi

- Yandaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

```
Sat Oct 12 2013 13:14:00 GMT+0300
```

Hatırlatma 🌀 `Date()` nesnesi ile alınan tarih bilgisi aslında sizin bilgisayarınızın sistem saatinden alınır, dolayısıyla bilgisayarınızın saat ve tarihi hatalı ise görüntülenen tarih de hatalı olacaktır.

Hatırlatma 🌀 `Date()` nesnesi içinde kullanılan gün ve ay isimleri işletim sisteminiz Türkçe olsa bile İngilizcedir



(8.8) TARİH NESNESİ METOTLARI

- Tarih nesnesinin içeriğini değiştirmek veya belirli bölümlerini ekranda görüntülemek için birçok hazır metot bulunmaktadır. Bunlardan en sık kullanılan birkaçı aşağıdaki bölümlerde verilmiştir.

(8.8.1) getTime() METODU

```
<html>
<body>
<script type="text/javascript">
  var tarih = new
  Date(2013,01,01,00,00,00);
  document.write(tarih.getTime());
</script>
</body>
</html>
```

Kod 8.16 getTime() metodu

- Bu metot ile 01.01.1970 yılından itibaren parametre olarak aldığı tarihe kadar geçen süreyi milisaniye cinsinden ekranda görüntüler. Bunun en büyük faydası, iki farklı tarih arasındaki farkı alırken artık yıllar veya farklı sayıda gün içeren aylar arasında fark hesabı yapma zorunluluğunu ortadan kaldırmasıdır.
- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda 01.01.1970 tarihinden itibaren 01.01.2013 yılına kadar geçen süre mili-saniye cinsinden ekrana aşağıdaki gibi yazdırılacaktır.

1359669600000

(8.8.2) getFullYear() METODU

- Bu metot ile tarih nesnesinin içerdiği değerin 4 haneli yıl bilgisi elde edilebilir.

Kod 8.17 getFullYear() metodu

```
<html>
<body>
<script type="text/javascript">
var tarih = new Date(2013,01,01,00,00,00);
document.write(tarih.getFullYear());
</script>
</body>
</html>
```

- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

2013

(8.8.3) getday() METODU

- Bu metot ile tarih nesnesinin içerdiği değerin gün bilgisi 0-6 arası sayısal değer olarak elde edilebilir. Bu değerler içinde sıfır (0) değeri Pazar gününü 6 değeri ise Cumartesi gününü belirtir.

Hatırlatma *↪ Günlerin numaraları işletim sisteminizde haftanın başlangıç günü ayarına göre değişiklik gösterebilir.*

Kod 8.18 getDay() metodu

```
<html>
<body>
<script type="text/javascript">
var tarih = new Date(2013,01,01,00,00,00);
document.write(tarih.getDay());
</script>
</body>
</html>
```

- Yukarıdaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

5

- Buradan hareketle 2013 yılının 1 Ocak gününün Cumartesi günü olduğu sonucuna varılır. Günleri sayı olarak değil, gün adı olarak yazdırmak isterseniz gün adlarını tutan bir dizi kullanabilirsiniz.

Kod 8.19 getDay() metodu ile gün adları yazdırma

```
<html>
<body>
<script type="text/javascript">
var tarih = new Date(2013,01,01,00,00,00);
var gunler = [ "Pazar",
               "Pazartesi",
               "Salı",
               "Çarşamba",
               "Perşembe",
               "Cuma",
               "Cumartesi"];
document.write(gunler[tarih.getDay()]);
</script>
</body>
</html>
```

- Yandaki kodları **sayfam.html** adıyla kaydederek web tarayıcısında çalıştırdığınızda şu çıktı elde edilir:

Cuma

(8.8.4) DİĞER TARİH METOTLARI

- Bu bölümde örnek verilen metotların dışında:
 - **saat** bilgisini almayı sağlayan `getHours()` ,
 - **saniye** bilgisini almayı sağlayan `getSeconds()` ,
 - **milisaniye** bilgisini almayı sağlayan `getMilliseconds()` ,
 - **dakika** bilgisini almayı sağlayan `getMinutes()` ,
 - **ay** bilgisini almayı sağlayan `getMonths()`

metotları da bulunmaktadır. Bunların her birinin kullanımı yukarıda verilen örnekler ile aynı olduğundan örnek verilmemiştir.

(8.9) ÖZET

- JavaScript dilinde fonksiyonlar çoğunlukla buton, seçenek kutusu, metin kutusu, açılan kutu gibi HTML form elemanlarına tanımlanan tıklama olayları ile çağırılır. Diğer tüm programlama dillerinde olduğu gibi JavaScript fonksiyonları da parametre alabilirler. JavaScript dili C++/C# ve Java gibi nesneye yönelik bir yapıda geliştirilmiştir. Nesneler `new` anahtar kelimesi kullanılarak oluşturulur. JavaScript dilinde diziler nesnedirler. Bu sebeple dizi oluşturulurken `new Array` anahtar kelimesi kullanılır. Diziler hem sayısal değerlerden oluşabildiği gibi hem de `string`(dizgi) türünden değerlerden oluşabilir. JavaScript dilinde tarih tipinden verileri depolamak için tarih nesnesi kullanılır. Tarih nesnesi diğer tüm nesne tiplerinde olduğu gibi `new` anahtar kelimesi ve `Date()` ifadesi ile oluşturulur.