

Web Projesi Yönetimi

Web Projesi Yönetim Aşamaları

Web Projesi Yönetim Aşamaları

- ▶ Web Projesinin Başlangıç Aşaması
- ▶ Web Projesinin Tanımlanması
- ▶ Web Projesinin Yapısı
- ▶ Tasarım Aşaması ve Prototip Web Sitesi Üretimi
- ▶ Web Sitesinin Üretim ve Test Aşaması
- ▶ Web Sitesinin Sunumu
- ▶ Değerlendirme ve Sürdürülebilirliktir

1. Web Projesinin Başlangıç Aşaması

- ▶ Endüstrinin Analizi
- ▶ Sektör Rekabet Analizi
- ▶ Endüstri ve Pazar Araştırması
- ▶ Hedef Kullanıcıların Analizi
- ▶ Kullanıcı Profillerinin Çıkarılması
- ▶ Ön Testler ve Araştırmalar
- ▶ Tüketici Pazar Araştırması

2. Web Projesinin Tanımlanması

- ▶ Genel Amaç ve Hedeflerin Tanımlanması
- ▶ Öncelikli Hedeflerin Belirlenmesi
- ▶ Amaçların Belirlenmesi
- ▶ Müşteri Anketlerinin Dağıtılması
- ▶ Proje Planının Yaratılması
- ▶ Proje Takımının Oluşturulması
- ▶ Proje Programının Belirlenmesi
- ▶ Bütçe Planı ve Ayarlamalar
- ▶ Nihai Site Yapısına Karar Verilmesi
- ▶ İşlevsellik Test Planı
- ▶ Kalite Güvenliği (QA) Planı

3. Web Projesinin Yapısı

- ▶ Web Projesinin Yapısı
- ▶ Kullanıcılarla Test Etme
- ▶ Web Sitesi İçeriği (Kapsamı)
- ▶ İçeriğe Ait Ana Hatların Belirlenmesi
- ▶ İçerik Teslim Planının Yapılması
- ▶ Web Sitesi Görünümü
- ▶ Site Haritasının Oluşturulması
- ▶ İsimlendirme Konvansiyonları
- ▶ Fonksiyonel Alan Adresleri
- ▶ Ekran Görünümü
- ▶ Ekran Şemalarının Yaratılması
- ▶ Yönlendirmenin Belirlenmesi
- ▶ İçerik ve İşlevsellik Çerçevesi
- ▶ Kullanıcı Görünümü
- ▶ Kullanıcı Senaryolarının Yaratılması
- ▶ Kullanıcı Yollarının Belirlenmesi
- ▶ Kullanıcılarla Test Etme

4. Tasarım Aşaması ve Prototip Web Sitesi Üretimi

- ▶ Görsel Tasarım Yaklaşımı
- ▶ Kavramsal Beyin Fırtınası
- ▶ Grafik Şablonların Yaratılması
- ▶ Tasarım Rehberinin Hazırlanması
- ▶ HTML Prototipinin Yaratılması
- ▶ HTML Prototip Geliştirme
- ▶ Test Etme ve Doğrulama
- ▶ İşlevsellik Testi

5. Web Sitesinin Üretimi ve Test Aşaması

- ▶ Web Sitesi Üretimi Öncesi Planlama
- ▶ Faaliyet Alanı ve Bütçenin Yeniden Değerlendirilmesi
- ▶ Teknik İhtiyaçların Belirlenmesi
- ▶ Versiyon Kontrol Yöntemleri
- ▶ Dosya Yapılarının Düzenlenmesi
- ▶ Site Bakım Planının Hazırlanması
- ▶ Web Sitesinin Yapılandırılması
- ▶ Grafiklerin Düzenlenmesi
- ▶ HTML Şablonlarının Oluşturulması
- ▶ Yazılı Metinlerin Adreslendirilmesi
- ▶ Sayfa Düzenlemeleri
- ▶ İçerik ve Üretimin Sonlandırılması
- ▶ Test Etme
- ▶ Kalite Güvenlik (QA) Test Planı
- ▶ Kalite Güvenlik Test Yönetimi
- ▶ Güvenlik Yönetimi (Virüs takibi ve önlenmesi)

6. Web Sitesinin Sunumu

- ▶ Son Değerlendirme ve Kalite Güvenliği
- ▶ Sitenin İnternette Sunumu
- ▶ Site Güvenliğinin Kontrol Edilmesi
- ▶ HTML Biçim Kılavuzu Hazırlanması
- ▶ Bakım Uygulamaları
- ▶ Dokümantasyon Takibi
- ▶ Site Tutundurma Faaliyetleri
- ▶ Tutundurma Planı
- ▶ Arama Motorlarına Katılımın Sağlanması

7. Değerlendirme ve Sürdürülebilirlik

- ▶ Başarı Aşamalarının Gözden Geçirilmesi
- ▶ Kullanıcı Geri Beslemesi Sağlanması
- ▶ Kullanıcıların İzlenmesi
- ▶ Ölçülebilir Sonuçların Elde Edilmesi
- ▶ Proje Sonrasının Değerlendirilmesi
- ▶ Proje Aşamalarının Yeniden Kontrolü

Javascript

Javascript

- ▶ Netscape firmasının 1995 yılında Netscape Navigator 2.0 ile birlikte geliştirdiği C dilinde yazılmış bir script dilidir.
- ▶ Pascal, C++, Java, C#, Basic, Visual C, Visual Basic, Visual C#, Delphi vb gibi bir programlama dili değildir.
- ▶ Kodlama sonunda com veya exe gibi derlenmiş dosya oluşturulmaz.
- ▶ Script dilleri, web sayfalarının dinamik ve etkileşimli olarak hazırlanmasını sağlar.
- ▶ Javascript ile HTML kaynak kodları değiştirilebilir. Böylece dinamik web sayfaları hazırlamak mümkün olur.
- ▶ Kullanıcı tarafı (client) olduğu için javascript kodları çok hızlıdır.
- ▶ Javascript ile hazırlanmış web sayfaları sadece javascript desteği veren web tarayıcılarında izlenebilir.
- ▶ JQuery kodları javascript ile yazılmaktadır. Bu nedenle jquery kullanabilmek için javascript kodları hakkında bilgi sahibi olunmalıdır.

Javascript'in Özellikleri

- ▶ Kullanıcı taraflı çalışan bir script dildir.
- ▶ Javascript kodları ile bilgisayardan herhangi bir dosya okunamaz, dosya üzerine yazılamaz veya dosya oluşturulamaz. (Çerezler (cookies) hariç)
- ▶ HTML sayfalarının bir parçasıdır. Tek başına bir anlam ifade etmez veya çalışmaz.
- ▶ Yorumlanabilmeleri için tarayıcıya ihtiyaç duyulur.
- ▶ JS ile karşı sunucuya dosya gönderilemez.
- ▶ JS kodları `<script>...</script>` etiketleri arasında yazılır.
- ▶ Kodların çoğalması durumunda harici dosya (.js uzantılı) kullanılır ve HTML sayfasından çağrılır.
- ▶ JS kodları herhangi bir metin editöründe yazılabilir.
- ▶ Her JS kodu farklı tarayıcılarda aynı sonucu vermeyebilir. Bazı kullanımlar bazı tarayıcılarda çalışmayabilir.

Sayfa İçerisinde Javascript Kodlarının Kullanımı

- ▶ Genellikle `<head>...</head>` etiketleri içerisinde `<script>...</script>` etiketleri arasında yazılır.
- ▶ HTML sayfasının diğer bölümlerinde de JS kodları yazılabilir.

```
<html>
<head>
<meta ....>
<title>...</title>
<script type="text/javascript" language="javascript">
//JS kodları
</script>
<style>...</style>
</head>
<body>
</body>
</html>
```

Harici Javascript Dosyasını Çağırma

- ▶ Harici javascript kodlarının yazıldığı dosya «.js» uzantısı ile kaydedilir.
- ▶ JS dosyası (ornek.js)

```
// JavaScript Document  
alert("Message...");
```
- ▶ HTML sayfası (ornek.html)

```
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
<title>JS Örnek</title>  
<script type="text/javascript" src="ornek.js" language="javascript"></script>  
</head>  
<body>  
</body>  
</html>
```

Yazım Kuralları

- ▶ Bütün ifadeler **harf** veya "_" ile başlar. İfadelerin ikinci veya sonraki karakterleri rakam olabilir.
- ▶ Değişken isimlerinin uzunluğu 255 karakter olabilir.
- ▶ Büyük-küçük harf ayrımı vardır. "XYZ" ile "xyz" aynı ifade değildir.
- ▶ Türkçe karakterler tercih edilmez. (çğiöşüÇĞİÖŞÜ)
- ▶ Anahtar kelimeler (if, for gibi) diğer tanımlamalarda kullanılamaz.
- ▶ Satır sonlarında << ; >> karakteri kullanılır.
- ▶ Kod blokları << { ... } >> arasında yazılır. Açılan blok işaretinin karşılığı mutlaka olmalıdır. Kod blokları kapatılmalıdır.
- ▶ << ' >> ile << " >> arasında fark yoktur. İç içe kullanımlarda tercih edilebilir. Özellikle HTML veya CSS özelliklerini JS içerisinde düzenlerken kullanılabilir.
- ▶ Metinlerde özel karakterlerin yer alması gerekiyorsa, öncesinde << \ >> işareti eklenir.
 - ▶ "Javascript \"SCRIPT\" bir dildir." → Javascript "SCRIPT" bir dildir.

Yazım Kuralları

- ▶ **Açıklama satırları**, `"//"` veya `"/* ... */"` işaretleri ile tanımlanır.
 - ▶ Her satırın başında `"//"` işareti konulabilir
 - ▶ `"/* ... */"` işaretleri arasında tüm açıklamalar tanımlanabilir.
 - ▶ Aynı zamanda bu işaretler, bazı JS kodlarının kapatılması için kullanılır.
- ▶ **Kod satırlarını bölmek için**, `"\"` işareti kullanılır.
 - ▶ Bölme işlemi sadece metinlerde kullanılır. JS kodları bölünemez.
- ▶ **Eski tarayıcılarda kod gizlemek için**, `"<!-- ... -->"` işaretleri kullanılır.
 - ▶ Tarayıcıların JS kodlarını açıklama satırları gibi algılamasını sağlar.

Değişkenler

- ▶ Javascript'te bilgi saklama ve bilgiyi kullanma amaçlı kullanılır.
- ▶ Değişkenler "var" ifadesi ile tanımlanır.
- ▶ Değişken veri tipi, değer atandıktan sonra JS tarafından belirlenir.

- ▶ Örnek tanımlamalar:

```
var x;
```

```
var y=5;
```

```
var a="ali";
```

```
var b=true;
```

```
var c='veli';
```

- ▶ Veri Türleri:

- ▶ String : Tek veya çift tırnak arasında yazılan ifadeler metin olarak değerlendirilir.
- ▶ Number: Tamsayılar, ondalık sayılar vs. Ondalık basamaklar "." işareti ile ayrılır.
- ▶ Boolean: Mantıksal "true/false" değerler için kullanılır.
- ▶ Null: Boş değer atamak için kullanılır. Genellikle herhangi bir değişkenin içeriğini boşaltmak için kullanılabilir.

Genel ve Yerel Değişkenler, Sabitler

- ▶ **Genel değişkenler**, tüm script içerisinde kullanılabilir.
- ▶ **Yerel değişkenler**, sadece tanımlandığı fonksiyon veya kod bloğu içerisinde kullanılabilir.

- ▶ **Örnek:**

```
var x=12.5;
function islem(){
  var y=4;
  alert(x*y);
}
```

```
islem();
alert(y); //çalışmaz
```

- ▶ **Sabitler:** Değeri değiştirilemeyen ifadelerdir. "const" ile tanımlanır.
 - ▶ `const pi=3.14;`

Operatörler-1

- ▶ Sayısal veya mantıksal bir veya birden fazla değer arasında atama, karşılaştırma ve hesaplama gibi işlemlerin yapılmasını sağlar.
- ▶ Operatörlerin tamamı ya bir sonuç döndürür ya da kullanıldığı değeri etkiler.
- ▶ **Aritmetik:** Toplama (+), Çıkarma (-), Çarpma (*), Bölme (/), Kalan (%)
- ▶ **Atama:** Atama (=), Topla Atama(+), Çıkar atama(-), Çarp atama(*), Böl atama(/)
- ▶ **Artırma ve Azaltma:** Artırma (++), Azaltma(--), $a++:a=a+1$, $a--:a=a-1$
- ▶ **Karşılaştırma:** Eşitse(==), Kesin eşitse (===), Eşit değilse (!=), Kesin eşit değilse (!==), Büyüğe (>), Küçüğe (<), Büyük veya eşitse (>=), Küçük veya eşitse (<=)
 - ▶ Kesin eşit ve kesin eşit değil operatörlerinde değerler ile birlikte veri türleri dikkate alınır.
- ▶ **Mantıksal:** Ve (&&), Veya (||), Değil (!)
- ▶ **String birleştirme:** Sadece metinlerde iki veya daha fazla metin "+" işareti ile birleştirilir.
- ▶ **Koşullu atama:** Koşullu değer atama işlemlerinde "?:" kullanılır.
var değişken= x==y? "Eşitse": "Eşit değilse";

Operatörler-2

- ▶ **IN:** Belirtilen değer veya indis, belirtilen dizi içerisinde varsa "true", yoksa "false" değerini döndürür. Ayrıca belirtilen özellik belirtilen nesne içerisinde varsa "true", yoksa "false" döndürür.
`var dizi=new Array(4, 6, 3, 8); 2 in dizi --> true,`
- ▶ **INSTANCEOF:** Nesne belirtilen türde ise "true" değeri döndürür.
`var dizi=new Array("Ali", "Veli"); dizi instanceof Array --> true`
- ▶ **NEW:** Kullanıcı tanımlı bir nesne veya ön tanımlı nesne tiplerinden birinin (Array, Boolean, Date, Function, Image, Number, Object, Option, RegExp, String) örneğini tanımlamak için kullanılır.
`var dizi=new Array(1,2,3);`
- ▶ **THIS:** Geçerli nesneyi ifade eder. Genel olarak metotlar içerisinde kullanılır.
`function fonk(nesne){...}`
`<input type="text" name="txt" onChange="fonk(this);"/>`
- ▶ **TYPEOF:** Değişkenin veri türünü döndürür.
`var x=5; typeof(x) --> number`

Karar Yapıları (if, Switch)

- ▶ Kodlar, JS tarafından işlenirken, akış yönünün belirli şartlara göre değiştirilmesini sağlayan ifadelerdir.
- ▶ **IF:** Belirtilen koşul/koşullar doğru ise kod bloğu çalıştırılır.
`var x=5; if(x==5) document.write("Eşit");`
- ▶ **IF...ELSE:** Belirtilen kod bloğu doğru ise IF'ten sonra yazılan kodlar, değilse ELSE'den sonra yazılan kodlar çalıştırılır.
`var x=5; if(x==5) document.write("Eşit"); else document.write("Eşit değil");`
- ▶ **SWITCH:** Koşullar belirli ise kullanılan karar yapısıdır.
`var x=1;
switch (x)
{
case 1: document.write("bir"); break;
case 2: document.write("iki"); break;
case 3: document.write("üç"); break;
default: document.write("yanlış"); break;
}`

Döngüler (For, While, Do-While, For..in)

- ▶ **FOR:** Herhangi bir kod bloğunu birden fazla çalıştırmak için kullanılan döngü yapısıdır. Belirtilen koşul doğru olduğu sürece döngü devam eder.

```
for(var i=1;i<=10;i++){document.write(i);}
```

- ▶ **WHILE:** Herhangi bir kod bloğunu birden fazla çalıştırmak için kullanılan döngü yapısıdır. For'dan farklı olarak başlangıç değeri ve artış değeri belirtilmez. Koşul doğru olduğu sürece döngü devam eder.

```
var i=0; while(i<10) {document.write(++i);}
```

- ▶ **DO...WHILE:** While döngüsüne benzer. While döngüsünden farklı olarak, kod bloğu en az bir defa çalıştırılır. Kod bloğu çalıştırıldıktan sonra koşulun doğru olup olmadığı kontrol edilir.

```
var i=0; do {document.write(++i);} while(i<10);
```

- ▶ **FOR...IN:** Bir nesnenin özelliklerine ya da bir dizinin öğelerine erişmek için kullanılır.

```
var dizi=new Array("Ali", "Veli");
```

```
for(indis in dizi) document.write(dizi[indis]);
```

```
var nesne={ad:"Ali", soyad:"Koş"};
```

```
for(ozellik in nesne) document.write(nesne[ozellik]);
```

İç içe Döngüler

```

tablo="<table border=1 cellspacing=0 cellpadding=1>";
for(i=1; i<=5; i++)
{
    tablo+="<tr>";
    for(j=1; j<=5; j++)
        tablo+="<td>"+i*j+"</td>";
    tablo+="</tr>";
}
tablo+="</table>";
document.writeln(tablo);

```

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

```

for(var i=1, j=1; i<=5 && j<=5; i++, j++)
    document.writeln(i*j);

```

1 4 9 16 25

Break, Continue

Break

- ▶ Herhangi bir döngüyü belirli bir yerde sonlandırmak için kullanılır.

```
for(i=1; i<=5; i++)  
{  
    document.writeln(i);  
    if(i==3) break;  
}
```

Continue

- ▶ Döngü içerisinde kullanıldığında döngü bir sonraki adımdan işlemine devam eder.

```
for(i=1; i<=5; i++)  
{  
    if(i==3) continue;  
    document.writeln(i);  
}
```


Try-Catch-Finally (Hata Yönetimi)

- ▶ Kodlarda sıfıra bölünme, sonsuz döngüye girme, tanımlanmamış değişkenlerle çalışma vs hatalara sebep olur.
- ▶ Hata oluşabilecek kodlar "**TRY**" bloğuna, hata oluşursa yapılacak işlemler "**CATCH**" bloğuna yazılır.
- ▶ Hata olsa da, olmasa da çalıştırılması gereken kodlar "**FINALLY**" bloğuna yazılır.

```
var x=5, y=0;
try
{
    document.write("Sonuç:"+x/y);
}
catch(error)
{
    document.write(error.message);
}
finally
{
    document.write("sürekli çalışan kodlar");
}
```

Try-Throw (Hata Yönetimi)

- ▶ Try bloğunda, özel hata blokları oluşturulabilir.
- ▶ Throw ile hata mesajları özelleştirilebilir.

```
var x=5, y=0;
try{
    if(y==0) throw "sıfıra bölme hatası";
    document.write("Sonuç:"+x/y);
}catch(err){
    document.write(err);
}
```

Diziler

- ▶ Aynı tipte birden fazla veriyi saklamak için kullanılan veri yapısıdır.
- ▶ Dizideki her değerın bir indeksi vardır. bu indekslerle değerlere ulaşılabilir.
- ▶ Dizi indeksleri 0'dan başlar.
- ▶ Dizi indeks numaraları dizi adından sonra köşeli parantezler içerisinde belirtilir. Dizi[indeks]

```
var dizi=new Array(); dizi[0]="Ali"; dizi[1]="Veli";  
var dizi=new Array("Ali", "Veli");  
var dizi=["Ali","Veli"];
```

```
var dizi=new Array(2);  
dizi[0]=new Array("Ali","Veli","Selami"); dizi[1]=new Array("Ayşe","Oya","İpek");
```

```
var dizi=new Array(new Array("Ali","Veli","Selami"), new Array("Ayşe","Oya","İpek"));
```

```
var dizi=[ ["Ali","Veli","Selami"], ["Ayşe","Oya","İpek"] ];
```

```
for(i=0;i<2;i++)  
    for(j=0;j<3;j++)  
        document.write(dizi[i][j]);
```

Dizi Metotları

- ▶ **LENGTH**: Dizinin eleman sayısını verir.
Dizi.length
- ▶ **CONCAT**: Dizileri birleştirir.
Dizi1.concat(dizi2)
- ▶ **JOIN**: Dizi içindeki tüm değerleri belirtilen özel karakter ile birleştirir ve string olarak geri döndürür. Özel karakter belirtilmezse "," işareti ile birleştirilir.
Dizi.join("-");
- ▶ **SHIFT**: Dizinin ilk elemanını verir ve diziden siler.
Dizi.shift();
- ▶ **POP**: Dizinin son elemanını verir ve diziden siler.
Dizi.pop();
- ▶ **PUSH**: Dizinin sonuna bir veya daha fazla değer ekler.
Dizi.push("1","2",...);
- ▶ **UNSHIFT**: Dizinin başına bir veya daha fazla değer ekler.
Dizi.unshift("1","2",...);

Dizi Metotları

- ▶ **REVERSE:** Diziyi ters çevirir.

`Dizi.reverse();`

- ▶ **SORT:** Diziyi alfabetik olarak sıralar.

- ▶ ASCII kodlarına göre sıralama yapar.

- ▶ Sayılar alfabetik olarak sıralanır.

- ▶ Sayıların gerçek değerlerine göre sıralanması için fonksiyon yazılmalıdır.

```
var dizi=new Array(245, 56, 25, 89, 123, 763);
```

```
dizi=dizi.sort();
```

```
dizi=dizi.sort(function(a,b){return b-a}); //sayıları sıralar.
```

```
for(i=0; i<dizi.length;i++) document.write(dizi[i]+"<br>");
```

- ▶ **SLICE:** Dizinin bir kısmını almak için kullanılır.

- ▶ `Slice(başlangıç indeksi, bitiş indeksi)`

- ▶ Bitiş indeksi belirtilmezse, dizinin sonuna kadar seçilir.

- ▶ Bitiş indeksi negatif değer olursa, dizinin sonundan belirtilen kadar değer alınmaz.

- ▶ Bitiş indeksinde bulunan değer alınmaz.

```
Dizi.slice(2); //2.indeksten başla, sonuna kadar al
```

```
Dizi.slice(2,5); //2.indeksten başla, 5. indeks hariç diğerlerini al
```

```
Dizi.slice(2,-1); //2.indeksten başla, sonuna kadar al, sondan 1 değeri alma
```

Dizi Metotları

- ▶ **SPLICE**: Dizi elemanlarından bir kısmını sildikten sonra, belirtilen değerleri diziye ekler.
 - ▶ `Dizi.splice(başlangıç indeks, silinecek eleman sayısı, eklenecek değerler)`
`Dizi.splice(1,4,"x","y","z");`
- ▶ **EVERY**: Dizi elemanlarının belirli bir niteliğe sahip olup olmadığını kontrol eder. `true/false` değeri döndürür.
- ▶ **FILTER**: Dizi elemanlarını fonksiyon aracılığıyla filtreler. Fonksiyonun `true/false` değerlerini döndürmesi gerekir.

```
function kontrol(deger){if(!isNaN(deger)) return true; else return false;}
var dizi=new Array(245, 56, 25, 89, 123, 763, "X");
document.write(dizi.every(kontrol)); --> false
var degerler=dizi.filter(kontrol);
document.write(degerler.join()); --> 245, 56, 25, 89, 123, 763
```
- ▶ **FOREACH**: Dizi elemanları üzerinde belirtilen fonksiyon ile işlem yapar.

```
var dizi=new Array(245, 56, 25, 89, 123, 763,"X");
function kontrol(deger){if(deger%2==0) document.write("çift"); else document.write("tek");}
dizi.forEach(kontrol);
```

Dizi Metotları

- ▶ **INDEXOF**: Belirtilen değerin, dizi içindeki konumunu verir.
`Dizi.indexOf("X");`
- ▶ **LASTINDEXOF**: Belirtilen değerin, dizi içindeki en son konumunu verir.
`Dizi.lastIndexOf("X");`
- ▶ **MAP**: Dizinin elemanlarını belirtilen bir fonksiyondan geçirir. İşlem sonuçları dizi olarak geri döner.

```
var dizi=new Array(3, 6, 8);  
function kare(n){ return n*n}  
var s=dizi.map(kare);  
document.write(s.join());
```
- ▶ **REDUCE**: Dizinin elemanları üzerinde fonksiyon ile işlem yapar.

```
var dizi=new Array(3, 6, 8, 9);  
function toplama(a,b){ return a+b;}  
document.write(dizi.reduce(toplama)); --> 26
```
- ▶ **SOME**: Dizi elemanlarını kontrol eder. true/false

```
var dizi=new Array(3, 6, 8, 9);  
function kontrol(x){ return (x>7);}  
document.write(dizi.some(kontrol)); -->true
```

Fonksiyonlar

- ▶ Bağımsız çalışan kod bloklarıdır.
- ▶ Kodların tekrar yazılmasına gerek kalmadan birden fazla çalıştırılması sağlanır.
- ▶ Fonksiyonlar değer döndürebilir veya sadece işlem yapılabilir.
- ▶ Fonksiyonlar parametresiz olabilir.
 - ▶ `function fonksiyon_adi(){...}`
 - ▶ `var fonksiyon_adi=function(){...}`
 - ▶ `function fonksiyon_adi(parametreler){...}`
 - ▶ `var fonksiyon_adi=function(parametreler){...}`
 - ▶ `function fonksiyon_adi(){return değer;}`
 - ▶ `var fonksiyon_adi=function(){return değer;}`

```
function topla(a,b){return a+b;} //var topla=function(a,b){return a+b;}  
document.write(topla(4,5));
```


Fonksiyonlar

- Fonksiyon parametreleri dizi olarak elde edilebilir.

```
<script>
  function Degerler(par)
  {
    var toplam=0;
    for(var i in Degerler.arguments)
      toplam+=Degerler.arguments[i];
    return toplam;
  }
  document.write(Degerler(4,3,8));
</script>
```

Kendini Çağırarak Çalışan Fonksiyonlar (Recursive)

- Kendini çağırabilen fonksiyonlara özyinelemeli (recursive) fonksiyon denir.

```
function faktoryel(n)
{
    if(n==1)
        return 1;
    else
        return n*faktoryel(n-1);
}

document.write(faktoryel(5));
```

Hazır Fonksiyonlar

- ▶ **parseInt**: String olarak belirtilen sayıları tamsayıya çevirir.
 - ▶ `parseInt("100", 2) --> 4`
 - ▶ `parseInt("100", 8) --> 64`
 - ▶ `parseInt("100", 16) --> 256`
 - ▶ `parseInt("100", 10) --> 100`
- ▶ **parseFloat**: String veriyi float sayı tipine çevirir.
 - ▶ `parseFloat("3.14") --> 3.14`
- ▶ **String**: Object değeri stringe çevirir.
`document.write(String(23));`
- ▶ **Number**: Object değeri sayıya çevirir.
`document.write(Number("23")*5);`
- ▶ **isNaN**: Değerin sayı olup olmadığını kontrol eder.
`document.write(isNaN("A"));`
- ▶ **toString**: Nesneleri string ifadeye çevirir. Nesne tanımlamasında özelleştirilebilir.

Nesneler

- ▶ Javascript, nesneye yönelik programlama yapılabilen bir dildir.
- ▶ JS'te sayfadaki her öge bir nesnedir.
- ▶ Örnek olarak "document" bir nesne, "write" ise document sınıfındaki bir metottur.
- ▶ Oluşturulan diziler birer nesnedir.
 - ▶ `var dizi=new Array();`
- ▶ JS'te özel nesnelere oluşturulabilir.
 - ▶ 1.Yöntem: Object ifadesi ile oluşturma
 - ▶ 2.Yöntem: "{...}" işaretleri arasında oluşturma
 - ▶ 3.Yöntem: fonksiyon kullanarak oluşturma

1.Yöntem: `new Object()` ile Nesne Oluşturma

- ▶ `var nesne_adi=new Object();` ile tanımlama yapılır.
- ▶ Tanımlama yapıldıktan sonra, özellikler değer atanarak belirlenir.
 - ▶ Özellikler, `nesne_adi.özellik_adi=değer;` şeklinde tanımlanabilir.
 - ▶ Özellikler, `nesne_adi["özellik_adi"]=değer;` şeklinde tanımlanabilir.
- ▶ Nesne içerisinde fonksiyon tanımlaması yapılabilir.

```
var nesne1=new Object();
nesne1.özellik1="abc";
nesne1["özellik2"]="123";
nesne1.toString=function(){return this.özellik1+this.özellik2;}
document.write(nesne1);
```

II.Yöntem: "{...}" ile Nesne Oluşturma

- ▶ `var nesne_adi={özellikler ve fonksiyonlar}` ile tanımlama yapılır.
- ▶ Tanımlama yapılırken özellikler ve fonksiyonlar "{...}" işaretleri arasında belirtilir.
- ▶ Özellikler, `özellik_adi:değer`, ... şeklinde tanımlanır.
- ▶ Nesne içerisinde fonksiyon tanımlaması yapılabilir.

```
var nesne1={  
    özellik1:"abc",  
    özellik2:"123",  
    toString:function(){return this.özellik1+this.özellik2;}  
};
```

```
nesne1.özellik1="def";  
nesne1["özellik2"]="456";  
document.write(nesne1);
```

III.Yöntem: "function" ile Nesne Oluşturma

- ▶ `function nesne_adi(özellik_değer_parametreleri){ özellikler...}` ile tanımlama yapılır.
- ▶ Tanımlama yapılırken özellikler ve fonksiyonlar "`{...}`" işaretleri arasında belirtilir.
- ▶ Özellikler, `özellik_adi=değer`, ... şeklinde tanımlanır.
- ▶ Nesne içerisinde fonksiyon tanımlaması yapılabilir.

```
function Nesne() //function Nesne(x,y)
{
    this.özellik1=""; //this.özellik1=x;
    this.özellik2=""; //this.özellik2=y;
    this.toString=function(){return this.özellik1+this.özellik2;}
}
```

```
var nesne1=new Nesne();
nesne1.özellik1="abc";
nesne1.özellik2="123";
document.write(nesne1);
```

Özellik veya Fonksiyon Ekleme ve Silme

► Özellik ekleme:

```
nesne1.özellik3="xyz";
```

► Fonksiyon ekleme:

```
nesne1.toString=function(){return this.özellik1+this.özellik2+this.özellik3;}  
document.write(nesne1);
```

► Özellik veya fonksiyon silme:

```
delete nesne1.özellik2;  
delete nesne1.toString;
```

► Özelliğin veya fonksiyonun varlığını kontrol etme:

```
document.write("özellik1" in nesne1);
```

► Örnek:

```
function Nesne()  
{  
    this.özellik1="";  
    this.özellik2="";  
    // this.toString=function(){return this.özellik1+this.özellik2;}  
}
```

```
var nesne1=new Nesne();  
nesne1.özellik1="abc";  
nesne1.özellik2="123";  
nesne1.özellik3="xyz";
```

```
delete nesne1.özellik2;  
delete nesne1.toString;
```

```
document.write(nesne1);
```

```
document.write("özellik1" in nesne1);
```


Nesne Özelliklerini Listeleme

```
function Nesne(x,y,z,t)
{
  this.ozellik1="";
  this.ozellik2=0;
  this.ozellik3=true;
  this.fonksiyon1=function(){document.write("mesaj");}
}
```

```
var nesne1=new Nesne();
for(var eleman in nesne1)
{
  document.write(eleman+": "+nesne1[eleman]);
}
```

Prototype

- ▶ JS nesnelere yeni bir özellik veya metod tanımlamak amacıyla kullanılır.
- ▶ JS nesnelere mevcut özellik veya metodlarının yeniden tanımlanması için kullanılır.

```
function ilkHarfler(x)
{
  var s=this.split(x);
  var s2="";
  for(var i in s) s2+=s[i][0];
  return s2;
}
```

```
String.prototype.ilkHarfler=ilkHarfler;
String.prototype.toUpperCase=function(){ return "XYZ";}
```

```
var ss="Celal Bayar Üniversitesi";
document.write(ss.ilkHarfler(" ")); //CBÜ
document.write(ss.toUpperCase()); //XYZ
```

String

- ▶ **length**: String ifadenin uzunluğunu verir.
`var s="Akhisar"; document.write(s.length); //7`
- ▶ **charAt**: String ifade içerisinde belirtilen konumdaki karakteri verir.
`var s="Akhisar"; document.write(s.charAt(2)); //h`
- ▶ **charCodeAt**: String ifade içerisinde belirtilen konumdaki karakterin UNICODE değerini verir.
`var s="Akhisar"; document.write(s.charCodeAt(0)); //65`
- ▶ **fromCharCode**: Belirtilen ASCII kodunun karakter karşılığını verir.
`document.write(String.fromCharCode(65,66,67)); //ABC`
- ▶ **indexOf**: Belirtilen metin parçasının konumunu belirtir.
`var s="akhisar"; document.write(s.indexOf("a")); //0`
- ▶ **lastIndexOf**: Belirtilen metin parçasının son konumunu belirtir.
`var s="akhisar"; document.write(s.lastIndexOf("a")); //5`
- ▶ **link**: HTML bağlantısı oluşturur.
`var s="akhisar";
document.write(s.link("http://www.akhisar.gov.tr"));`

String

- ▶ **replace**: String ifadenin bir kısmını değiştirir.
`document.write(s.replace("hi","xy")); //akxysar`
- ▶ **search**: String ifade içerisinde aranan metnin konumunu verir.
`document.write(s.search("hi")); //2`
- ▶ **slice**: String ifadenin bir kısmını keser.
`document.write(s.slice(2,4)); //hi`
- ▶ **split**: String ifadeyi belirtilen karakterlerden parçalara ayırır. Dizi olarak parçaları verir.
`var s="Celal Bayar Üniversitesi";`
`var s2=s.split(" ");`
`for(var x in s2) document.write(s2[x][0]); //CBÜ`
- ▶ **concat**: String ifadeleri birleştirir.
`var s1="Akhisar";`
`var s2="MYO";`
`document.write(s1.concat(s2)); //Akhisar MYO`

String

- ▶ **sub**: String ifadeyi alt simge olarak gösterir.
`var s1="2", s2="4"; document.write("H"+s1.sub()+"SO"+s2.sub()); //H2SO4`
- ▶ **sup**: String ifadeyi üst simge olarak gösterir.
`var s1="2", s2="4"; document.write("H"+s1.sup()+"SO"+s2.sup()); // H2SO4`
- ▶ **substr**: Belirtilen konumdan itibaren, belirtilen miktarda karakteri verir.
`var s="Akhisar"; document.write(s.substr(2,4)); //hisa`
- ▶ **substring**: Belirtilen iki konum arasındaki karakterleri verir
`var s="Akhisar"; document.write(s.substring(2,4)); //hi`
- ▶ **toLowerCase**: String ifadeyi küçük harflere dönüştürür.
`var s="AKHISAR"; document.write(s.toLowerCase()); //akhisar`
- ▶ **toUpperCase**: String ifadeyi büyük harflere dönüştürür.
`var s=" akhisar "; document.write(s.toUpperCase()); // AKHISAR`
- ▶ **match**: Düzenli ifade eşleşen string parçalarını dizi olarak geri döndürür.

Math

Özellik	Açıklama
E	2.718 değerini verir
LN2	2'nin logaritmasını verir
LN10	10'nun logaritmasını verir
LOG2E	E'nin logaritmasını verir
LOG10E	10'nun logaritmasını verir
PI	Pi sayısını verir
SQRT1_2	½'nin karekökünü verir
SQRT2	2'nin karekökünü verir

Math

Metot	Açıklama
<code>abs</code>	Sayının mutlak değeri verir
<code>ceil</code>	Ondalık sayıyı en yakın üst sayıya yuvarlar
<code>floor</code>	Ondalık sayının tam kısmını verir.
<code>round</code>	Ondalık sayıyı üste veya alta yuvarlar
<code>max</code>	2 sayıdan büyük olanı verir
<code>min</code>	2 sayıdan küçük olanı verir
<code>pow</code>	A sayısının B. kuvvetini hesaplar
<code>random</code>	0 ile 1 arasında rastgele sayı üretir
<code>sqrt</code>	Sayının karekökünü hesaplar

Date

<code>date</code>	Geçerli tarih veya belirtilen tarih üzerinde işlem yapılmasını sağlayan nesne tanımlanır.
<code>getDate</code>	Ay içerisindeki gün (1-31)
<code>getDay</code>	Hafta içerisindeki gün (0-6)
<code>getFullYear</code>	Yıl (0-2013)
<code>getMonth</code>	Ay (0-11)
<code>getYear</code>	Yıl (1900-2013)
<code>getHours</code>	Saat (0-23)
<code>getMinutes</code>	Dakika (0-59)
<code>getSeconds</code>	Saniye (0-59)
<code>getMilliseconds</code>	Mili Saniye (0-999)
<code>getTime</code>	1 Ocak 1970 tarihinden itibaren geçen süre (ms)
<code>toLocaleString</code>	Tarih ve Saati yerel formata göre gösterir
<code>toDatestring</code>	Tarihi genel formata göre gösterir
<code>toLocaleDateString</code>	Tarihi yerel formata göre gösterir
<code>toTimeString</code>	Saati genel formata göre gösterir
<code>toLocaleTimeString</code>	Saati yerel formata göre gösterir

Date

```
var tarih=new Date();  
document.write("<br>getDate:"+tarih.getDate());  
document.write("<br>getDay:"+tarih.getDay());  
document.write("<br>getFullYear:"+tarih.getFullYear());  
document.write("<br>getMonth:"+tarih.getMonth());  
document.write("<br>getYear:"+tarih.getYear());  
  
document.write("<br>getHours:"+ tarih.getHours());  
document.write("<br>getMinutes:"+ tarih.getMinutes());  
document.write("<br>getSeconds:"+ tarih.getSeconds());  
document.write("<br>getMilliseconds:"+ tarih.getMilliseconds());  
document.write("<br>getTime:"+ tarih.getTime());  
  
document.write("<br>toLocaleString:"+ tarih.toLocaleString());  
document.write("<br>toDateString:"+ tarih.toDateString());  
document.write("<br>toTimeString:"+ tarih.toTimeString());  
document.write("<br>toLocaleTimeString:"+ tarih.toLocaleTimeString());
```

Number

Özellik/Metot	Açıklama
MAX_VALUE	En büyük sayıyı verir. 1.79×10^{308}
MIN_VALUE	En küçük sayıyı verir. (5×10^{-324})
toFixed	<p>Ondalık sayılarda virgülden sonraki basamak sayısını düzenler.</p> <pre>x=123.456789; x.toFixed(2); --> 123.45</pre>
toPrecision	<p>Ondalık sayılarda toplam basamak sayısını düzenler.</p> <pre>x=123.456789; x.toPrecision(4); --> 123.4</pre>

RegExp (Düzenli ifade)

- ▶ Düzenli ifadeler olarak çevrilen (**Regular Expression**); verilen string ifadelerin bölünmesi, karşılaştırılması, ayrılması gibi değişik formatların uygulanmasını sağlayan desenlerdir.
- ▶ Email vb. bilgilerin girişinin kontrol edilmesi sağlanır.
- ▶ Kullanım alanları:
 - ▶ Arama motorları
 - ▶ Veri doğrulama
 - ▶ Sözdizimi doğrulama
 - ▶ Kelime işleme
 - ▶ Bilgi edinme
- ▶ RegExp nesnesi iki kısımdan oluşur.
 - ▶ Kullanıcı tarafından girilecek bilgiye uygulanacak desen
 - ▶ Oluşturulan desene yapılacak düzenlemeler
 - ▶ Düzenli ifade = / desen / ayarlar
 - ▶ Ayarlar:
 - ▶ g: Düzenli ifadenin her tarafında eşleme aranır.
 - ▶ i: Eşleşmelerde büyük-küçük harf duyarlılığının olmaması için kullanılır.
 - ▶ m: Çok satırlı metinlerde düzenli ifade eşleşmesi aramak için kullanılır.

RegExp (Düzenli ifade)

- ▶ Desen ve ayarlar sınırlayıcı bir karakter içerisine alınması gerekir.
- ▶ Genellikler "/" işareti kullanılır.
- ▶ **ESCAPE:** String ifade de özel karakter aranıyorsa, karakterin önüne "\" işareti eklenir.

\n	Yeni satır (newline)
\r	Satırbaşı
\t	Sekme (Tab) karakteri
\v	Dikey tab karakteri
\0	Null karakteri

RegExp (Düzenli ifade) – Özellikler ve Metotlar

Özellik/Metot	Açıklama
<code>global</code>	String ifadenin tamamının karşılaştırılması durumunu kontrol eder.
<code>ignoreCase</code>	String ifadedeki küçük-büyük harf ayrımının dikkate alınıp alınmayacağını kontrol eder.
<code>exec</code>	Düzenli ifade ile eşleşmeyi kontrol eder, eşleşenler dizi olarak geri döndürülür
<code>test</code>	Düzenli ifade ile eşleşmeyi kontrol eder. (true/false)

RegExp (Düzenli ifade) – Özel Karakterler

\	Belirtilen karakteri ifade içinde arar. s="Javascript!"; x=/!/g; document.write(x.test(s)); --> true
^	Verilen ifadenin satırın başında yer alıp almadığını kontrol eder s="Javascript!"; x=/^J/g; document.write(x.test(s)); --> true
\$	Verilen ifadenin satırın sonunda yer alıp almadığını kontrol eder. s="Javascript!"; x=/!\$/g; document.write(x.test(s)); --> true
.	Herhangi bir karakteri temsil eder. s="Javascript!"; x=./g; document.write(x.test(s));
\b	Sonrasında gelen karakterleri kelime içerisinde arar. s="!Javascript!"; x=/\b!/g; document.write(x.test(s)); -->true
\B	Sonrasında gelen karakterleri kelime başında ve sonunda arar. s="!Javascript!"; x=/\B!/g; document.write(x.test(s)); -->true
?=	Aranacak ifade ile düzenli ifadenin birlikte geçenlerini döndürür. s="Javascript dili"; x=/pt(?= dil)/g;
?!	Aranacak ifade ile düzenli ifadenin birlikte geçmeyenlerini döndürür. s="Javascript dili"; x=/ja(?! dil)/g;

RegExp (Düzenli ifade) – Özel Karakterler

X Y	X veya Y karakterleri ile karşılaştırma <code>s="mehmetdikmen.net"; x=/com net/g; document.write(x.test(s));</code>
\d	Rakamları temsil eder. <code>s="a1"; x=/\d/g; document.write(x.test(s));</code>
\D	Rakamların haricindeki karakterleri temsil eder. <code>s="a1"; x=/\D/g; document.write(x.test(s));</code>
\s	Boşlukları temsil eder. <code>s="a b"; x=/\s/g; document.write(x.test(s));</code>
\S	Boşluklar hariç diğer karakterleri temsil eder. <code>s="a b"; x=/\S/g; document.write(x.test(s));</code>
\w	Harf, rakam, alt çizgi karakterleri temsil eder. <code>s="a b"; x=/\w/g; document.write(x.test(s));</code>
\W	Harf, rakam, alt çizgi dışındaki karakterleri temsil eder. <code>s="a%b"; x=/\W/g; document.write(x.test(s));</code>

RegExp (Düzenli ifade) – Özel Karakterler

[ABC]	Köşeli parantezler içerisindeki karakterlerin varlığını test eder. s="Javascript123"; x=/[as3]/g; document.write(x.test(s));
[A-Z]	Belirtilen karakterlerin arasındakileri test eder. s="Javascript123"; x=/[as3]/g; document.write(x.test(s));
[^ABC]	Köşeli parantezler içerisindeki karakterlerin dışındakileri test eder. s="Javascript123"; x=/[as3]/g; document.write(x.test(s));
[^A-Z]	Belirtilen karakterler aralığının dışındakileri test eder. s="Javascript123"; x=/[as3]/g; document.write(x.test(s));
{x}	Öncesinde gelen karakterlerin x kadar tekrarlanmasını kontrol eder. s="Jaavascript123"; x=/a{2}/g; document.write(x.test(s));
{x,}	Öncesinde gelen karakterlerin en az x kadar tekrarlanmasını kontrol eder. s="Jaavaascript123"; x=/a{2,}/g; document.write(x.test(s));
{x,y}	Öncesinde gelen karakterlerin en az x, en fazla y kadar tekrarlanmasını kontrol eder. s="Jaavaascript123"; x=/a{2,3}/g; document.write(x.test(s));

RegExp (Düzenli ifade) – Özel Karakterler

+ Öncesindeki karakterin 1 veya daha fazla tekrarlanacağını
`s="html,html,html"; x=/ml+/g; document.write(s.match(x));`

? Öncesindeki karakterin 0 veya 1 defa tekrarlanacağını belirtir.
`s="html,html,html"; x=/ml?/g; document.write(s.match(x));`

***** Öncesindeki karakterin 0 veya daha fazla tekrarlanacağını belirtir.
`s="html,html,html"; x=/ml*/g; document.write(s.match(x));`

() Parantez içindeki ifade aranır.
`s="html,html,html"; x=/ (ml) /g; document.write(s.match(x));`

RegExp (Düzenli ifade) - Örnek

- ▶ `var s1="_a1";`
- ▶ `var s2="0(555) 555 55 55";`
- ▶ `var s3="abc@xyz.com.tr";`
- ▶ `var s4="11-03-2013";`

- ▶ `var x1=/\w{0,}\d{1,}/g;`
- ▶ `var x2=/0(5[0-9]{2}\s){3}\s[0-9]{2}/g;`
- ▶ `var x3=/^[^0-9][a-zA-Z0-9_]+([\.[a-zA-Z0-9_]+)*@[a-zA-Z0-9_]+([\.[a-zA-Z0-9_]+)*\.[a-zA-Z]{2,4}$/g;`
- ▶ `var x4=/^\d{1,2}[\V.-]\d{1,2}[\V.-]\d{2,4}$/g;`

- ▶ `document.write(x1.test(s1));`
- ▶ `document.write(x2.test(s2));`
- ▶ `document.write(x3.test(s3));`
- ▶ `document.write(x4.test(s4));`

Tarayıcı Nesneleri – Window

- ▶ DOM'un en üstteki nesnesidir. Web sayfasının bulunduğu penceredir.
- ▶ En üstte olduğu için document, navigatör, screen, location, history gibi nesnelere içine alır.
- ▶ Metotlarda "window" yazılmasına gerek yoktur.
- ▶ **screenX**: Ekran ile tarayıcının sol kenarı arasındaki değeri piksel cinsinden verir.
- ▶ **screenY**: Ekran ile tarayıcının üst kenarı arasındaki değeri piksel cinsinden verir.

```
function goster1(){ document.title=window.screenX+" "+window.screenY; }  
<input type="button" name="göster1" value="göster1" onclick="goster1()" />
```

- ▶ **print**: Web sayfasının içeriğini yazıcıya döken arayüzü açar. Gerekli ayarlar yapıldıktan sonra sayfa yazdırılabilir.
window.print();

Window

- ▶ **resizeBy**: Pencerenin yüksekliğinin ve genişliğinin belirtilen miktarda değiştirilmesi.
`window.resizeBy(250, 200);`
- ▶ **resizeTo**: Pencerenin genişlik ve yüksekliğinin belirtilen miktarda boyutlandırılması.
`window.resizeTo(500, 450);`
- ▶ **moveBy**: Pencerenin yatay ve dikey ekseninde taşınması.
`window.moveBy(50, 150);`
- ▶ **moveTo**: Pencerenin belirtilen konuma taşınması.
`window.moveTo(50, 150);`
- ▶ **scrollBy**: Kaydırma çubuklarının mevcut konumlarından belirtilen miktar kadar kaydırılmasını sağlar. Yatay veya dikey olarak artırma/azaltma yapılabilir.
`window.scrollBy(0, 100);`
- ▶ **scrollTo**: Kaydırma çubuklarının belirtilen koordinantlara konumlandırılmasını sağlar.
 - ▶ `window.scrollTo(50, 150);`

Window

- ▶ **alert**: Bilgi/uyarı vermek amacıyla kullanılan mesaj kutularıdır.
`alert("Mesaj kutusu...");`
- ▶ **prompt**: Kullanıcının bilgi girişi yapması istenilen penceredir.
`var x=prompt("Bilgi gir:","varsayılan değer");`
`alert("Girilen değer:"+x);`
- ▶ **confirm**: Evet/Hayır, True/False gibi alternatifli sorular sorularak cevaplar alınmasını sağlar.
`var x=confirm("Soru");`
`alert("Cevap:"+x);`

Window

- ▶ **open**: Belirtilen URL adresini girilen özelliklere göre yeni pencerede açar.
`var pencere=window.open("http://localhost");`
- ▶ `window.open("URL","Konum","Özellikler");`
 - ▶ URL: açılacak bağlantı belirtilir.
 - ▶ Konum: pencerenin nasıl açılacağı belirtilir.
 - ▶ `_blank`: Yeni pencerede açar.
 - ▶ `_self`: Geçerli pencerede açar.
 - ▶ Özellikler:
 - ▶ `directories`: yer imleri çubuğu durumu. (yes/no, 1/0)
 - ▶ `fullscreen`: Tam ekran modu. (yes/no, 1/0)
 - ▶ `height`: Pencere yüksekliği. (piksel)
 - ▶ `width`: Pencere genişliği
 - ▶ `left`: Pencerenin soldan konumu
 - ▶ `top`: Pencerenin üstten konumu
 - ▶ `location`: Adres satırının durumu (yes/no, 1/0)
 - ▶ `menubar`: Tarayıcı menü çubuğu durumu
 - ▶ `resizable`: Boyutlandırılabilme durumu
 - ▶ `scrollbars`: Kaydırma çubuklarının gösterim durumu
 - ▶ `titlebar`: Başlık çubuğunun gösterim durumu
 - ▶ `toolbar`: İleri-geri düğmelerinin durumu
 - ▶ `status`: Durum çubuğunun gösterim durumu
- ▶ **close**: Open ile açılan pencereyi kapatır.

Window

- ▶ **setInterval**: Belirtilen fonksiyonun veya kodun belirtilen süre sonunda tekrar çalıştırılmasını sağlar.
- ▶ **clearInterval**: SETINTERVAL ile başlatılan işlemi sonlandırır.

```
<script>
var x;
var i=-1;
var resimler=new Array("1.jpg","2.jpg","3.jpg","4.jpg");
function islem(){
    i++;
    i%=resimler.length;
    document.getElementById("img1").src=resimler[i];
    document.getElementById("span1").innerHTML=resimler[i];
}
function basla(){ x=window.setInterval("islem()", 1000); }
function dur(){ window.clearInterval(x);}
</script>
```

```
<input type="button" value="Basla" onclick="basla();" />
<input type="button" value="Dur" onclick="dur();" />
<br><img id="img1" width="320" height="240"/>
<br><span id="span1"></span>
```

Window

- ▶ **setTimeout**: Belirtilen fonksiyonun, belirtilen süre sonunda bir defa çalıştırılmasını sağlar.
- ▶ **clearTimeout**: SETTIMEOUT ile başlatılan işlemi sonlandırır.

```
<script>
var x;
var i=-1;
var resimler=new Array("1.jpg","2.jpg","3.jpg","4.jpg");
function islem(){
    i++;
    i%=resimler.length;
    document.getElementById("img1").src=resimler[i];
    document.getElementById("span1").innerHTML=resimler[i];
    x=window.setTimeout("islem()", 1000);
}
function basla(){ islem();}
function dur(){ window.clearTimeout(x);}
</script>
```

```
<input type="button" value="Basla" onclick="basla();" />
<input type="button" value="Dur" onclick="dur();" />
<br><img id="img1" width="320" height="240"/>
<br><span id="span1"></span>
```


Document

- ▶ HTML sayfasında bulunan bütün nesnelere temelini oluşturur.
- ▶ Sayfa özelliklerine, sayfada bulunan diğer nesnelere erişmek için kullanılır.
 - ▶ `write/writeline`: Sayfa üzerine yazdırma metodudur.
 - ▶ `characterSet`: Sayfanın karakter setini gösterir.
 - ▶ `domain`: Alan adını gösterir.
 - ▶ `title`: Sayfa başlık bilgisini gösterir.
 - ▶ `referrer`: Önceki sayfanın adresini gösterir.
 - ▶ `URL`: Mevcut sayfanın adresini gösterir.
 - ▶ `width-height`: Genişlik ve yükseklik
 - ▶ `lastModified`: Son değiştirilme tarihini gösterir.

- ▶ `document.write("
characterSet:"+document.characterSet);`
- ▶ `document.write("
domain:"+document.domain);`
- ▶ `document.write("
title:"+document.title);`
- ▶ `document.write("
referrer:"+document.referrer);`
- ▶ `document.write("
URL:"+document.URL);`
- ▶ `document.write("
width:"+document.width);`
- ▶ `document.write("
height:"+document.height);`
- ▶ `document.write("
lastModified:"+document.lastModified);`

Document

```
<form name="form1">  
  <input type="text" name="txt1" />  
  <input type="text" name="txt2" />  
</form>
```

```
<a href="#" name="link1">link1</a>  
<a href="#" name="link2">link2</a>
```

```
  
  

```

```
<script>  
  document.write("<br>Form Sayısı: "+document.forms.length);  
  document.write("<br>Link Sayısı: "+document.links.length);  
  document.write("<br>Resim Sayısı: "+document.images.length);  
  
  document.write("<br>Form1 eleman sayısı : "+document.forms[0].length);  
  document.write("<br>Form1 1.alt eleman : "+document.forms[0][1].name);  
</script>
```

Document

- ▶ **getElementById**: Verilen id numarasına göre ilgili nesneye ve özelliklere erişilir.
- ▶ **getElementsByName**: Verilen <name> özelliğine göre ilgili nesneye ve özelliklerine ulaşılır. Nesne özelliklerine doğrudan ulaşamaz, item özelliğinde indeks numarası belirtilmelidir.
- ▶ **getElementsByTagName**: HTML içerisinde belirtilen etikette olan tüm nesnelerin listesini verir.

```
<input type="text" name="txt1" id="txt1" value="metin kutusu1" />  
<input type="text" name="txt1" id="txt2" value="metin kutusu2" />  
<input type="text" name="txt2" id="txt3" value="metin kutusu3" />  
<input type="text" name="txt2" id="txt4" value="metin kutusu4" />
```

```
<script>  
  document.getElementById("txt1").style.backgroundColor="#008";  
  var etiketler1=document.getElementsByName("txt1");  
  for(var i=0; i<etiketler1.length;i++)  
    etiketler1.item(i).style.color="#ff0";  
  var etiketler2=document.getElementsByTagName("input");  
  for(var i=0; i<etiketler2.length;i++)  
    document.write("<br>" +etiketler2.item(i).value);  
</script>
```

Document

- ▶ **createElement**: HTML elementinin oluşturulmasını sağlar.
- ▶ **createTextNode**: Elementlerin içerisine metin ekler.

```
var table=document.createElement("table");
var tbody=document.createElement("tbody");
for(var i=1; i<5;i++)
{
    var tr=document.createElement("tr");
    for(var j=1; j<4; j++)
    {
        var td=document.createElement("td");
        var text=document.createTextNode(i+"sattır, "+j+"sütun");
        td.appendChild(text);
        tr.appendChild(td);
    }
    tbody.appendChild(tr);
}
table.appendChild(tbody);
table.setAttribute("border","1");
document.getElementsByTagName("body")[0].appendChild(table);
```

Navigator

appName	Tarayıcıismi
appVersion	Tarayıcıversiyonu
browserLanguage	Kullanılantarayıcıdili
cookieEnabled	Çerez(cookie)oluşturmadurumu
javaEnabled()	Javanınaktiflikdurumu
onLine	İnternetbağlantısıdurumu
platform	Tarayıcınınyüklediğiortam
plugins	Eklentiler
systemLanguage	Sistem dili
userAgent	Tarayıcı kod ve versiyonu
userLanguage	Kullanıcı dili

```
document.write("<br>appName:"+navigator.appName);
document.write("<br>appVersion:"+navigator.appVersion);
document.write("<br>browserLanguage:"+navigator.browserLanguage);
document.write("<br>cookieEnabled:"+navigator.cookieEnabled);
document.write("<br>javaEnabled:"+navigator.javaEnabled());
document.write("<br>onLine:"+navigator.onLine);
document.write("<br>platform:"+navigator.platform);
document.write("<br>plugins:"+navigator.plugins);
document.write("<br>systemLanguage:"+navigator.systemLanguage);
document.write("<br>userAgent:"+navigator.userAgent);
document.write("<br>userLanguage:"+navigator.userLanguage);
```

Screen

<code>width</code>	Ekran genişliği
<code>height</code>	Ekran yüksekliği
<code>availHeight</code>	Kullanılabilir ekran yüksekliği
<code>availWidth</code>	Kullanılabilir ekran genişliği
<code>availLeft</code>	Tarayıcının yerleşebileceği uzaklık (soldan)
<code>availTop</code>	Tarayıcının yerleşebileceği uzaklık (üstten)
<code>colorDepth</code>	Ekran renk derinliği

```
document.write("<br>width:"+screen.width);  
document.write("<br>height:"+screen.height);  
document.write("<br>availHeight:"+screen.availHeight);  
document.write("<br>availWidth:"+screen.availWidth);  
document.write("<br>availLeft:"+screen.availLeft);  
document.write("<br>availTop:"+screen.availTop);  
document.write("<br>colorDepth:"+screen.colorDepth);
```

History

- ▶ **length**: Ziyaret edilen sayfa sayısını verir.
- ▶ **back**: Bir önceki sayfaya gider.
- ▶ **forward**: Bir sonraki sayfaya gider.
- ▶ **go**: Ziyaret edilen herhangi bir sayfaya gider.

```
<script>
  function geri(){window.history.back();}
  function ileri(){window.history.forward();}
  function git(){window.history.go(-2);}
  document.writeln("<<Ziyaret edilen sayfa sayısı:"+window.history.length);
</script>
```

```
<input type="button" value="Geri" onclick="geri()" />
<input type="button" value="İleri" onclick="ileri()" />
<input type="button" value="2 Geri Git" onclick="git()" />
```

History

- Geri tuşunu iptal eden JS kodlarını

```
<body onload=" geritusiptal()">  
<script>  
    function geritusiptal(){ window.history.forward(); }  
    setTimeout(" geritusiptal()", 0);  
</script>  
  
<a href="deneme1.html">Deneme sayfası</a>
```


Location

- ▶ Adres çubuğundaki URL bilgilerinin tutulduğu nesnedir.
 - ▶ **hash**: «#» sonrasındaki bilgiyi verir.
 - ▶ **host**: Sunucu adı ve port numarasını verir.
 - ▶ **hostname**: port bilgisi olmadan sadece sunucu adını verir.
 - ▶ **href**: URL adresinin tamamını verir.
 - ▶ **pathname**: Yol ve dosya bilgisini verir.
 - ▶ **protocol**: Bağlantı protokolünü verir. (http, https, ftp vb.)
 - ▶ **search**: URL'deki «?» işaretinden sonrasını verir.

```
document.write("<br>hash:"+window.location.hash);  
document.write("<br>host:"+window.location.host);  
document.write("<br>hostname:"+window.location.hostname);  
document.write("<br>href:"+window.location.href);  
document.write("<br>pathname:"+window.location.pathname);  
document.write("<br>protocol:"+window.location.protocol);  
document.write("<br>search:"+window.location.search);
```

Location

- ▶ toString : Location nesnesinin içerdiği URL adresini verir.
- ▶ assign : Yeni bir URL adresi yükler.
Geri tuşu aktiftir. Önceki sayfalar geçmişte tutulur.
- ▶ reload : Güncel sayfayı tekrar yükler.
- ▶ replace : Güncel sayfayı belirtilen adres ile değiştirir.
Geri tuşu çalışmaz. Önceki sayfa yok olur.

```
<script>
  document.write(window.location.toString());
  function ass(){window.location.assign("location.html?x=2");}
  function rel(){window.location.reload();}
  function rep(){window.location.replace("location.html?x=5");}
</script>
```

```
<input type="button" value="assign" onclick="ass()" />
<input type="button" value="reload" onclick="rel()" />
<input type="button" value="replace" onclick="rep()" />
```

Event – Key (Klavye Olayları)

	KEYPRESS	KEYDOWN/KEYUP
altKey	-	ALT tuşunu kontrol eder.
ctrlKey	-	CTRL tuşunu kontrol eder.
shiftKey	-	SHIFT tuşunu kontrol eder.
charCode	Karakterin ASCII kodunu verir. Küçük-Büyük harf ayrımı vardır. A: 65, a:97	-
keyCode	Karakterin ASCII kodunu verir. Küçük-Büyük harf ayrımı vardır. A: 65, a:97	Basılan tuşun ASCII kodunu verir. Küçük-Büyük harf ayrımı yoktur. A:65, a:65
which		
	Özel tuşların ASCII kodları elde edilmez.	Özel tuşların ASCII kodları elde edilebilir.

Event – Key (Klavye Olaylari)

```

<script>
function press(e)
{
    document.getElementById("KCP").value=e.keyCode;
    document.getElementById("CCP").value=e.charCode;
    document.getElementById("WP").value=e.which;
}
function down(e)
{
    if(e.altKey) document.getElementById("ACS").value="ALT";
    if(e.ctrlKey) document.getElementById("ACS").value="CTRL";
    if(e.shiftKey) document.getElementById("ACS").value="SHIFT";
    document.getElementById("KCD").value=e.keyCode;
    document.getElementById("CCD").value=e.charCode;
    document.getElementById("WD").value=e.which;
}
window.onkeypress=press;
window.onkeydown=down;
</script>

```

```

<br /><input type="text" id="KCP" name="KCP" /><label for="KCP">KeyCodePress</label>
<br /><input type="text" id="CCP" name="CCP" /><label for="CCP">CharCodePress</label>
<br /><input type="text" id="WP" name="WP" /><label for="WP">WhichPress</label>
<br /><input type="text" id="KCD" name="KCD" /><label for="KCD">KeyCodeDown</label>
<br /><input type="text" id="CCD" name="CCD" /><label for="CCD">CharCodeDown</label>
<br /><input type="text" id="WD" name="WD" /><label for="WD">WhichDown</label>
<br /><input type="text" id="ACS" name="ACS" /><label for="ACS">Alt-Ctrl-Shift</label>

```

Event – Key (Klavye Olaylari)

```
<script>
function digit(e)
{
    var key= window.event? window.event.keyCode: e ? e.which:null;
    if((key>=48 && key<=57) || key==8 || key==9 || key==13 || key==27)
        return true;
    else
        return false;
}

function toplama()
{
    var s1=document.getElementById("txt1").value;
    var s2=document.getElementById("txt2").value;
    document.getElementById("txtSonuc").value=parseInt(s1)+parseInt(s2);
}
</script>
```

```
<input type="text" id="txt1" onKeyPress="return digit()">
<input type="text" id="txt2" onKeyPress="return digit()">
<input type="button" value="Topla" onclick="toplama()" />
<input type="text" id="txtSonuc" readonly="readonly">
```

Event – Mouse (Fare Olaylari)

<code>button</code>	Farenin tuşlarını kontrol eder
<code>clientX, clientY</code> <code>x, y</code>	Tarayıcı sayfasında farenin konumu
<code>layerX, layerY</code> (IE hariç) <code>offsetX, offsetY</code> (Firefox hariç)	Katman içerisinde farenin konumu
<code>pageX, pageY</code>	Sayfa üzerinde fare konumu
<code>screenX, screenY</code>	Ekran üzerinde fare konumu

Event – Mouse (Fare Olaylari)

```
<script type="text/javascript">
  window.onmousedown=mousedown;
  window.onmousemove=mousemove;
  function mousedown(e){document.title=e.button;}
  function mousemove(e)
  {
    var s="";
    s+="
```

```
<div id="div1" style="width:400px; height:300px; background:#ff0;
  position:absolute; left:200px; top:200px;"></div>
```

FORM

- ▶ Kullanıcı ile etkileşime girilmesini sağlayan nesnelere bir arada bulunduğu ve girilen bilgilerin sunucuya aktarıldığı DOM nesnesidir.
- ▶ **action**: Formdaki nesnelere ait bilgilerin gönderileceği sayfayı belirtir.
- ▶ **enctype**: Gönderilecek bilgileri şifreleme türüdür.
 - ▶ application/x-www-form-urlencoded
 - ▶ multipart/form-data
 - ▶ text/plain
- ▶ **id**: Form nesnesine ait bir kimliği belirtir.
`id="form1"`
- ▶ **length**: Form nesnesi içindeki nesne sayısını verir.
`document.forms.item(0).length;`
`document.getElementById("form1").length;`
- ▶ **method**: Bilgilerin gönderilme şeklini ifade eder.
 - ▶ GET: Bilgi adres çubuğu ile gönderilir.
 - ▶ POST: Bilgi adres çubuğunda gösterilmeden gönderilir.
- ▶ **name**: Form nesnesinin ismini belirtir.
- ▶ **target**: Action özelliği ile gönderilecek sayfanın nasıl açılacağını belirtir.

FORM

- ▶ **elements**: Form içerisindeki nesnelere indeks numarası ile erişimi sağlar.

```
var form = document.forms.item(0);  
for(i=0; i<=form.length;i++)  
    document.write(form.elements[i].value);
```

- ▶ **reset**: Form içerisindeki nesnelere varsayılan değerlere geri döndürür.

```
document.forms.item(0).reset();  
document.getElementById("form1").reset();
```

- ▶ **submit**: Bilgilerin "action" ile belirtilen sayfaya aktarılmasını sağlar.

```
document.forms.item(0).submit();  
document.getElementById("form1").submit();
```

```
<form action="hedef.php" enctype="application/x-www-form-urlencoded"  
method="POST" name="form1" id="form1" target="_self">  
  <input type="text" id="txt1" name="txt1"/>  
  <input type="button" id="btn1" name=" btn1 "/>  
</form>
```

INPUT

- ▶ Klavye veya fare aracılığıyla bilgi girişlerinin yapıldığı nesnelerdir.
- ▶ `type` özelliği ile nesne türü belirtilir.
 - ▶ `text`
 - ▶ `password`
 - ▶ `hidden`
 - ▶ `radio`
 - ▶ `checkbox`
 - ▶ `button`
 - ▶ `submit`
 - ▶ `reset`
 - ▶ `image`
 - ▶ `file`

Input - Text/Password

readonly	Metin kutusunun sadece okunabilir olmasını sağlar.
size	Metin kutusunun karakter sayısını belirler.
value	Metin kutusunun değerini verir.
maxlength	Maksimum karakter sayısını belirtir.
onblur	Metin kutusundan ayrılma durumunda çalıştırılan olaydır.
onchange	Metin kutusunun değişmesi durumunda çalıştırılan olaydır.
onfocus	Metin kutusuna odaklanma durumunda çalıştırılan olaydır.
onkeypress	Metin kutusu üzerindeyken klavyeden tuşa basılması ile çalıştırılan olaydır.
onselect	Metin kutusunun içeriğinin seçilmesi durumunda çalıştırılan olaydır.

```

<script>
function Sec(el)
{
    el.select();
}
function Kontrol(el)
{
    var r=/[a-z]{1,}[A-Z]{1,}[0-9]{1,}/g;
    var txt=el.value;
    if(!r.test(txt))
        alert("Yanlış...");
}
</script>

```

```

<input type="text" name="txt" id="txt" onfocus="Sec(this)" onblur="Kontrol(this)" />
<input type="password" name="pass" id="pass" onfocus="Sec(this)" onblur="Kontrol(this)" />

```

Input – CheckBox

- ▶ Genellikle çoklu seçimlerde kullanılır.
- ▶ Aynı anda birden fazla değer seçilebilir.

```
<script>
function TumunuSec(form)
{
    for(i=0; i<form.length;i++)
        form[i].checked=true;
}
function Secimlptal(form)
{
    for(i=0; i<form.length;i++)
        form[i].checked=false;
}
function DigerSec(form)
{
    for(i=0; i<form.length;i++)
        form[i].checked=!form[i].checked;
}
</script>
```

```
<form name="form1">
<input type="checkbox" name="ch" id="ch1" value="1" />
<input type="checkbox" name="ch" id="ch2" value="2" />
<input type="checkbox" name="ch" id="ch3" value="3" />
```

```
<input type="button" value="Tümünü Seç" onclick="TumunuSec(document.form1.ch)" />
<input type="button" value="Seçimi Kaldır" onclick="Secimlptal(document.form1.ch)" />
<input type="button" value="Diğerlerini Seç" onclick="DigerSec(document.form1.ch)" />
</form>
```

Input – CheckBox Örnek

```
<script>
function kontrol(element) {
  var el = element.name;
  var secilen = 0;
  var ch = document.getElementsByName(el);
  for (var i = 0; i < ch.length; i++)
    if (ch.item(i).checked) secilen++;
  if (secilen > 1)
    alert("Lütfen seçili olan diğer kutunun işaretini kaldırınız.");
  return (secilen > 1 ? false : true);
}
</script>
```

```
<form name="form1">
<input type="checkbox" name="ch" id="ch1" value="1" onclick="return
  kontrol(this)" />
<input type="checkbox" name="ch" id="ch2" value="2" onclick="return
  kontrol(this)" />
<input type="checkbox" name="ch" id="ch3" value="3" onclick="return
  kontrol(this)" />
</form>
```

Input - Radio

- Birden fazla seçenek arasından sadece birisinin seçilmesi amacıyla kullanılır.

```
<script>  
function kontrol(element)  
{  
    alert("Seçilen Değer:"+element.value);  
}  
</script>
```

```
<form name="form1">  
<input type="radio" name="rb" id="rb1" value="1" onclick="return  
    kontrol(this)" />1  
<input type="radio" name="rb" id="rb2" value="2" onclick="return  
    kontrol(this)" />2  
<input type="radio" name="rb" id="rb3" value="3" onclick="return  
    kontrol(this)" />3  
</form>
```

Input – Button, Submit, Reset

- ▶ **Button** : Herhangi bir fonksiyonun çalıştırılması için kullanılır.
- ▶ **Submit** : Form elemanlarına girilen değerleri belirtilen sayfaya gönderir.
- ▶ **Reset** : Form elemanlarına girilen değerleri temizler.

```
<script>
function kontrol(element)
{
    alert("Seçilen Element:"+element.value);
}
</script>
```

```
<form name="form1">
<input type="radio" name="rb" id="rb1" value="1" onclick="return
    kontrol(this)" />1
<input type="radio" name="rb" id="rb2" value="2" onclick="return
    kontrol(this)" />2
<input type="button" value="Kontrol" onclick="kontrol(this)" />
</form>
```

Input – Image, File, Hidden

- ▶ **Image** : Resim gösterme ve resmin seçilen konumuna göre işlem yapılmasını sağlayan elementtir.
 - ▶ Hedef sayfada resmin tıklanan kısmının koordinantları elde edilir.

```
<form name="form1" method="get">  
<input type="image" name="img" id="img" src="1.jpg"/>  
</form>
```

- ▶ **File** : Sunucuya dosya göndermek için kullanılır.
 - ▶ Dosyanın elde edileceği hedef sayfa php, aspx gibi dillerle yazılmış sayfalar olmalıdır.
- ▶ **Hidden** : Değerleri göstermeden hedef sayfaya gönderilmesi için kullanılır.

Select

- ▶ Değer listesinden bir veya birden fazla değer seçilmesi için kullanılır.
 - ▶ Length: Seçenek sayısı
 - ▶ Multiple: Çoklu seçimi belirler
 - ▶ SelectedIndex: Seçilen değer indeks numarasını belirtir
 - ▶ Size: Aynı anda gösterilecek değer sayısıdır.
 - ▶ Seçenek belirtilmezse varsayılan değer "1" dir.
 - ▶ 1 değeri verilirse açılır liste kutusu olarak gösterilir.
 - ▶ 1'den büyük değer verilirse liste kutusu görünümü alır. Çoklu seçim yapılabilir.
 - ▶ Add: Listeye yeni değer ekler
 - ▶ Remove: Listedeki değer siler

```
<select size="1" name="select1">  
<option value="1">Seçenek1</option>  
<option value="2">Seçenek2</option>  
<option value="3">Seçenek3</option>  
</select>
```

```
<select size="3" name="select2" multiple="multiple">  
<option value="1">Seçenek1</option>  
<option value="2">Seçenek2</option>  
<option value="3">Seçenek3</option>  
</select>
```

Select Örnek

```
<script>
function ekle()
{
    var sel=document.getElementById("select1");
    var x1=document.getElementById("txt1");
    var x2=document.getElementById("txt2");
    var opt=document.createElement("option");
    opt.text=x1.value;
    opt.value=x2.value;

    sel.add(opt);
}
function sil()
{
    var sel=document.getElementById("select1");
    sel.remove(sel.selectedIndex);
}
</script>

<select size="1" name="select1" id="select1">
<option value="1">Seçenek1</option>
<option value="2">Seçenek2</option>
<option value="3">Seçenek3</option>
</select>

<input type="text" name="txt1" id="txt1" />
<input type="text" name="txt2" id="txt2" />
<input type="button" value="Ekle" onclick="ekle()" />
<input type="button" value="Sil" onclick="sil()" />
```

Textarea

- ▶ Text kutularından farklı olarak çok satırlı ve uzun metin girişlerinde kullanılır.
 - ▶ Rows/Cols: Satır ve sütun sayısını belirler

```
<script>
function kontrol(e)
{
    var maks=50;
    var metin=document.getElementById("ta");
    var kalan=document.getElementById("kalan");
    if(metin.value.length<=maks)
    {
        kalan.innerHTML=(parseInt(maks)-parseInt(metin.value.length));
        return true;
    }
    else
        return false;
}
</script>
```

```
<textarea name="ta" id="ta" onkeydown="return kontrol()" rows="10"
    cols="100"></textarea>
<span id="kalan"></span>
```



JQuery

Jquery Nedir?

JQuery

- ▶ JQuery istemci tarafında çalışan javascript dilini basitleştirmek için tasarlanmış kütüphanedir.
- ▶ Amacı, bütün web tarayıcılarında çalışabilen, basit yazılan ve hızlı sonuç alınabilen uygulamaları geliştirmektir.
- ▶ İlk olarak 2006 yılının Ocak ayında yayınlanmıştır. (BarCamp)
- ▶ JQuery aşağıdaki özellikleri içerir:
 - ▶ DOM elementlerini kullanarak çapraz tarayıcılar için seçici oluşturma
 - ▶ DOM bağlantılarında gezinti ve değişiklik yapma
 - ▶ DOM düğümlerini kullanarak CSS benzeri seçiciler oluşturma
 - ▶ Olayları oluşturma ve yönetme
 - ▶ Animasyonlar ve efektler oluşturma
 - ▶ Ajax
 - ▶ Eklenti desteği geliştirebilme
 - ▶ Çapraz tarayıcı desteği
- ▶ JQuery dosyasını sayfaya ekleme
 - ▶ `<script src="jquery-1.9.1.min.js"></script>`

Jquery ile ilgili siteler

- ▶ JQuery siteleri:
 - ▶ <http://jquery.com>
 - ▶ <http://jqueryui.com>
- ▶ İndirme linkleri:
 - ▶ <https://developers.google.com/speed/libraries/devguide?hl=tr>
 - ▶ <http://jquery.com/>
 - ▶ <http://jqueryui.com/>
- ▶ Doğrudan kullanım linkleri:
 - ▶ <http://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js>
 - ▶ <http://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/jquery-ui.min.js>
 - ▶ <http://code.jquery.com/jquery-1.12.2.js>
 - ▶ <http://code.jquery.com/jquery-1.12.2.min.js>
 - ▶ <http://code.jquery.com/ui/1.11.4/jquery-ui.js>
 - ▶ <http://code.jquery.com/ui/1.11.4/jquery-ui.min.js>
- ▶ En son versiyonu kullanma:
 - ▶ <http://code.jquery.com/jquery-latest.js>
 - ▶ <http://code.jquery.com/jquery-latest.min.js>
- ▶ JQuery Öğrenme:
 - ▶ <http://learn.jquery.com/>

Jquery ile Çalışma

► Sayfanın tamamı yüklendikten sonra çalışmaya başlamak için aşağıda belirtilen tanımlamalar kullanılmalıdır.

- `$(document).ready(function(e) { ... });`
- `$(function(e) { ... });`
- `$(function(e) { ... });`
- `(function($) { ... })(jQuery);`

```
<script>
    $(document).ready(function(e) {
        $("#div1").text("deneme");
    });
</script>
<div id="div1"></div>
```

jQuery Bildirimi

- ▶ JQuery seçici fonksiyonu «jQuery» veya «\$» ile başlar.
- ▶ jQuery("seçici") veya \$("seçici")

```
<script src="jquery.min.js"></script>
```

```
<script>
  jQuery(document).ready(function(e) {
    alert(jQuery("#p1").text());
    alert($("#p2").text());
  });
```

```
</script>
```

```
<p id="p1">Paragraf1</p>
```

```
<p id="p2">Paragraf2</p>
```


Jquery Seçici

- ▶ CSS seçicilerine benzer.
- ▶ Genel seçici: "*" işareti ile tanımlanır.
 - ▶ * { color:red;} (CSS)
 - ▶ \$("*").css("color","red");
- ▶ Element seçici: HTML elementinin adı yazılır.
<p>paragraf1</p>
CSS : p{ display:none; }
Jquery : \$('p').hide();
- ▶ Kimlik (ID) seçici: Seçiciden önce "#" işareti eklenir.
<p id="p1">paragraf1</p>
CSS : #p1{ display:block; }
Jquery : \$('#p1').show(); veya \$('p#p1').show();
- ▶ Sınıf (Class) seçici: Seçiciden önce "." işareti eklenir.
<p class="p1">paragraf1</p>
CSS : .p1{display:block; }
Jquery : \$('.p1').show(); veya \$('p.p1').show();

Jquery Seçici

- ▶ Çocuk bağıni seçme (children, >):
 - ▶ `#p3>* {color:red;} (CSS)`
 - ▶ `$("#p3").children().css("color","red");`
 - ▶ `$("#p3>*").css("background-color","yellow");`
- ▶ Ebeveyn bağıni seçme (parent):
 - ▶ `$("#span").parent().css("color","yellow");`
- ▶ Kardeş bağıni seçme (siblings, +, ~):
 - ▶ `$("#span").siblings().css("color","yellow");`
 - ▶ `$("#span+i").css("color","yellow");`
 - ▶ `$("#span~i").css("color","yellow");`
- ▶ Birden fazla seçim: Seçicilerin arasına ", " eklenir.
 - ▶ `a,b,c{ color:red; } (CSS)`
 - ▶ `$('#a,b,c,').css("color","red");`
- ▶ Stil tanımlama:
 - ▶ `$('#p').css("background-color","yellow");`
 - ▶ `$('#p1').css({"background-color":"black", "color":"white"});`
 - ▶ `$('#p2').css({backgroundColor:"black", color:"white"});`

Jquery CSS Metotlari

- ▶ *addClass*: Elemente belirtilen CSS sınıfını ekler.
 - ▶ `$("#a").addClass("stil");`
- ▶ *removeClass*: Elementten belirtilen CSS sınıfını kaldırır.
 - ▶ `$("#a").removeClass("stil");`
- ▶ *hasClass*: Belirtilen sınıfın varlığını kontrol eder.
 - ▶ `$("#a").hasClass("stil");`
- ▶ *toggleClass*: Sınıf varsa siler, yoksa ekler.
 - ▶ `$("#a").toggleClass("stil");`

Jquery Dolaşım Seçicileri

- ▶ **find:** CSS seçicilerini kullanarak istenilen elementin seçilmesi sağlanır.
 - ▶ `$("#div1").find("span").animate({left:"100px"},"slow");`
- ▶ **gt(n):** Sıralanmış bağların sırasına göre belirtilen sıra numarasından sonrasını seçer.
 - ▶ `$("#tr:gt(2)").css("color","red");`
- ▶ **lt(n):** Sıralanmış bağların sırasına göre belirtilen sıra numarasından öncesini seçer.
 - ▶ `$("#tr:lt(2)").css("color","red");`
- ▶ **eq(n):** Belirtilen sıra numarasındaki elementi seçer
 - ▶ `$("#tr:eq(2)").css("color","red");`
- ▶ **first:** İlk elementi seçer
 - ▶ `$("#tr:first").css("color","red");`
- ▶ **last:** Son elementi seçer
 - ▶ `$("#tr:last").css("color","red");`
- ▶ **even:** çift elementleri seçer
 - ▶ `$("#tr:even").css("color","red");`
- ▶ **odd:** Tek elementleri seçer
 - ▶ `$("#tr:odd").css("color","red");`

Jquery Dolaşım Seçicileri

- ▶ **contains:** Belirtilen bilginin geçtiği elementleri seçer
 - ▶ `$("#tr:contains(4)").css("color","red");`
- ▶ **has:** Belirtilen elementin geçtiği elementleri seçer
 - ▶ `$("#tr:has(span)").css("color","red");`
- ▶ **not:** Belirtilen elementin dışındakileri elementleri seçer
 - ▶ `$("#td:not(#td4)").css("color","red");`
- ▶ **empty:** Belirtilen elementin dışındakileri elementleri seçer
 - ▶ `$("#td:empty").css("background-color","red");`
- ▶ **next:** Belirtilen elementten sonra gelen elementi seçer
- ▶ **prev:** Belirtilen elementten önce gelen elementi seçer
- ▶ **nextAll:** Belirtilen elementten sonra gelen elementleri seçer
- ▶ **prevAll:** Belirtilen elementten önce gelen elementleri seçer

DOM Bağları Ekleme ve Silme

- ▶ *text*: Seçilen elementin metin değerini elde etmek veya değiştirmek için kullanılır
 - ▶ `$("#p").text();` veya `$("#p").text("yeni metin");`
- ▶ *html*: Seçilen elementin html içeriğini elde etmek veya değiştirmek için kullanılır
 - ▶ `$("#p").html();` veya `$("#p").html("<u>xyz</u>
abc");`
- ▶ *append*: Seçilen elemente yeni bir elementi son çocuk bağı olarak ekler.
 - ▶ `$("#p").append("xyz");`
- ▶ *appendTo*: Belirtilen ifadeyi, belirtilen hedef seçiciye son çocuk bağı olarak ekler.
 - ▶ `$("#xyz").appendTo("#p");`
- ▶ *prepend*: Seçilen elemente yeni bir elementi ilk çocuk bağı olarak ekler
- ▶ *prependTo*: Belirtilen ifadeyi, belirtilen hedef seçiciye ilk çocuk bağı olarak ekler.

DOM Bağları Ekleme ve Silme

- ▶ `after`: Seçilen elementin sonrasına yeni bir element ekler
- ▶ `before`: Seçilen elementin öncesine yeni bir element ekler
- ▶ `insertAfter`: Belirtilen ifadeyi, belirtilen hedef elementin sonrasına ekler
- ▶ `insertBefore`: Belirtilen ifadeyi, belirtilen hedef elementin öncesine ekler
- ▶ `replaceAll/replaceWith`: Seçilen elementleri belirtilen yeni element ile değiştirir
- ▶ `remove`: Belirtilen elementi siler
- ▶ `empty`: belirtilen elementin tüm çocuk bağlarını siler
- ▶ `detach`: `remove` gibidir. Farkı silinen hafızada kalır
- ▶ `clone`: Seçilen elementi çoğaltır.
- ▶ `wrap`: seçilen elementlerin etrafını HTML elementleri ile çevreler
- ▶ `wrapAll`: Seçilen elementlerin tamamını çevreler
- ▶ `wrapInner`: Seçilen elementlerin içeriğini çevreler
- ▶ `attr`: Elementin özellik bilgisini elde etme veya değiştirme
- ▶ `removeAttr`: Elementin belirtilen özelliğini siler

DOM Bağları Ekleme ve Silme

```
$(document).ready(function(e) {  
    $("#p1").text("yeni paragraf metni");  
    $("#p1").html("<b>yeni html içerik</b>");  
    $("#p1").append("<u>son kısmına eklenen içerik</u>");  
    $("#p1").prepend("<u>ilk kısmına eklenen içerik</u>");  
   ("<i>son kısmına eklenen 2.içerik</i>").appendTo("#p1");  
   ("<i>ilk kısmına eklenen 2.içerik</i>").prependTo("#p1");  
    $("#p1").after("<font color=red>sonrasına eklenen</font>");  
    $("#p1").before("<font color=red>öncesine eklenen</font>");  
   ("<b>replace all ile deęiřti</b>").replaceAll("#p1");  
   ("&b").replaceWith("<p id='p1'>replace with ile deęiřti</p>");  
    $("#p1").clone().appendTo("body");  
    $("#p1").clone().appendTo("body");  
    $("#p1").remove();  
    $("#p1").detach();  
    $("#p1").wrap("<div style='border:1px solid #f00;'></div>");  
    $("#img").attr("width","500px").animate({width:"100px"},2000);  
    $("#img").removeAttr("width");  
});
```


Show, Hide, Toggle, Delay

- ▶ Show : Nesneyi bir defa gösterir.
 - ▶ `$('#seçici').show([süre],[efekt], [fonksiyon]); //jquery`
 - ▶ `$('#seçici').show([efekt], [süre], [fonksiyon]); //jqueryui`
- ▶ Hide: Nesneyi bir defa geçici olarak gizler.
 - ▶ `$('#seçici').hide([süre],[efekt], [fonksiyon]); //jquery`
 - ▶ `$('#seçici').hide([efekt], [süre], [fonksiyon]); //jqueryui`
- ▶ Toggle: Nesneyi gösterir veya gizler.
 - ▶ `$('#seçici').toggle([süre],[efekt], [fonksiyon]); //jquery`
 - ▶ `$('#seçici').toggle([efekt], [süre], [fonksiyon]); //jqueryui`
- ▶ Delay: Animasyonu belirli bir süre bekletir.
 - ▶ `$('#seçici').delay([süre]);`
- ▶ Efekt:
 - ▶ jquery: swing/linear
 - ▶ JQueryui: blind, bounce, clip, drop, explode, fold, highlight, puff, pulsate, scale, shake, size, slide
- ▶ Süre : slow, fast, milisaniye cinsinden değer

Show

```
<script src="jquery.min.js"></script>
<script src="jquery-ui.min.js"></script>
```

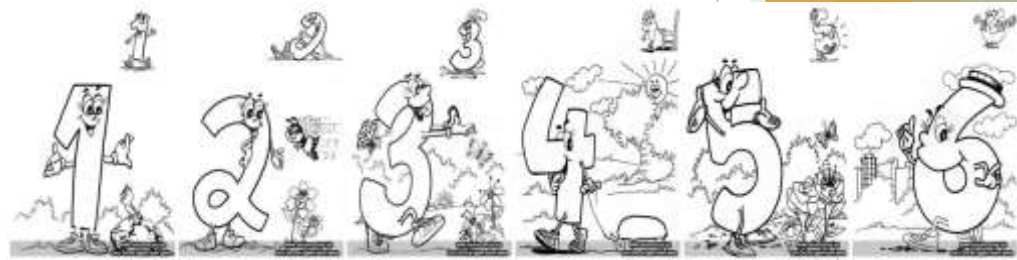
```
<style type="text/css">
  img{width:200px; height:300px; float:left; display:none;}
</style>
```

```






```



```
<script>
  $(document).ready(function(e) {
    var f5=function(){alert("5.resim...");}
    $("#img1").show();
    $("#img2").show(2000);
    $("#img3").show(2000, 'swing');
    $("#img4").show(2000, function(){alert("4.resim...")});
    $("#img5").show(2000,'linear', f5);
    $("#img6").show('explode', 2000, function(){alert("6.resim...")});
  });
</script>
```

Hide

```
<script src="jquery.min.js"></script>
<script src="jquery-ui.min.js"></script>
```

```
<style type="text/css">
  img{width:200px; height:300px; float:left; display:block;}
</style>
```

```






```

```
<script>
  $(document).ready(function(e) {
    var f5=function(){alert("5.resim...");}
    $("#img1").hide();
    $("#img2").hide(2000);
    $("#img3").hide(2000, 'swing');
    $("#img4").hide(2000, function(){alert("4.resim...");});
    $("#img5").hide(2000, 'linear', f5);
    $("#img6").hide('explode', 2000, function(){alert("6.resim...");});
  });
</script>
```

Toggle

```
<script src="jquery.min.js"></script>
<script src="jquery-ui.min.js"></script>

<style type="text/css">
  img{width:200px; height:300px; float:left; display:none;}
</style>



<script>
  var effects=new Array(
    "blind","bounce","clip","drop","explode",
    "fold","highlight", "puff","pulsate",
    "scale","shake","size","slide");

  $(document).ready(function(e){
    toggle();
    function toggle()
    {
      var ef=Math.floor(Math.random()*effects.length);
      $(document).attr("title",effects[ef]);
      $("#img1").toggle(effects[ef], 2000,toggle);
    }
  });
</script>
```

FadeIn, FadeOut, FadeToggle, FadeTo

- ▶ **FadeIn** : Belirlenen sürede nesnenin şeffaflığını artırarak gösterir.
- ▶ **FadeOut** : Belirlenen sürede nesnenin şeffaflığını azaltarak gösterir.
- ▶ **FadeTo**: Belirlenen şeffaflık değerine ulaşılmasını sağlar.
- ▶ **FadeToggle**: fadeIn/fadeOut fonksiyonlarının birleşimidir.

```
<script src="jquery.min.js"></script>
```

```
<style type="text/css">  
  img{width:200px; height:300px; display:none;}  
</style>
```

```

```

```
<script>  
$(document).ready(function(e){  
  fadeToggle();  
  function fadeToggle()  
  {  
    $("#img1").fadeToggle('slow', fadeToggle);  
    //$("#img1").fadeTo('slow', 0.3);  
  }  
});  
</script>
```

FadeTo

```
<script src="jquery.min.js"></script>
<style type="text/css">
  img{width:200px; height:300px; display:none;}
</style>

<script>
  $(document).ready(function(e){
    fadeTo ();
    function fadeTo ()
    {
      $("#img1").fadeTo(500, 0.1)
        .delay(500)
        .fadeTo(500,1, fadeTo);
    }
  });
</script>
```

SlideDown, SlideUp, SlideToggle

- ▶ *SlideDown*: Nesneyi yukarıdan aşağıya doğru açarak gösterir.
- ▶ *SlideUp*: Nesneyi aşağıdan yukarıya doğru kapatılarak gizler.
- ▶ *SlideToggle*: Nesneyi gizler veya gösterir.

```
<script src="jquery.min.js"></script>
```

```
<style type="text/css">
```

```
img{width:200px; height:300px; float:left; display:none;}
```

```
</style>
```

```

```

```
<script>
```

```
$(document).ready(function(e){
```

```
toggle();
```

```
function toggle()
```

```
{
```

```
$("#img1").slideToggle(2000, function(){ toggle()});
```

```
}
```

```
});
```

```
</script>
```

Animate

- ▶ Animate: CSS özelliklerini değiştirerek animasyon yapılır
- ▶ Stop: Animasyonu durdurur.
 - ▶ `animate({Özellik}, Süre)`
 - ▶ `animate({Özellik}, {duration:süre, easing:'ease...', step:function, })`

```
<script src="jquery.min.js"></script>
<script src="jquery-ui.min.js"></script>
```

```
<style type="text/css">
img{width:200px; height:300px; border-spacing:0;}
</style>
```

```

```

```
<script>
$(document).ready(function(e){
  var fnComplete=function(){alert("bitti")};
  $('#img1').animate({ borderSpacing: 360 }, {
    step: function(now,fx) { $(this).css('transform','rotate('+now+'deg)'); },
    duration:1000,
    easing:'easeInBack',
    complete:fnComplete
  });
});
</script>
```


Animate - easing

<i>easeIn...</i>	<i>easeOut...</i>	<i>easeInOut...</i>
<i>Quad</i>	<i>Quad</i>	<i>Quad</i>
<i>Cubic</i>	<i>Cubic</i>	<i>Cubic</i>
<i>Quart</i>	<i>Quart</i>	<i>Quart</i>
<i>Quint</i>	<i>Quint</i>	<i>Quint</i>
<i>Sine</i>	<i>Sine</i>	<i>Sine</i>
<i>Expo</i>	<i>Expo</i>	<i>Expo</i>
<i>Circ</i>	<i>Circ</i>	<i>Circ</i>
<i>Elastic</i>	<i>Elastic</i>	<i>Elastic</i>
<i>Back</i>	<i>Back</i>	<i>Back</i>
<i>Bounce</i>	<i>Bounce</i>	<i>Bounce</i>

Animate - Stop

```

<script src="jquery.min.js"></script>
<script src="jquery-ui.min.js"></script>
<style type="text/css">
    img{width:100px; height:100px; border-spacing:0;}
</style>
<button id="anim">Çalıştır</button>
<button id="stop">Durdur</button>
<br />

<script>
    $(document).ready(function(e){
        var fnStep=function(now,fx) { $(this).css('transform','rotate('+now+'deg)'); }
        var fnAnimate=function(){
            $('#img1').animate({ width:"600"},{duration:1000})
                .delay(1000)
                .animate({ height:"600"},{duration:1000})
                .delay(1000)
                .animate({ borderSpacing: 360 }, {
                    step: fnStep,
                    duration:1000,
                    easing:'easeInOutBack'
                })
        };
        var fnStop=function(){$('#img1').stop()};

        $("#anim").click(fnAnimate);
        $("#stop").click(fnStop);
    });
</script>

```

Jquery UI

- ▶ <http://jqueryui.com/demos/>
- ▶ Etkileşim
 - ▶ Draggable
 - ▶ Droppable
 - ▶ Resizable
 - ▶ Selectable
 - ▶ Sortable
- ▶ Araçlar
 - ▶ Accordion
 - ▶ Autocomplete
 - ▶ Button
 - ▶ Datepicker
 - ▶ Dialog
 - ▶ Menu
 - ▶ Progressbar
 - ▶ Slider
 - ▶ Spinner
 - ▶ Tabs
 - ▶ Tooltip
- ▶ Efektler
 - ▶ Add Class
 - ▶ Color Animation
 - ▶ Effect
 - ▶ Hide
 - ▶ Remove Class
 - ▶ Show
 - ▶ Switch Class
 - ▶ Toggle
 - ▶ Toggle Class
- ▶ Hizmetler
 - ▶ Position
 - ▶ Widget Factory

DragDrop

```

<html>
<head>
  <meta charset="utf-8" />
  <title>jQuery - DragDrop</title>
  <link rel="stylesheet" href="jquery-ui.css" />
  <script src="jquery.min.js"></script>
  <script src="jquery-ui.min.js"></script>
  <style>
    #draggable { width: 100px; height: 100px; padding: 0.5em; float: left; margin: 10px 10px 10px 0; }
    #droppable { width: 200px; height: 200px; padding: 0.5em; float: left; margin: 10px; }
  </style>
  <script>
    $(function() {
      $( "#draggable" ).draggable();
      $( "#droppable" ).droppable({
        drop: function( event, ui ) {
          $( this )
            .addClass( "ui-state-highlight" )
            .find( "p" )
              .html( "Sürüklendi ve bırakıldı!" );
        }
      });
    });
  </script>
</head>
<body>
  <div id="draggable" class="ui-widget-content"><p>Beni sürüle...</p></div>
  <div id="droppable" class="ui-widget-header"><p>Buraya bırak...</p></div>
</body>
</html>

```



Sortable

```

<html>
<head>
  <meta charset="utf-8" />
  <title>jQuery UI Sortable - Default functionality</title>
  <link rel="stylesheet" href="jquery-ui.css" />
  <script src="jquery.min.js"></script>
  <script src="jquery-ui.min.js"></script>
  <style>
    #sortable { list-style-type: none; margin: 0; padding: 0; width: 60%; }
    #sortable li { margin: 0 3px 3px 3px; padding: 0.4em; padding-left: 1.5em; font-size: 1.4em; height: 18px; }
  </style>
  <script>
    $(function() {
      $( "#sortable" ).sortable();
      $( "#sortable" ).disableSelection();
    });
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
  <li class="ui-state-default">Item 6</li>
  <li class="ui-state-default">Item 7</li>
</ul>
</body>
</html>

```

Item 1	Item 4
Item 2	Item 7
Item 3	Item 5
Item 4	Item 2
Item 5	Item 6
Item 6	Item 1
Item 7	Item 3

Ajax Metotları

Metot	Açıklama
<code>\$.ajax</code>	GET, POST vb. gibi AJAX isteği oluşturur.
<code>ajaxComplete</code>	Bütün AJAX isteklerinin tamamlanmasıyla devreye girer.
<code>ajaxError</code>	AJAX isteği bir hata ile karşılaşıldığında devreye girer.
<code>ajaxSend</code>	AJAX isteği gönderilmeden önceki durumu tanımlar.
<code>\$.ajaxSetup</code>	Genel olarak AJAX istekleri için varsayılan ayarları tanımlar.
<code>ajaxStart</code>	İlk AJAX isteği ve işlevinin başlamasıyla devreye girer.
<code>ajaxStop</code>	Bütün AJAX istekleri bittiğinde devreye girer.
<code>ajaxSuccess</code>	AJAX isteği başarıyla tamamlandığında devreye girer.
<code>\$.get</code>	Sunucudan GET isteği ile veri alır ve verir.
<code>\$.getJSON</code>	GET isteği ile sunucudan kodlanmış JSON verileri yükler.
<code>\$.getScript</code>	GET isteği ile sunucudan javascript yükler ve çalıştırır.
<code>load</code>	Seçilen bir nesneyi (dosya vb.) sunucudan yükler, HTML veri halinde döndürür.
<code>\$.param</code>	AJAX istekleri için URL sorgu dizisi olarak kullanılır.
<code>\$.post</code>	Sunucudan POST isteği ile veri yükler.
<code>serialize</code>	Form nesnelerini anahtar-değer ikilisi olarak kodlar.
<code>serializeArray</code>	Form nesnelerini anahtar-değer ikilisi biçiminde bir JSON nesnesi olarak kodlar.

Ajax Metotları

```
$.ajax({type:"GET/POST",  
        url:"veri sayfası",  
        data:"giden veri",  
        success:function(gelenveri){ "işlemler"}  
    })
```

```
$.get("veri sayfası", "giden veri", "gelen veri");  
$.post("veri sayfası", "giden veri", "gelen veri");  
$.getJSON("veri sayfası", "giden veri", "gelen veri");
```

\$.ajax

- Sunucuya veri gönderme ve veri alma işlemlerinde kullanılır.

- PHP sayfası (ajax_sonuc.php)

```
<?php
    $d1=$_GET["deger1"]; //$d1=$_POST["deger1"];
    $d2=$_GET["deger2"]; //$d2=$_POST["deger2"];
    echo $d1+$d2;
?>
```

- HTML sayfası (ajax.html)

```
<script src="jquery.min.js"></script>
<script type="text/javascript">
    $(document).ready(function(e) {
        $("#button").click(function(){
            $.ajax({
                type:"GET", //type:"POST"
                url:"ajax_sonuc.php",
                data:$("#frm").serialize(),
                success:function(sonuc){
                    $("#sonuc").html(sonuc);
                }
            });
        });
    });
</script>
<form id="frm">
    <input type="text" id="deger1" name="deger1" />
    <input type="text" id="deger2" name="deger2" />
</form>
<button>Hesapla</button>
<div id="sonuc"></div>
```


\$.get

- ▶ GET metodu ile sunucuya veri gönderme ve alma işlemlerinde kullanılır.
- ▶ PHP sayfası (get_sonuc.php)

```
<?php
    $d1=$_GET["deger1"];
    $d2=$_GET["deger2"];
    echo $d1+$d2;
?>
```

- ▶ HTML sayfası (get.html)

```
<script src="jquery.min.js"></script>
<script type="text/javascript">
    $(document).ready(function(e) {
        $("button").click(function(){
            $.get("get_sonuc.php",
                $("#frm").serialize(),
                function(sonuc){$("#sonuc").html(sonuc)}
            );
        });
    });
</script>
<form id="frm">
    <input type="text" id="deger1" name="deger1" />
    <input type="text" id="deger2" name="deger2" />
</form>
<button>Hesapla</button>
<div id="sonuc"></div>
```

\$.post

- ▶ POST metodu ile sunucuya veri gönderme ve alma işlemlerinde kullanılır.
- ▶ PHP sayfası (post_sonuc.php)

```
<?php
    $d1=$_POST["deger1"];
    $d2=$_POST["deger2"];
    echo $d1+$d2;
?>
```

- ▶ HTML sayfası (post.php)

```
<script src="jquery.min.js"></script>
<script type="text/javascript">
    $(document).ready(function(e) {
        $("#button").click(function(){
            $.post("post_sonuc.php",
                $("#frm").serialize(),
                function(sonuc){ $("#sonuc").html(sonuc) }
            );
        });
    });
</script>
<form id="frm">
    <input type="text" id="deger1" name="deger1" />
    <input type="text" id="deger2" name="deger2" />
</form>
<button>Hesapla</button>
<div id="sonuc"></div>
```

\$.getJSON

- ▶ JSON (Javascript Object Notation) biçiminde veri almak için kullanılır.

- ▶ PHP sayfası (getJSON_sonuc.php)

```
$id=$_GET["id"]; $json=null;
if($id==1) $json=array("id"=>"1", "ad"=>"Ali", "soyad"=>"Koş", "tel"=>"1234");
if($id==2) $json=array("id"=>"2", "ad"=>"Veli", "soyad"=>"Coş", "tel"=>"5678");
echo json_encode($json);
```

- ▶ HTML sayfası (getJSON.html)

```
<script type="text/javascript" src="jquery.min.js"></script>
<script type="text/javascript">
    $(document).ready(function(){
        var gelen=function(veri, status, xhr){
            $("#sonuc").empty();
            $.each(veri, function(anahtar, deger){
                $("#sonuc").append(anahtar+": "+deger+"<br>");
            });
        }
        $("button").click(function(){
            var giden=$("#frm").serialize();
            $.getJSON("getJSON_sonuc.php", giden, gelen);
        });
    });
</script>
<form id="frm"><input id="id" name="id"></form>
<button>Göster</button>
<div id="sonuc"></div>
```

load

- ▶ HTML, PHP vb. sayfalardan veya text dosyasından veri almak için kullanılır.

- ▶ PHP sayfası (load.php)

```
<?php  
    echo "Bu içerik load metodu ile yazdırıldı."  
?>
```

- ▶ HTML sayfası (load.html)

```
<script src="jquery.min.js"></script>  
<script type="text/javascript">  
    $(document).ready(function(e) {  
        $("button").click(function(){  
            $("#txt").load("load.php");  
        });  
    });  
</script>  
<div id="txt">Değişecek metin</div>  
<button>İçeriği Al</button>
```

ajaxStart

- ▶ AJAX isteği başladığında yapılacak işlemler tanımlanır.
 - ▶ `$(seçici).ajaxStart(function(){...});`

```
<script src="jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(e) {
    $(document).ajaxStart(function() {
        $("#loading").html('');
    });

    $("button").click(function(){
        $("#txt").load("test1.php");
    });
});
</script>
```

```
<div id="txt">Değişecek metin</div>
<button>İçeriği Al</button>
<div id="loading"></div>
```

ajaxComplete

- ▶ AJAX isteği tamamen bittiğinde yapılacak işlemler tanımlanır.
 - ▶ `$(seçici).ajaxComplete(function(event, XMLHttpRequest, ajaxOptions){...});`

```
<script src="jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(e) {
    $(document).ajaxComplete(function(ev, xhr, op){
        $("#loading").css("display","none");
    });
    $("#button").click(function(){
        $("#loading").html('');
        $("#txt").load("test1.php");
    });
});
</script>
<div id="txt">Değişecek metin</div>
<button>İçeriği Al</button>
<div id="loading"></div>
```

ajaxSuccess

- ▶ AJAX isteği başarıyla tamamlandığında yapılacak işlemler tanımlanır.
 - ▶ `$(seçici).ajaxSuccess(function(event, XMLHttpRequest, ajaxOptions){...});`

```
<script src="jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(e) {
    $(document).ajaxSuccess(function(ev, xhr, op) {
        $("#loading").html("AJAX isteği başarılı...");
    });

    $("#button").click(function(){
        $("#loading").html('');
        $("#txt").load("test1.php");
    });
});
</script>
<div id="txt">Değişecek metin</div>
<button>İçeriği Al</button>
<div id="loading"></div>
```

ajaxStop

- ▶ Bütün AJAX istekleri tamamlandığında yapılacak işlemler tanımlanır.

- ▶ `$(seçici).ajaxStop (function(){...});`

```
<script src="jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(e) {
    $(document).ajaxStop(function() {
        $("#loading").html("AJAX istekleri tamamlandı...");
    });

    $("#button").click(function(){
        $("#loading").html('');
        $("#txt1").load("test1.php");
        $("#txt2").load("test1.php");
    });
});
</script>
```

```
<div id="txt1">Değişecek metin1</div>
<div id="txt2">Değişecek metin2</div>
<button>İçeriği Al</button>
<div id="loading"></div>
```


ajaxError

- ▶ AJAX istekleri sırasında hata oluştuğunda yapılacak işlemler tanımlanır.
 - ▶ `$(seçici).ajaxError (function(event, XMLHttpRequest, ajaxOptions, thrownError){...});`

```
<script src="jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(e) {
    $(document).ajaxError(function(ev, xhr, op, th) {
        $("#loading").html("AJAX isteklerinde hata oluştu...");
    });

    $("#button").click(function(){
        $("#loading").html('');
        $("#txt").load("test2.php");
    });
});
</script>
```

```
<div id="txt">Değişecek metin</div>
<button>İçeriği Al</button>
<div id="loading"></div>
```