T.C.
CELAL BAYAR ÜNİVERSİTESİ
TURGUTLU MESLEK YÜKSEKOKULU

BİLGİSAYAR TEKNOLOJİSİ VE PROGRAMLAMA

JavaScript DERS NOTU

Seval ÖZBALCI
seval.ozbalci@bayar.edu.tr

MANİSA 2003

İÇİNDEKİLER

JAVASCRIPT

Java ile JavaScript oldukça fazla karıştırılmaktadır. Java, Sun firması tarafından Pascal ve Delphi dillerinden esinlenerek yazılmış bir programlama dilidir. Sonuçta işletim sistemlerinden bağımsız bir program elde edersiniz. Yani exe veya com uzantılı dosya oluşturur. Fakat JavaScript bu tür bir programlama dili değildir. Yorumlanması için bir tarayıcıya ihtiyaç duyar. Bu yüzden script dilidir. Html dosyasını içine gömülüdür. Sonuçta elinizde exe veya com uzantılı bir dosya yoktur. Javascript, Netscape firması tarafından C dilinden esinlenilerek yazılmıştır. Yazılma amacı Html'in sahip olmadığı bazı özelliklerin web sayfalarında kullanılmak istenmesidir. Ziyaretçi ile etkileşim gibi önemli unsurlarda Html'in eksiklerini tamamlayıcı unsurlara ihtiyaç duyulması sonucunda Netscape firması bu konuya ağırlık vererek JavaScript script dilini internet ortamına kazandırmıştır.

Netscape ve Internet Explorer tarayıcılarının JavaScript kodundaki anlayışları farklıdır. Nedeni ise, Netscape JavaScript dilini hazırladığında Microsoft'un bu dilin özelliklerini veya yazılım tarzını tam anlamıyla Internet Explorer'a eklemeyip kendi yazım kurallarını belirlemesidir. Bu yüzden JavaScript kodu yazarken bu iki tarayıcı özelliklerini de göz önünde bulundurmalıyız.

Java Script'in Genel Bazı Özellikleri

- Javascript kodlarını yazmak için Windows kullanıcıları için NotePad, Mac. kullanıcıları için Simple Text yeterlidir.
- JavaScript kodları <script> etiketi ile başlar, </script> etiketi ile biter.
- <script> etiketi, JavaScript'i anlamayan eski sürüm tarayıcıların bu kısmı geçmeleri içindir.

Genellikle yazım tarzı aşağıdaki şekildedir:

```
<script>
<!—
JavaScript kodları
-->
</script>
```

İyi bir programcı kod satırlarında açıklama yapar. Bu satır şu işlemi gerçekleştiriyor gibi açıklayıcı bilgiler yazar kodlarının yanına. JavaScript'te bu tür

açıklamalar // ile başlar ve // ile biter. Eğer açıklamanız bir satırdan fazla ise /* ile başlar */ ile biter.

Örnek:

```
// bu satır kullanılacak değişkenlerin tanımlanması
/* açıklama satırı 1
açıklama satırı 2
açıklama satırı 3 */
```

JavaScript kodları Html kodların arasında yer alır. Veya uzantısı js olan dosyalarda saklanarak yine Html içerisinden çağırılır. Java Appletleri gibi Html'den ayrı bir unsur değildir. Javascript Html'in bir parçasıdır.

Kullanılacak yere göre Html'in içerisinde kullanılır. Fakat genelde <head> </head> etiketleri arasında kullanılır.

Javascript kodları bittiğinde elinizde asla kendi başına çalışan uzantısı exe veya com olan bir dosya olmaz. Her zaman için tarayıcı tarafından yorumlanması gerekir. Yorumlanması demek Javascript kodunun çalışması anlamındadır.

Nesne ve buna uygulanan olaylar ile ilgili bir takım görevleri vardır. Javascript kullandığı her unsuru nesne olarak algılar. Siz bu nesneleri tıklamak, üzerine gelmek, üzerinde çıkmak gibi olaylar ile çalıştırırsınız ki bu da Javascript'in ziyaretçi ile etkileşmesi demektir.

NOT: Her konuyu şöyle bir okuyup hemen ardından uygulamalısınız. Uygulamada dikkat edilmesi gereken; her kod örneğinin (değişkenler konusundakiler gibi) kendi başına bir iş yapmayabileceğidir. Bu tür kodlar diğerlerinden farklı olarak arka planı açık sarı ve italik olarak yazılmıştır. Uygulayabilmek için diğer birtakım kodlara da ihtiyaç duyulur. Bu yüzden bu tip konuları sadece bilginiz olması amacıyla okuyup geçiniz.

DEĞİŞKENLER

Genel Değişken Özellikleri

Değişkenler Javascript'te ve diğer programlama dillerinde olduğu gibi bilgi depolamak bu bilgiyi kullanmak amacıyla kullanılırlar. Değişkenler "var" komutu ile oluşturulurlar. Karakter olarak kullanıldıklarında işlem yapılamazlar. Nümerik olarak kullanıldıklarında ancak işlem yapabilirler. Kullanımına bir örnek verelim.

Örnek:

```
var sayi;
var sayi1=10;
var yazi1="10";
```

Burada birinci satırdaki "sayi" değişkeni script kodunun herhangi bir yerinde kullanılmak üzere oluşturulmuştur. İkinci satırda "sayi1" adındaki değişkenin değeri hemen o satırda = ifadesinden sonra verilmiştir. Böyle değişken tanımıda yapılabilir. Üçüncü satırda ise değişkenin karakter ifadesi olarak kullanımını görüyoruz. Burada önemli olan karakter değişkenlerin alıntı " " ifadesinin arasında kullanılmasıdır. Her değişkenden sonra ; işareti konulmalıdır. Tarayıcı, bir başka komut satırına geçtiğini bu yol ile anlar.

Şimdi değişkenlerle ilgili matematik işlemlerinin nasıl olacağını görelim. Bunu daha iyi açıklayabilmek için örnekler üzerinde çalışalım.

Örnek:

```
var sayi1=10;
var sayi2=20;
var sayi3=sayi1+sayi2;
```

Birinci ve ikinci satırlarda değişkenler oluşturduk. Üçüncü satırdaki ise sayi3 değişkeni ile diğer iki değişkeni topladık. Burada önemli olan işlem yapmak istediğimizde değişken değerinin alıntı " " işaretlerinin arasına konmamasıdır. Üçüncü satırı - ileride göreceğimiz write () fonksiyonu ile - tarayıcıda yazdırırsak göreceğimiz değer 30'dur.

Şimdi de değişkenleri karakter olarak tanımladığımızda neler olduğuna bakalım.

Örnek:

```
var sayi1="10";
var sayi2="20";
var sayi3= sayi1+sayi2 ;
```

Bir önceki örnekten farklı olarak değişken değerlerinin alıntı işaretleri içerisinde yazdık. Eğer sayi3 adlı değişkeni tarayıcıda bastırırsak göreceğimiz ifade 1020 ifadesidir. Yani tarayıcı karakter olarak tanımladığımız değişkenleri ardada ekledi. Burada unutulmaması gereken şey bunun sadece + işleminde geçerli olmasıdır. Diğer işlem türlerinde bu tür bir sonuç alınamaz.

Değişkenlere ad verirken uymamız gereken kurallar.

1) Değişken isimleri harf veya _ karakteri ile başlayabilir. Rakam kullanmak istersek 2. karakterden sonra kullanabiliriz. Yani değişkenin ilk karakteri rakam olamaz. Değişken isimlerine örnekler;

```
var url="webteknikleri"; doğru
var _rakam=12; doğru
var a1=123; doğru
var 3uzler="üçüzler" yanlış
```

2) Değişken tanımlarken bir veya birden fazla boşluk bırakmak tanımlama açısından herhangi bir sorun teşkil etmez.

3) Değişken adı verirken kullandığımız harflerin büyük veya küçük olması bazı tarayıcılarda fark etmezken çoğu tarayıcıda farklı bir değişken anlamındadır. Yani;

```
var say=1;
var Say=1;
```

Birçok tarayıcıda farklı değişkenler olarak algılanır.

Değişkenlerin İşlem Operatörleri İle Kullanımı

Değişkenlere işlem yaptırabilecek operatörleri ve özelliklerini inceleyelim. Operatörleri birkaç kısıma ayırarak inceleyelim;

- Aritmetik operatörler
- Karşılaştırmak operatörleri
- Mantıksal operatörler
- Özel operatörler

ARİTMETİK OPERATÖRLER

Her zaman kullandığımız bu operatörler + , - , * , / , % 'dir.Değişkenlerin çeşitli aritmetik operatörlerle kullanımına bir örnek verelim.

Örnek:

```
var i=10;
var j=11;
var k=12;
var m,n;
m=i*j+k;
n=i*(j+k);
```

İlk üç satırda değişkenlerimizi hem tanımlayıp hem de değer atadık. Dördüncü satırda ise m ve n değişkenlerini tanımlayıp değer atamadık. Diğer satırlarda ise m ve n değişkenlerinden istenen işlemleri tanımladık. Buna göre son iki satırın sonuçları farklıdır. Çünkü parantezlerin işlem önceliği vardır.

Beşinci satırın cevabı (10*11)+12 = 122 şeklinde olacaktır. Son satırda ise sonuç 10*(11+12) = 230 olacaktır. Diğer bir işlem operatörü olan % 'nin yaptığı işlemi şu şekilde anlatabiliriz. % operatörü % işaretinin solundaki değişkeni sağındaki değişkene böler ve kalanı verir. Örnek olarak;

```
var a=100; var b=9; var c=100%9;
```

Burada c değişkenin değeri 100/9'un kalanı 1'dir. Yani c değişkeninin değeri 1 olacaktır. Diğer -(eksi) ve / (bölme) operatörlerinin işlemleri kendilerine atanan çıkartma ve bölme işlemidir. Bu operatörlerin kısa kullanımı içinde Javascript bize kolaylık sağlar. Bu operatörleri sıralamak istersek;

-= : *= : /= : %= : ++ : --

Bu operatörlerin kısa uzun şekilde yazılışları ise aşağıdaki gibidir.

```
x+=y x=x+y
x-=y x=x-y
x*=y x=x*y
x/=y x=x/y
x%=y x=x%y
x++ x=x+1
x-- x=x-1
```

Bu operatörlerin nasıl işlem yaptığını bir örnekte görelim.

```
var x,y,z;
```

```
x=10; y=20; z=30;
x++; x+=y; z--; y*= z;
```

Şimdi her zamanki gibi işlem satırlarının cevaplarını birlikte bulalım. x++ satırı x=x+1 işleminin yapılmasını söyler. Buna göre x değişkeni 11 değerini alır. İkinci satıra geldiğimizde x+=y satırı x=x+y işleminin yapılmasını söyler. Bir önceki satırdaki x'in değeri 11 idi. Böylelikle yeni x'in değeri 11+20=31 olacaktır. Diğer satırda z-- işlemi sonucunda z'nin değeri 29 olacaktır. Son satırda ise y=y*z işlemi ile y değişkeni 20*29= 380 değerini alacaktır.

KARŞILAŞTIRMA OPERATÖRLERİ

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

== operatörü iki değişkenin birbirine eşitliğini kontrol eder.

!= operatörü iki değişkenin birbirine eşit olmadığı durumlarda kullanılır.

< operatörü bilindiği üzere küçüktür operatörüdür. Soldaki değişkenin sağdakinde küçüklüğünü kontrol eder.

<= soldaki değişkenin sağdaki değişkene küçük eşitliğini kontrol eder.

> soldaki değişkenin sağdaki değişkene göre büyük olup olmadığını kontrol eder.

>= soldaki değişkenin sağdaki değişkene büyük eşitliğini kontrol eder.

MANTIKSAL OPERATÖRLER

İki değişkene bağlı karşılaştırılmaların yapılmak istendiği durumlarda kullanılır. Operatörler && , || , ! operatörleridir. && And (ve) operatörü iki değişkenin de değeri doğru olması istendiğinde kullanılır. || Or (veya) operatörü iki değişkenden en az birinin doğru olması durumu istediğinde kullanılır. ! Not (değil) operatörü değişkenin değeri doğru ise yanlış , yanlış ise doğru olması istendiği durumlarda kullanılır.

ÖZEL KARŞILAŞTIRMA OPERATÖRÜ

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

değişken_1 [karşılaştırma operatörü] değişken_2 ? değişken_3 : değişken_4

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

a < b ? c : d

Yukarıdaki satırda yapılması istenen işlem; a değişkeninin b değişkeninden küçük olup olmadığı karşılaştırılıyor. Buna göre cevap doğruysa işlemin sonucu c değişkeninin değeri değilse d değişkeni oluyor. Şimdi tüm bu öğrendiklerimizi bir Javascript kodunda görelim. Bu bizim ilk Javascript kodumuz olacak.

Örnek :

```
<script Language="JavaScript 1.2">
<!—
var i=1; var j=2;
var k=3; var m=4;
var n=5;
var p=6;
var q=7;
i=+j;
j++;
k--;
m=m+k;
n=*j;
i < j ? 3 : 1 ;
k >= n ? 0 : 1 ;
k=2 && j=5 ? p : q ;
i=2 || j=3 ? m : n ;
p!=2 ? k : 10 ;
-- >
</script>
```

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

```
i+=j;
```

Bu işlem daha da önce gördüğümüz gibi bize i=i+j işlemini yapmamızı söyler. Buna göre i değişkeninin değeri 3 olacaktır. Hemen altındaki satırda bulunan j++ işlemi dolayısıyla da j değişkeni 3 değerini alacaktır. Diğer işlem satırında k--işlemi ile de k değişkeni 2 değerini alır. Bir diğer satırdaki m=m+k işlemi ile m(m=4) değişkeni k(k=2) değişkeni toplanarak 6 değerini alır. n=*j işlemi ile de n(n=5) değişkeni 3*5=15 değerini alacaktır.

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

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

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

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

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

p!=2 ? k : 10 ;

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

Internet Explorer & Netscape Farkı

Giriş kısmında belirttiğimiz gibi Javascript kodlarında MSIE (Microsoft Internet Explorer) ve NN (Netscape Navigator) yönünden farklılık vardır. Bu tarayıcının html dökümanı nasıl modellediğine bağlıdır. Tarayıcının nesne döküman modeli, bir Html sayfasındaki çeşitli elemanların tarayıcı tarafından nasıl algılanıp yorumlandığı ile ilgilidir. Javascript gerçekte W3C (Web tekniklerinin standartlarını belirleyen kurum www.w3c.org) konsorsiyumu tarafından belirlenen kodlardan oluşmamıştır. Tarayıcı üreten firmalar bu konuları kendilerince yorumlayıp tarayıcılarına yerleştirmişlerdir. Yani kendi nesne döküman modellerini oluşturmuşlardır.

Biz bu kısımda her iki tarayıcınında nesne döküman modelini incelemeyeceğiz. Bize gerekli olan kısmını öğreneceğiz. Şimdi tarayıcı farkının nasıl ayırt edilebileceğini görelim.

```
ie4 = (document.all) ? true : false ;
nn4 = (document.style) ? true : false ;
```

Biz bu iki satırla bir önceki ders olan değişkenler ve mantıksal operatörler yardımıyla iki tarayıcıyı da kontrol altına aldık. Bir diğer örnekle yapıyı pekiştirelim.

```
<script language="Javascript">
<!-- // Kodları eski sürüm tarayıcılardan saklayalım.
ie4 = (document.all) ? true : false ;
nn4 = (document.style) ? true : false ;
if (ie4)
{
// MSIE 4.0 için uygun kodları buraya yazacağız
}
else
{
// NN 4.0 için uygun kodları buraya yazacağız.
}
//Saklamayı bitir -->
</script>
```

Şimdi bu kodları inceleyelim. Değişken tanımlama kısmında anlamadığınız bir kısım varsa 1. Değişkenler kısmına tekrar geri dönmelisiniz.

If (ie4) ve if (nn4) Burada ileriki derslerde öğreneceğimiz koşul ifadesini kullanıyoruz. Bu kodları Javascript'in anlayış tarzı şu şekilde olacaktır. Eğer nn4 , ie4 değişkenlerinden doğru olanı ie4 ise -ki bunu true ve false değerlerinden algılar- alt satıra geçip ona uygun kodu uygulayacaktır. Şayet ie4=false yani nn4=true ise diğer if koşulu yorumlanarak işleme konulacaktır. Bu da nn4 için gerekli kodun çalıştırılması demektir.

Örnek Uygulama 1:

```
<html>
<head>
<title>Tarayıcı kontrolü</title>
<head>
<script language="Javascript">
<!-- // Kodları eski sürüm tarayıcılardan saklayalım.
function tarayici() {
ie4 = (document.all) ? true : false ;
nn4 = (document.style) ? true : false ;
if (ie4)
{
// MSIE 4.0 için uygun kodları buraya yazacağız.
```

```
window.location="ie.htm";
}
else
{
// NN 4.0 için uygun kodları buraya yazacağız.
window.location="nn.htm";
}
}
//Saklamayı bitir -->
</script>
</head>
<body onLoad=tarayici()>
</body>
</html>
```

1. Yukarıda verilen kodları herhangi bir editör (NotePad gibi) yardımıyla yazıp tara.htm olarak kaydedin.

```
<html>
<head>
<title>MSIE tarayıcı kullanıyorsunuz</title>
</head>
<body>
<h3>Tarayıcınız Internet Explorer</h3>
</body>
</html>
```

2. Bu kodu ie.htm olarak kaydedin.

```
<html>
<head>
<title>Netscpae tarayıcı kullanıyorsunuz</title>
</head>
<body>
<h3>Tarayıcınız Netscape Navigator</h3>
</body>
</html>
```

3. Bu kodu nn.htm olarak kaydedin.

Önemli! Bu üç Html dosyasının da aynı klasörde olması gereklidir.

Tara.htm adlı dosyada anlamadığınız kodlar olduğunu görüyorsunuz. Şimdilik bu kodları anlamanız gerekli değil. Yeri geldikçe bu kodların nerede ve nasıl kullanılacağını göreceğiz.

Ekrana Çıktı Ve Klavyeden Bilgi Giriş

Html üzerinden klavye aracılığı ile ziyaretçiden bilgi almayı ve herhangi bir değişken vb. türde yazıların html e nasıl yazdırılacağını göreceğiz.

PROMPT ()

Hemen başlayalım. Ziyaretçiden bilgi alma iki tür JavaScript komutuyla gerçekleşir. Birisi Prompt yani bizim burada bahsedeceğimiz komut. Diğeri ise Form yoluyla bilgi alınması. Form yoluyla alınan bilgiler formun Html üzerinde yer alması yüzünden Prompt komutu ile birbirinden ayrılır. Prompt komutu ile Html sayfasından hariç bir pencere açılır. Alınmak istenen bilgi ziyaretçiye bu yol ile sorulur ve hemen altındaki boşluk yardımıyla cevap alınır. Şimdi kodun nasıl kullanıldığına bir göz atalım.

```
prompt ("Sorulan soru" , "Cevap örneği")
```

Bu komutun yorumlanışı şu şekildedir. Html üzerinde Html'den bağımsız bir pencere aç (bu prompt komutu ile yapılır). İlk çift tırnak içerisinde olan kelime veya kelime grubu, pencerenin üst kısmında ve değiştirilemeyen kısımdır. Burada soru veya pencerenin niçin açıldığı ile ilgili bir açıklama verilir. İkinci çift tırnakta ise doldurulacak olan cevap satırının içinde seçili bir haldedir. Bu ise genel olarak cevap örneği olarak ziyaretçiye sunulur. Kullanılması zorunlu değildir. Kullanılmadığınızda undefined gibi tanımlanmamış uyarısı alınır.

```
prompt ("Tarayıcınızın türünü giriniz ?" ,"lütfen cevabı ie veya nn olarak
veriniz");
```

Şimdi kullanıcıdan nasıl bilgi alınacağını gördük fakat bu bilgiyi nasıl kullanabiliriz ? Bu sorunun cevabı prompt komutunu var ile bir değişkene atmak suretiyle kullanabiliriz olacaktır. Yani ;

```
var tara=prompt ("Tarayıcınızın türünü giriniz ?" ,"lütfen cevabı ie veya nn
olarak veriniz");
```

Biz bu satır ile ne yapmış olduk? Ziyaretçiden prompt komutu ile tarayıcısı sorduk ve bunu var değişken tanımlama komutuyla tara değişkenine atadık. Ziyaretçiden aldığımız bu bilgi sonucunda tara değişkeni ya ie yada nn değerini alacaktır. Biz daha sonraki satırlarda bu değişkeni belli bir koşul koyarak kullanabiliriz. Mesela daha önceki örneklerimizde olduğu gibi ie ise şu sayfayı aç nn ise şu sayfayı aç.

Bir öneri: Bu tip tarayıcı tanıma yöntemi oldukça yanlış bir yöntemdir. Çünkü ziyaretçiden alınan bilgi ile bizim daha sonra kullandığımız koşul ifadeleri uyuşmayabilir. Bu yüzden kodumuz doğru çalışmaz.

WRİTE()

Html dosyasına yazı yazdırma komutu ise **write** dır. Bu kodun yazım kurallarını inceleyecek olursak ;

```
document.write ("görüntülenmek istenenler" , değişken_ismi );
```

Kodu inceleyelim: Javascript html üzerinde yazı yazmak istediğinde write komutunu tek başına kullanamaz. Bunun için document fonksiyoneli (yardımcısı manasında) ile birlikte kullanılır. document.write komutundan sonra parantez açılır. Daha sonra yazılmak istenen sıraya göre değişken ismi veya görüntülenmek istenenler yazılır. Değişkenler çift tırnak içerisinde yazılmazlar. Sadece görüntülenmek istenenler çift tırnak içerisinde yazılır.

Şimdi prompt komutu ile write komutunu birleştirerek bir kod hazırlayalım. Bu kodumuzda prompt aracılığıyla ziyaretçiye adını sorup ad değişkenine atıyoruz. Daha sonra bu değişkeni write komutu yardımıyla ziyaretçiye Merhaba diyoruz. Şimdi kodlara geçelim.

Örnek :

```
<html><head><title>Prompt , write örneği </title></head>
<body>
<!-- //Kodları eski tarayıcılardan gizliyoruz
var isim = prompt ("İsminizi Giriniz " , "Küçük harf veya büyük harf
kullanabilirsiniz" );
document.write ("Merhaba " , isim , " !" );
// Saklamayı bitir -->
</script>
</body>
</html>
```

Eski kodlarımıza göre bu kodun biraz farklı olduğu rahatlıkla görülebilir. Javascript kodumuz <head> etiketleri arasında kullanılmaz. Daha öncede değindiğimiz şekilde uygulanması istenen sıraya bağlı olarak kodlar yerini almıştır.

Biz aslında Html'de font kurallarını kullanarak yazı da yazdırabiliriz. Hiçbir kural uygulamadığınızda tarayıcının standart (default) değerleri kullanılır. Örneğin, Merhaba dedikten sonra alınan ismin bir alt satırda görüntülenmesini istiyorsak bunu Javascript'e şu şekilde yaptırabiliriz (
 türünde Html etiketlerinin tümünü Javascript'e yaptırabilirsiniz).

```
document.write ("Merhaba" , "<br>" , isim)
```

KOŞUL VE DÖNGÜLER

Bilgisayarı bilgisayar yapan konular bunlardır. Çünkü hiçbir bilgisayar kendi kriterleriyle yorum yapamaz. Bizim koşullar ve döngüler ile verdiğimiz, önceden belirlenmiş kıstasları göz önünde bulundurarak gerekli seçimleri yapar.

Koşul Yapıları

Javascript'in en önemli özelliklerinden biri koşul yapılarıdır. Aslında bu konu sadece Javascript'in değil bilgisayarın da en önemli konusudur. Şimdi konunun inceliklerine bir göz atalım.

IF (EĞER)

Javascript'te çoğu dilde olduğu gibi koşul yapısının kodu If (eğer) komutudur. Yazılım şekli ise şu şekildedir.

```
If (a==b)
//koşul doğru ise ilk satır işleme konulur
// koşul doğru değilse ilk satırın altındaki komut satırı işleme konulur.
```

Şimdi kodumuzu biraz inceleyelim :

Koşul komutu yani if ile işleme başlıyoruz. Daha sonra karşılaştırılacak değişkenler veya başka nesneler parantez içerisinde sorgulanıyor. Dikkat ederseniz çift eşittir kullandık. Çünkü tek eşittir işareti değer atama işlemidir. Çift değişken ile koşul yapısı sağlanır. Eğer koşul doğruysa hemen altındaki satır işleme konulur. Eğer koşul yanlış ise ikinci satır işleme konulur. Yok ben koşul doğru ise 2 ve daha çok işlem yaptırmak istiyorsanız bunun cevabı yapılması istenen işlemlerin { } arasında yer almasıdır. Yani :

```
If (a==b)
{
// 1.işlem
//2. İşlem
...
...
}
```

IF .. ELSE (EĞER ... DEĞİLSE)

Bu bölümde ise If koşul ifademize Else komutunu ekleyerek koşul yapısı örneklerle açıklanacaktır.

```
If ( a==b )
{
// şunları şunları yap
}
else
{
//değilse şunları yap
}
```

Yani örnekten de anlaşıldığı gibi if koşulu ile a ile b nin eşitliği karşılaştırılıyor. Eğer doğruysa hemen altındaki kısım işleme konuluyor. Else ile yok değilse altındaki kısmı işleme koy diyoruz.

```
If (a==b)
{
//şunları yap
}
if (a==c)
{
//şunları yap
}
else
{
//şunları yap
}
```

Şimdi bu kod Javascript'e: a değişkeni b değişkenine eşitse normal olarak alt satırı işleme koymasını, eğer bu karşılaştırma yanlış ise altındaki işlemleri geçerek a'nın c'ye eşitliğini kontrol etmesini ve bu da değilse (else) alt satırdaki işlemleri devreye koymasını bildirir.

Else yapısı genel olarak bir karşılaştırma sonucunda cevap yanlış ise diğer bütün durumlarda şu işi yap manasında kullanılır.

Örnek Uygulama 2:

```
<html>
<head>
<title>Koşul yapıları </title>
</head>
<body>
<script language="JavaScript">
<!-- //eski sürüm tarayıcılardan kodumuzu saklayalım
var gun = prompt ("Bugün günlerden ne ?" ,"lütfen küçük harf kullanınız");
if (gun=="pazar")
```

```
{
document.write ("Bugün günlerden " , gun , " olduğuna göre hatfa sonundayız"
,"<br>")
document.write ("<b>" , "İyi tatiller.." , "</b>")
}
else
{
document.write ("Bugün günlerden pazar olmadığına göre tatil gününde değiliz
!" ,"<br>")
document.write ("İyi çalışmalar..")
}
//saklamayı bitir-->
</script>
</body>
</html>
```

Döngü Yapısı

Javascript'te diğer programlama dillerinde olduğu gibi istediğiniz işlemi 2 veya daha fazla kez yaptırmak için belli program kodları mevcuttur. Bu diğer dillere çok benzer olan for döngü komutudur. Bu komutun yaptığı işlem, istenilen fonksiyon veya fonksiyon parçalarını istenilen değerde tekrar etmektir.

FOR DÖNGÜSÜ

```
for ( değişken_başlangıç_değerler1 , değişken_başlangıç_değeri2 ; döngü
sayısı ; değişecek_değişken_adı_ve_türü )
{ yapılması istenen işlemler }
```

Burada parantezler içerisinde verilen değişken_başlangıç_değerler kısmı ve değişecek_değişken_adı_ve_türü kısmını yazmanız gerekmez. Döngü içerisinde kullanılan değişken daha sonrada istenilen şekilde arttırılabilir veya azaltılabilir. Yapı gözünüzü korkutmasın hemen bir örnekle daha iyi anlayalım.

```
for (a=0 , b=0 ; c<=3 ; c++)
{ yapılması istenen işlemler }
for ifadesi için kısa yazılım :
var a,b=0;
for (;c<=3;c++)
{ yapılması istenen işlemler }
```

Şimdi bunu tam bir örnekle daha da pekiştirelim. Varsalım ki elimizde bir çarpım tablosu yapmak istiyoruz. Buna göre 5 sayısı için 1'den 10'a kadar sayıları bir

tablo içerisinde vereceğiz. Şimdi bu durumda for döngüsüz 10 adet tablo yazmamız gerekecekti fakat biz for döngüsü ile işlemi 1 satıra indirgeyeceğiz.

```
<html>
<head>
<title>for döngüsü</title>
</head>
<body>
<script language="JavaScript">
<!-- //eski sürüm tarayıcılardan kodumuzu saklayalım
var cevap=0;
for ( sayi=0 ; sayi>=10 ;);
{
sayi--;
var cevap = 5 * sayi ;
document.write( "5 * " , sayi , " =" , cevap ,"<br>")
}
//saklamayı bitir-->
</script>
</body>
</html>
```

Burada gördüğünüz gibi işlem tek bir satıra ingirdendi. Şimdi de for döngüsünün yapmak istediğimiz işlemlerde yetersiz kaldığı durumlarda kullanabileceğimiz yapıları görelim.

ŞARTLI DÖNGÜ YAPISI WHİLE

Javascript kodu yazarken -programda bir önceki örnekte olduğu gibi- sayaç değişkeninin her değeri için istediğiniz işlemi yapmasını istemeyebilirsiniz. Bunun için while komutunu kullanırsınız ki bu Javascript'e "İstediğim işi şu şart sağlanıyorsa yap !" demiş olursunuz. While döngüsünde for döngüsünden farklı olarak döngü içerisindeki değişkenlerin tanımlanması gerekir. Şimdi yazım kurallarına bir göz atalım.

```
while ( döngü şartı ) { şart doğruysa yapılacak işlemler}
şart doğru değilse yapılacak işlemler
```

DO .. WHİLE YAPISI

Do ... while yapısı genel olarak bir döngünün yapısını eğer şart doğruysa tekrar et manasındadır. Yani do ile başlangıçta hiçbir koşul olmadan işlem yapılır. Daha sonra while şartı doğru ise tekrar do yapısında geri dönülür. Bunu bir örnek ile açıklamak gerekirse;

Örneğin bir ticari siteniz var. İnsanlar sizden gelip istedikleri ürünleri satın alıyorlar. Bir ürün için siparişlerini verdiler ve bizde bunun karşılığı olarak ücret + kargo + kdv miktarını hesapladık ve müşterimize dedik ki istediğiniz ürün şu fiyata şu gün elinizde olur. Bu hesaplamaların hepsini do yapısı ile yap dedik. Ve sonra sorduk daha başka ürünlerde almak istiyor musunuz? İşte bu da while yapısı ile sorulur. Şayet cevap evet ise do yapısı tekrarlanır değilse do döngü yapısında çıkılır.

Bu tür bir örnek yapalım ;

Bizim kitap, cd ve kaset sattığımız varsayalım. Bizden de 2 kitap ve 3 cd aldığını varsayarsak ;

```
var kitap=2000000; var cd=3000000; var kaset=1500000;
do {
var kitapistek =prompt ("Kaç tane kitap almak istiyorsunuz ?" , "lütfen rakam giriniz");
var cdistek= prompt ("Kaç tane cd almak istiyorsunuz ?" , "lütfen rakam giriniz");
var kasetistek= prompt ("Kaç tane kaset almak istiyorsunuz ?" , "lütfen rakam giriniz");
var kitaptutar=kitapistek*2000000;
var cdtutar=cdistek*3000000;
var kasettutar=kasetistek*1500000;
var toplamtutar = kitaptutar+cdtutar+kasettutar;
document.write (kitapistek ," tane kitap ", cdistek ," tane cd " , kasetistek , "tane kaset siparişiniz alınmıştır ", "<br>");
document.write ("<br>" , "Aldığınız ürünlerin toplam tutarı = " ,toplamtutar);
var istek =prompt("Başka ürünlerde satın almak istiyor musunuz ?", "e veya h giriniz");
}
while (istek="e")
document.write ("<br>" ,"Bizden alışveriş yaptığınız için teşekkürler")
```

BREAK VE CONTİNUE İFADELERİ

While komutu ile şartı belirledikten sonra yapılan işlemin kesilmesi veya devam etmesi otomatik hale gelmektedir. For döngüsü içerisinde de bu tür bir olayı break ve contine ifadeleri ile gerçekleştiririz.

Javascript break ifadesini gördüğü anda döngü işlemini keser ve bir sonraki komut satırını işleme koyar. Continue ifadesinde ise döngü break ifadesindeki gibi kesilir fakat işleme konulan satır bir sonraki satır değildir. Continue'de döngü başına dönülür.

Örnek:

```
for ()
{işlem1; işlem2; break; }
```

Burada işlem2 ile verilen kısımda örnek olarak bir sorgu yapılabilir. Sorgu doğru ise break ifadesine gelinir ve burada döngü kesilir.

```
for ()
{ işlem1; işlem2; continue;}
```

Burada yine işlem2 ile sorgu yapılırsa contine ifadesi ile döngünün devamı sürdürülür.

Önemli : Break ve Continue ifadeleri her komutu kesmek veya devam ettirmek için kullanılamaz. Mesela bir if (Eğer) ifadesi şart doğru değilse break ile kes denilemez. Sadece döngü içerisinde döngünün kesilmesi veya devam ettirilmesi için kullanılabilir.

SWİTCH-CASE İFADESİ

Bu ifade genel olarak menü kullanımında veya sorgu işlemlerinde işe yarar. Swicht ile ifade alınır case ifadesi ile işlemler sorgulanarak yapılır. Yazım kurallarına bir göz atalım.

```
switch (parametreler)
{ case "ifade1" :
case "ifade2" :
... }
```

Bir örnek verelim. Burada web sayfamızdaki herhangi bir işlemde çıkıp çıkmak isteyip istemediği soruluyor. Cevap evet ise işlem istenilen yönde yönlendiriliyor. Cevap hayır ise döngüden çıkılmaktadır. Burada kendimizi ziyaretçinin klavyesinde Caps Lock tuşuna basılı olup olmadığını önemsemiyoruz. Çünkü koşul ifademizi hem küçük harf hem de büyük harfe göre yazıyoruz.

```
var sec;
sec = prompt ("Çıkmak istiyor musunuz " ,"Evet için E veya e , Hayır için H veya h giriniz")
switch (sec)
{ case "e" : case "E" :
document.write ("Tekrar hoşgeldiniz")
//yapılması istenen işlemler
case "h": case "H" :
document.write ("Bizi tercih ettiğiniz çin teşekkürler")
break
//Çıkılması istendiği için döngüyü kesmek için break komutunu kullanıyoruz.
```

FONKSİYON KAVRAMI

Çoğu zaman Javascript kodunuzda bir işlemin birden fazla yapılması gerekebilir. Hatta kimi zaman Javascript'e bir işlem yaptırmadan önce başka bir işlemi yaptırmak istenebilir. İşte bu tür tekrarlanan işin yapılması için gerekli işlem ve komut gruplarına **Fonksiyon** adı verilir. Fonksiyonlar genelde, A isimli gruptaki işlemleri yap oradan bir değer al bunu B isimli gruba götür gibi işlemler için kullanılır. Bu tür komut sistemleri Javascript'te en çok kullanılan komut türlerindendir. Fonksiyonun yazım kuralları şu şekildedir:

```
function fonksiyon_ismi (parametre1 , parametre2 , .... )
{ yapılması istenen işlemler }
```

Fonksiyona Değer Gönderme ve Değer Alma

Bir fonksiyonun Javascript içerisindeki ilk önemli görevi diğer fonksiyonlardan veya herhangi bir yerden bir değer alıp onu kendi içerisinde işletip sonra istenilen fonksiyona veya yere göndermektir.

Mesela herhangi bir muhasebe işleminin yapılıp bize geri gönderilmesini istediğimiz düşünelim. Genel yapı olarak kodumuz şu şekilde olacaktır. Veri1 ve Veri2'nin işleme konulacağı fonksiyonların tanımlanması Veri1'in alınması Veri2'in alınması Veri1'in fonksiyona gönderilmesi Veri2'nin fonksiyona gönderilmesi Alınan verilerin ekrana yazdırılması

Şimdi bu genel kodu Javascript'te nasıl yapacağımızı görelim :

```
<html>
<head>
<script language="JavaScript">
<!-- //eski sürüm tarayıcılardan kodu gizleyelim
function veri1(ilkveri)
{ var ilktoplam = (ilkveri * 30 )/100 ;
return ilktoplam ; }
function veri2 (ikinciveri)
{ var ikincitoplam = (ikinciveri * 45 )/100;
return ikincitoplam; }
-- >
</script>
</head>
<body>
<script language="JavaScript">
<!—
```

```
var data1 = prompt ("Birinci miktarı giriniz" ,"rakam gir");
var data2 = prompt ("İkinci miktarı giriniz" , "rakam gir");
document.write ("İlk işleminizin sonucu = " , veri1(data1) );
document.write ("İkinci işlemin sonucu = " , veri2(data2) );
-- >
</script>
</body>
</html>
```

İlk satırların function tanımlama olduğunu görüyorsunuz. Burada veri1 , veri2 adlı iki tane fonksiyonu tanımladık. Diğer satırlarda prompt komutu ile klavyeden bilgi girişi sağladık. Daha sonra bu verileri fonksiyonlara göndererek istediğimiz işlemi yaptırdık ve daha sonrada bunu return yöntemiyle geri aldık. Bu kısma kadar yaptığımız fonksiyona bir değer göndermekti. veri1(data1) komutuyla prompt yoluyla aldığımız data1 değişkenini veri1 adlı fonksiyona gönderdik. Yani function veri1(ilkveri) şeklindeki fonksiyona biz data1 değişkenin gönderdik. Fonksiyon bu değeri yani data1 değişkenin aldığında otomatik olarak ilkveri değişkenine atadı. Böylelikle ilkveri=data1 oldu. Daha sonra istenilen işlemler yapıldı. Ve ardından return ilktoplam değeri geri gönderildi. Bu değer daha sonra ekrana yazdırıldı. Diğer veri2 adlı değişken için de aynı tür bir işlem sözkonusudur.

NESNELER VE ÖZELLİKLERİ

Günümüzde bilişim Teknolojileri ile ilgilenen hemen herkesin duyduğu bir terim var: Nesneye Yönelik Programlama (OOP). Nedir bu Nesneye Yönelik Programlama? Bu tip programlamada kullanılan her öğe bir nesne olarak kabul edilir. Bu nesnelerin özelliklerini kullanarak onları değiştirerek program yazılır. Javascript'te bu tür bir programlama dilidir. Örneğin webde sörf yaparken herkesin karşısına çıkan formlar birer nesnedir. Bu nesnelerin tepkiye göre cevap vermesi gibi özellikler de onun yani nesnenin özellikleridir.

Örneğin şimdiye kadar çoğu kez kullandığımız document.write komutu aslında bir nesnenin özelliğine atıfta bulunmaktan başka bir şey değildir. Yani document nesnesinin write özelliğini kullanarak html sayfamıza yazı yazdırıyoruz.

Window Nesnesi

Genel olarak pencere özellikleri ile ilgili bir nesnedir.

PENCERE AÇMAK VE KAPAMAK

Birçok yerde gördüğünüz pencere açma pencerelerin çeşitli özelliklerini değiştirme işte bu nesne yardımıyla yapılmaktadır. Şimdi bu nesneyi nasıl kontrol edeceğiz onu görelim.

Pencere açmak için :

```
window.open("Url_adı" , "pencere_adı" , "pencere_özellikleri");
```

Pencere kapatmak için :

```
window.close();
```

Pencere kapatmak için window.close() komutu vermek yeterlidir. Burada kapatılan pencere o anda kullanılmakta olan penceredir.

Pencere açarken ise, window.open ile pencerenin açılmak istendiği belirtilir. Parantez içerisinde verilenler ise açılması istenen pencerenin özelliklerini belirtir.

- **Url_adı :**

Buraya yazılacak dosya ismi açılacak pencerenin içerisinde olacaktır.

Örnek :

```
window.open("http://webteknikleri/owp/index.htm")
```

veya ;

```
window.open("index.html")
```

- **Pencere_adı :**

Bu açılacak pencerenin adını belirtir. Birden çok pencere ile işlemler yapıyorsanız hangi pencereye bir komut gönderdiğinizin belli olması için gereklidir.

Örnek :

```
Window.open("index.html" , "ana");
```

- **Pencere_özellikleri :**

Bu özellikte adından belli olduğu ölçüde pencerenin özellikleri ile ilgilidir. Bir pencerenin değiştirilebilir özellikleri şunlardır :

- menubar : Tarayıcıların en üst kısmında bulunan File(Dosya) , Edit(Düzen) vb. menülerin bulunduğu satırdır.
- toolbar : Tarayıcılarda üst kısımda Back(Geri) , Forward(İleri) vb. tuşların bulunduğu kısımdır.
- location : Tarayıcılarda ziyaret etmek istediğiniz web adresini yazdığınız kısım.
- status : Tarayıcıların en alt kısmında hangi dosyanın yüklendiği ile ilgili bilgi veren kısımdır.
- scrollbars : Sağ tarafta bulunan sürgü çubuklarıdır.
- resizable : Pencerenin boyutlarının kullanıcıya bırakılması veya kesin değerler almasıyla ilgilidir.
- width : Açılacak olan pencerenin piksel cinsinden genişliğidir.
- height : Açılacak olan pencerenin piksel cinsinden boyudur.
- left : Açılacak olan pencerenin ekranın sol tarafından kaç piksel uzaklıkta olacağını belirler.
- Top : Açılacak olan pencerenin ekranın üstünden kaç piksel aşağıda olacağını belirler.

Eğer pencere özellikleri kısmında değişmesini istemediğiniz bir özellik varsa onu yazmanıza gerek yoktur. Bu değerler tarayıcının standart(default) değerlerinden alınır.

Örnek:

Şimdi bir pencere açalım. Açtığımız pencerede dosya, düzen ve ileri, geri tuş takımı olmasın. Pencerenin boyutları 200x300 piksel olsun. Şimdi buna göre kodumuz:

```
window.open
("http://webteknikleri/owp/index.htm",
 "webteknikleri" ,
 " menubar=no,
   toolbar=no,
   scrollbars=yes,
   location=yes,
   width=200, heigt=300";)
```

WİNDOW.LOCATİON.PROTOCOL

Window nesnesinin location.protocol nesnesi ise yüklenen dosyanın sabit diskten mi yoksa internetten mi yüklendiğini gösterir.

- **http:** ile dosyanın internetten yüklendiğini belirtir.
- **file:** ile dosyanın sabit diskten yüklendiğin belirtir.

Mesela şöyle bir örnekle dosyanın nerden yüklendiğini kontrol edelim.

```
if (window.location.protocol == "http:" )
{ document.write ("Bu belge Internet'ten geliyor.") }
else
{ document.write ("Bu belge sabit diskten geliyor") }
```

WİNDOW.HİSTORY.GO

Window'un histrory özelliği ile bir önceki sayfaya erişim sağlanabilir. Örneğin kullanıcı herhangi bir formu doldurmadı ve işlem yapılamadı bu durumda bir hata mesajı ile kullanıcıyı uyardıktan sonra history nesnenisin kullanarak bir önceki sayfaya kullanıcıyı gönderebilirsiniz. Bunun için gerekli kod yazımı şu şekildedir.

```
Window.history.go(-1)
```

Bir önceki sayfaya -1 ile ulaşabilirsiniz. Bu değeri arttırarak daha önceki sayfalara da ulaşabilirsiniz.

STATUS BAR KULLANIMI

Status bar window nesnesinde belirttiğimiz gibi tarayıcıların en alt kısmında yer alan hangi dosyaya gidileceği veya yüklendiği ile ilgili bilgi veren kısımdır. Status barı değiştirmek için şu kodları yazmalıyız.

```
window.status="Webteknikleri'nden Merhaba !";
```

Bu şekilde kullandığımız bir status kodu ile sayfa açık kaldığı sürece Webteknikleri'nden Merhaba ! yazısı karşımızda olacaktır.

Tarayıcı Nesnesi

Tarayıcılar Javascript tarafından bir nesne olarak algılanır. Bu nesnenin özelliklerini şöyle sıralayabilir.

- appname Tarayıcı adı
- appVersion Tarayıcının Versionu
- appCodeName Tarayıcının kod adı
- userAgent Tarayıcının anamakinaya(server) kendini tanıtırken verdiği isim

```
<html>
<head><title>Tarayıcı Özellikleri</title></head>
<body>
<script language="javascript1.2">
<!—
document.write("Şu anda kullandığınız tarayıcının özellikleri :" , "<br>");
document.write(navigator.appname + navigator.appVersion +
navigator.appCodeName + navigator.useAgent ) ;
-->
</script>
</body>
</html>
```

Çerçeve (Frame) Nesnesi

Çerçeve tekniği bir web sayfası üzerinde birden fazla web sayfası görüntülenmek istendiğinde kullanılır. Daha ayrıntılı bilgi için HTML adlı bölümümüze bakınız.

Javascript açısından her bir çerçeve bir pencere sayılır. Bunların içeriğini kontrol etmek için belli komut stilleri vardır. Şimdi onları görelim:

- Top : En üst pencere (Yani tarayıcının kendisi)
- Parent : Herhangi bir çerçeveyi oluşturan düzenleyici bölüm
- Self : Çerçevenin kendisi

Javascript çerçeve düzenleyicileri (FrameSet) içerisindeki diğer alt çerçeveleri 0'dan başlayarak numaralandırır. Bu numaralar yardımıyla çerçeve özelliklerini değiştirebiliriz. Örneğin iki tane çerçeveye sahip bir sayfada birinci çerçeve parent.frames[0] diğeri ise parent.frames[1] olarak adlandırılır. Örneğini verdiğimiz gibi iki çerçeveli bir web sayfamız olduğunu varsayalım.

Ana sayfamız ki bu sayfa tarayıcıya sayfanın iki html sayfasında oluştuğunu söyleyen, çerçeve düzenleyicisinin olduğu sayfanın kodları şu şekilde olsun.

1. İlk sayfayı frame.html olarak kaydedin.

```
<html>
<head><title>Frame (Cerceve)</title></head>
<!-- frames -->
<frameset cols="50%,*">
<frame name="sol" src="sol.html">
<frame name="sag" src="sag.html">
</frameset>
</html>
Bu sayfayı sol.html olarak kaydedin.
<html>
<head><title>Sol Frame</title></head>
<body>
<script language="javascript1.2">
<!—
parent.frames[0].document.write("Herhangi bir hesaplama vb.unsur icin
kullanilacak kod turu", "<br>" , "SOL taraf icin");
-->
</script>
</body>
</html>
```

2. Aşağıdaki sayfayı sag.html olarak kaydedin.

```
<html>
<head><title>Sag Frame</title></head>
<body>
<script language="javascript1.2">
```

```
<!—
parent.frames[1].document.write("Herhangi bir hesaplama vb.unsur icin
kullanilacak kod turu", "<br>" , "SAG taraf icin" );
-->
</script>
</body>
</html>
```

Form Nesnesi

Javscript açısından Html'in en önemli nesneleri Formlardır. Çünkü ziyaretçi ile etkileşmede en büyük unsurlardan birisi Formlardır. Html kendi form nesnesini kendisi oluşturabilir. Bu bakımdan Javascript'e düşen bir göre yoktur. Javascript form verilerinin yorumlanması ve işlenmesinde devreye girer. Şimdi form unsurlarını tanıyalım :

- Name : Formun ismi
- Action : Formun işleneceği perl veya cgi programının tanımlanacağı etiket
- Enctype : Formun kodlanma türü
- Method : Formun gönderme(post) mi yoksa alma(get) işlemi göreceğini belirler.
- Target : Pencere ismi
- OnSubmit : Gönderme metodunun ismi

Bunlardan yararlanarak bir form nesnesinin kodunu yazalım.

```
<form name="mail" action="http://www.webteknikleri.com/cgi-bin/mail.pl"
method="POST">
Form unsurları
</form>
```

Şimdi biz bu kodda "Form Unsurları" diye bir şeyden söz ettik. Bu form unsurları ziyaretçiden nasıl bilgi alınacağını belirleyen unsurlardır. Bunlar bir metin alanı veya aşağı düşmeli bir menü olabilir. Form etiketi içerisindeki tüm unsurlar element adlı dizi-değişken içerisine yazılırlar ve form unsurları kullanılırken bu tip bir atıfta bulunmanız gerekir.

TEXT ALANI

Text alanı form unsurlarının en önemlilerindendir. Ziyaretçilerden bilgi almak amacıyla kullanılır. Kullanımı şu şekildedir.

```
<input type="text">
```

Şeklinde kullanılır. Bu nesnenin kullanılan etiketleri :

- Name : text alanının ismi
- Size : text alanının web sayfasında görülecek kısmının büyüklüğü
- Maxlenght : text alanına en fazla kaç karakter girilebileceğini belirler.

İşte bir tam teşekküllü text alanı :

```
<form name="mail" action="http://www.webteknikleri.com/cgi-bin/mail.cgi"
method = POST>
<input type="text" name="email" size=15 maxlenght=40>
</form>
```

Buraya kadar işimiz Html ile hallettik. Şimdi bu form nesnesinin özelliklerini Javascript aracılığıyla nasıl değiştirilebileceğini görelim.

- document.form_ismi.elements[numara]. değiştirilmek_istenen özellik.
- numara : değiştirilmek istenen elemanın numarasıdır.

Değiştirilmek istenen özellikler şunlar olabilir.

- value : içerisindeki değer
- lenght : nesnenin uzunluğu
- name :ismi

Şimdi bir örnek verelim (bu örnek bir önceki form içindir).

```
document.mail.elements[0].lenght=20
```

ŞİFRE ALANI

Bu alanlar şifreli bilgi almak için kullanılır. Bu alana bir bilgi girildiğinde karakterler gözükmez onun yerine asteriks * işareti görülür.

Örnek:

```
<form action="http://" name="mail">
```

```
<input type="Password" name="sifre" >
</form>
```

Bu tür form unsurlarına erişimde text alanı gibi aynı şekildedir.

BUTONLAR

Form unsuru olarak iki tür buton vardır. Bunlar form unsurlarını form görevine göre göndermeye veya almaya yönelik Gönder (Submit) düğmesi bir diğeri ise Form unsurlarının tümünün ilk halini almasını sağlayan Sil (Reset) düğmesidir. Şimdi bunların nasıl kullanıldıklarını görelim.

```
<form action="http://" name="mail">
<input type="Submit" name="gonder" value="GONDER"><br>
<input type="Reset" name="sil" value="SIL">
</form>
```

RADYO (RADİO) DÜĞMELERİ

Bu tür düğmelerin en büyük özelliği radyo düğmeleri gibi olmasıdır. Yani herhangi bir menü üzerinde sadece bir seçim yaptırmak istiyorsanız bu tür form öğelerini kullanırsınız. Şimdi bunun ile ilgili bir örnek yapalım.

```
<HTML>
<HEAD>
<TITLE>Radio</TITLE>
<SCRIPT LANGUAGE = "JavaScript1.2">
function sorgu (secim)
{var deger = null
for (var i=0; i<secim.length; i++)
{if (secim[i].checked)
{ deger = secim[i].value
break } }
return deger }
</SCRIPT>
</HEAD>
<BODY>
<FORM name="form1"> <p>
<input type=radio name="firma" value="Bilemediniz Yazilim">Microsoft</p>
<p><input type=radio name="firma" value="Bilemediniz Yazilim">Borland</p>
<p><input type=radio name="firma" value="BilemedinizYazilim">Adobe</p>
<p><input type=radio name="firma" value="Tebrikler Bildiniz">Copmaq</p>
```

```
<input type=button value="Bunlardan hangisi bilgisayar ureticisidir"
onClick="alert(sorgu(this.form.firma))">
</FORM>
</BODY>
</HTML>
```

Gördüğünüz gibi bu form unsuruna da(öğesi) diğer form unsurları gibi aynı şekilde ulaşılır. Fakat diğerlerinden farklı olarak burada checked(seçilmiş) yardımcı etiketini kullandık. Burada Javascript ile bir döngü oluşturarak hangi nesnenin seçili(checked) olduğunu kontrol ediyoruz. Ve alert ile sorulan sorunun cevabının doğruluğunu ziyretçiye bildiriyoruz.

SELECT UNSURU

Select öğesi form nesnelerimizden menü yoluyla ziyaretçi ile etkileşme yollarından bir tanesidir. Bu bir tür seçme kutusudur. Aşağı düşmeli kutu sayesinde kutu içerisindekilerden birisini seçebilirsiniz. Name , Multiple ve Size özelliklerine sahiptir. Bu nesnemizde nesnenin yönelendirilmesi açısından onBlur, OnFocus, OnChange özellikleri kullanılabilir. Nesnenin özelliklerine ulaşım için en çok kullanılan etiket yardımcısı ise value(değer) dır.

Radyo kutularında yaptığımız örneği şimdide Select öğesine uygulayalım.

```
<HTML>
<HEAD>
<title>Select</title>
<SCRIPT LANGUAGE = "JavaScript1.2">
function secim(secilen)
{ return secilen.options[secilen.selectedIndex].value }
</SCRIPT>
</HEAD>
<BODY>
<FORM name="soru">
<p><SELECT NAME="firma">
<OPTION value="Bilemediniz Yazilim">Microsoft</OPTION>
<OPTION value="Bilemediniz Yazilim">Borland</OPTION>
<OPTION value="Bilemediniz Yazilim">Adobe</OPTION></P>
<OPTION value="Tebrikler Bildiniz">Compaq</OPTION></P><br>
<input type=button value="Bunlardan hangisi bilgisayar ureticisidir"
onClick="alert(secim(this.form.firma))">
</FORM>
</BODY>
</HTML>
```

OLAYLAR

Ziyaretçiye sunulan bir web sayfası üzerinde ziyaretçinin yaptığı her tür hareket bir bağlantıyı tıklaması , bir resmin üzerine gelmesi , resmin üzerinde ayrılması , bir formu yanlış doldurup hataya yol açması hep bir olaydır.

onClick

Web sayfası üzerinde bir nesnenin mosue ile üzerine tıklanması sonucu onClick olayı gerçekleşir. Olayın gerçekleşmesi için nesneni tıklanıp bırakılması gereklidir. Yani mouse tuşuna basıldığında onClik olayı gerçekleşmiş olmaz. onClick olayı tuşa basılıp bırakıldıktan sonra gerçekleşir. Bağlantılar, resimler, form düğmeleri tıklanabilecek nesneler arasındadır. OnClick yönlendiricisine bu durumda ne yapacağını Html etiketleri arasında bildirmeniz gerekir.

Örnek:

```
<html><head><title>onClick</title>
<script language="javascript1.2">
<!—
function merhaba()
{alert ("beni tikladiniz"); }
-->
</script></head>
<body>
<input type="button" name="tikla" value="tikla" onClick=merhaba()>
</body>
</html>
```

Burada yaptığımız işlem html etiketleri arasında yer verdiğimiz bir butona tıklama (onClick) ile ne yapacağını merhaba fonksiyonuna git diyoruz. Fonksiyonda ekrana bir uyarı ile beni tıkladınız diye bir uyarı mesajı geçiyor.

Şimdi burada alert nesnesini de görmüş olduk. Alert nesnesi ziyaretçiye herhangi bir durumda uyarı vermek amacıyla kullanılır. Görüldüğü üzere parantez içerisinde çift tırnak içine uyarı yazısı yazılır.

OnDblClick olayı da onClick olayı ile yapı olarak aynıdır. onClick'ten farkı nesneye çift tıklandığında çalışmasıdır. Diğer yöntemler onClick ile aynıdır.

onMouseOver , onMouseOut

Bu tür nesne olayları ingilizce adı (onMouseOver = mouse işaretçisi/imleç üzerindeyken , onMouseOut = mouse işaretçisi üzerinden ayrıldığında) üzerinde olmakla birlikte mouse-un nesnenin üzerinde olup olmadığı ile ilgilenir.

Örnek:

```
<html>
<head><title>onMouseOver ve onMouseOut </title>
<script language="javascript1.2">
<!—
function uzerinde()
{window.status="Tıklayın ve Webteknikleri.com adresine gidin" }
function disinda()
{window.status="Webteknikleri.com baglantisina tıklayın" }
-->
</script></head>
<body>
<a href="http://www.webteknikleri/index.htm" onMouseOver = uzerinde()
onMouseOut =disinda()> Webteknikleri.com </a>
</body>
</html>
```

onMouseOver ve onMouseOut metodu ile ilgili bir diğer örnek:

```
<html>
<head><title>OnMouseOver</title>
<script language="javascript1.2">
<!—
function altavista()
{document.web.mesaj.value="En unlu web motorlarindan biri" }
function altavistasil()
{ document.web.mesaj.value="" }
function yahoo()
{ document.web.mesaj.value="En unlulerden bir tane daha" }
function yahoosil()
{document.web.mesaj.value="" }
function hotbot()
{document.web.mesaj.value="Ve bir tanesi daha" }
function hotbotsil()
{document.web.mesaj.value="" }
-->
</script></head>
<body>
```

```
<a href="www.altavista.com" onMouseOver="altavista()" onMouseOut =
"altavistasil()">
Altavista</a><br>
<a href="www.yahoo.com" onMouseOver="yahoo()"
onMouseOut="yahoosil()">Yahoo</a><br>
<a href="www.hotbot.com" onMouseOver="hotbot()"
onMouseOut="hotbotsil()">Hotbot</a><p>
<form name="web">
<input type="text" name="mesaj" size="50">
</form>
</body>
</html>
```

onSubmit

Web üzerinde sörf yaparken çoğunlukla karşımıza çıkan formlar biz doldurduktan sonra sayfanın bağlı bulunduğu server (ana makine) ya gönderilir. Fakat biz bu onSubmit olayı ile form gönderilmeden önce formun düzgün doldurulup doldurulmadığını kontrol edebiliriz.

Bunu örnek bir kod ile açıklayalım. Html sayfamızda body etiketleri arasında doldurulmasını istediğimiz bir form var ve ilgili kodun başlangıcı şöyle :

```
<form action="mail.pl" method="post" onSubmit="dogrula()">
```

Bu satır ile formun gönderilmesiyle (onSubmit) dogrula fonksiyonunu çağırıyoruz. dogrula fonksiyonu da şu şekilde olabilir (Bu fonksiyon head etiketleri arasında olan script etiketleri arasında olmalıdır).

```
function dogrula()
{ confirm ("Formu düzgün doldurduysanız OK'i tıklayınız'); }
```

Bu fonksiyonda kullandığımız confirm nesnesi ile kullanıcıya OK ve Cancel tuşları ile "Emin misin ? Form Gönderiliyor!" denilmektedir.

onReset

Bu olay ile web sayfanızda bulunan formdaki yazdıklarınızın tamamen silinir. Yani yazdığınızın yanlış olduğunu farkettiniz bu durumda Sil (Reset) tuşunu tıklarsınız ve size boş bir form gelir. Yalnız burada birşeyi belirtmek isterim. Reset(Sil) tuşuna tıkladıktan sonra tarayıcının back(geri) düğmesini tıkladığınızda formunuzda yazdıklarınız tekrar geri gelmez. Fakat siz onReset olayı ile bu durum için son bir ziyaretçiye seçenek sunabilirsiniz.

Örnek:

```
<html>
<head><title>onReset</title>
<script language="javascript1.2">
<!—
function sil()
{ return confirm('Silmek istediginize emin misiniz?'); }
-->
</script>
</head>
<body>
<form onReset="return sil()">
<input type="text" name="mail">
<input type="reset" value="sil">
</form>
</body>
</html>
```

onChange

Web sayfası üzerinde ziyaretçinin değiştirebileceği üç tür alan vardır. Bunlar text (metin) textarea (geniş metin alanı) select (seçim alanı) dır. Mouse u bu alanlar üzerine getirip tıkladığınızda onChange(değişti) olayını gerçekleştirmiş olursunuz. Şimdi bunu select yöntemi ile nasıl olduğunu görelim. Örneğimizde aşağı düşmeli bir menü tasarlayacağız ve şeçili durumda olan web sayfasına gönderme yapacağız.

```
<html>
<head><title>OnChange</title>
<script language="javascript1.2">
<!—
function degisti()
{ window.open("www.altavista.com"); }
-->
</script>
</head>
<body>
<form method="post">
<p><select name="degistir" size="1" onChange="degisti()">
<option>Adresi tikla
<option>Altavista
</select>
</form>
</body>
```

```
</html>
```

onLoad, onUnLoad

Bu olaylar bize sayfanın yüklenmeye başlamasında (onLoad) sayfadan ayrılıncaya (onUnLoad) kadar olan yapılacak işlemler için gereklidir. Bir Javascript fonksiyonun web sayfası yüklenmeye başladığında otomatik olarak çalışmasını istiyorsak onLoad olayını kullanırız. Eski DOS'çular bilirler Autoexec.bat dosyası nasıl makine açıldığında yapılmak istenenleri yapıyorsa onLoad olayında da sayfa yüklenmeye başladığında nelerin otomatik olarak başlatılacağını belirleyebiliriz. Mesela sayfa yüklenmeye başladığında (onLoad) ziyaretçiye "Web sitemiz hoş geldiniz" diyebiliriz. Sayfadan ayrıldığında (onUnLoad) ise "İyi sörfler" diyebiliriz. Örnek kodlara geçmeden önce şunu belirtmekte yarar var. Bildiğiniz üzere web sayfası kod açısından iki kısıma ayrılır. Bunlar head ve body kısmıdır. Tarayıcı açısından body kısmı asıl kısımdır. Head kısmında sayanın nasıl görüntüleneceği gibi bölümler yer alır. Bu yüzden onLoad ve onUnload kısmı body etiketleri arasında yer alır.

Örnek:

```
<html><head><title>onLoad onUnLoad</title>
<script language="javascript1.2">
<!—
function hosgeldiniz()
{
alert("Web Sitemize Hosgeldiniz")
}
function gulegule()
{
alert("Iyi sorfler..")
}
-->
</script>
</head>
<body onLoad="hosgeldiniz()" onUnload="gulegule()">
</body>
</html>
```

onError, onAbort

Ziyaretçi sayfayı herhangi bir neden yüzünden tam haliyle yükleyememiş olabilir. Bu nedenler aktarım hızı veya tarayıcının Javascript kodunu tam manasıyla yorumlayamamış olmasıdır. İşte bu durumda Error(hata) oluşur. Html üzerinde oluşan en sık error(hata) resim haritalarının (image-map) tam anlamıyla yüklenmemesinden

kaynaklanır. Çünkü bu durumda resim tam yüklenmemiştir. Bu da ziyaretçinin resim üzerinde tıklayacağı yerlerin yorumlanmamasını doğurur.

Örnek:

```
<img src="resim.gif" onError="alert("Resim dosyası tam olarak
yüklenemedi')">
```

Ziyaretçi resimlerin yüklenmesi çok uzun sürüp yüklemeyi stop(dur) tuşu ile kestiyse bu durumda onAbort olayı gerçekleşir. Bunun sonucu olarak ziyaretçiye bir hata mesajı verebilirsiniz. Bu durum daha önce bahsettiğimiz image-map ler içindir.

```
<img src="resim.gif" onAbort="alert("Resim harita dosyası tam olarak
yüklenemedi. İlgili resim bir harita olduğu için yüklenmesini tavsiye
ederiz.')">
```

JAVASCRİPT İLE DHTML

Bu kısımda Javascript ile Katman (layer) özelliklerinin nasıl değiştirilebileceğini göreceğiz. Javascript bize Html sayfamızı oluşturan önemli unsurlardan biri olan layer(katman) ların tüm özelliklerini değiştirmemize olanak sağlar. Ayrıca hemen her yerde gördüğünüz resim değiştirme tekniğini de göreceğiz.

Katman Özelliklerini Değiştirme

İşe katman nedir sorusuyla başlayalım. Katman adı üzerinde sayfamızın üzerinde ne sayfadan bağımsız ne de her yönüyle sayfamıza bağlı bir unsurdur. Katman kullanarak istediğimiz herhangi bir yapıyı (yazı,resim,video,form) sayfamızın istediğimiz yerine koordinatları vermek koşulu ile yerleştirebiliriz. Zaten katmanın kullanım alanı en çok budur. Şimdi bir katman oluşturalım ve değiştirilebilir özelliklerini görelim.

```
<html>
<head><title>Layer</title></head>
<body>
<div id="denem" style="position:absolute ; left:100px ; top:200px;
width:300px ; height:400px ; visibility:visible" >
Su anda bir katman(layer)in icerisindeyim
</div>
</body>
</html>
```

Layer oluşturmak istediğinizde etiketi ile başlar etiketi ile kodunuz tamamlarsınız. Şimdi katman özelliklerine geçelim :

- id : Katmanın ismi
- style : Katmanın özelliklerini belirtir
- absolute : Katmanın koordinatlarının kesin olacağını belirler
- left : Katmanın soldan kaç piksel sonra başlayacağını belirler
- top : Katmanın üstten kaç piksel sonra başlayacağını belirler
- width : Katmanın kaç piksel genişliğinde olacağını belirler
- height : Katmanın kaç piksel boyunda olacağını belirler
- visibility : Katmanın görünür mü görünmez mi olacağını belirler

Internet Explorer ve Netscape tarayıcılarının doküman nesne modelleri farklı olduğundan katmana ulaşma teknikleri de farklıdır.

- **Internet Explorer kod tekniği:**

```
katman_adı.style.değiştirilmesi_istenen_özellik=yeni_değer;
```

Örnek :

```
deneme.style.left=50px;
```

- **Netscape Navigator kod tekniği:**

```
document.katman_adı.değiştirilmesi_istenen_özellik=yeni_değer;
```

Örnek :

```
document.deneme.left=50px;
```

Örnek (katmanın yerinin değiştirilmesi):

```
<html><head><title>Katman</title>
<script language="javascript1.2">
<!—
function tara()
{ var tarayici= navigator.appName
if (tarayici=="Netscape") degisim = document.katman;
if (tarayici=="Microsoft Internet Explorer") degisim = katman.style; }
function hareket1() {
degisim.left=100
degisim.top=100 }
function hareket2() {
degisim.left=300
degisim.top=300 }
-->
```

```
</script></head>
<body onLoad="tara()">
<div id="katman" style="position:absolute ; left:400px; top:10px">
Bu katmanin yeri degisecek
</div>
<p><p><p>
<a href="javascript:hareket1()">Burayı tıklayın ve katmanınız 100x100'e
gitsin</a><br>
<a href="javascript:hareket2()">Burayi tıklayın ve katmanınız 300x300'
gitsin</a>
</body></html>
```

Buradaki örnekte olduğu gibi sizde katmanın diğer özelliklerini (width,height) değiştirebilirsiniz. Fakat görünebilirlik özelliği için özel bir durum vardır. Katman özelliklerine erişimde olduğu gibi bu özellikte de Internet Explorer ve Netscape Navigator farklılıkları vardır.

- **Internet Expolorer için Görünebilirlik özelliği;**

Katmanı görünebilir kılmak için:

```
katman_adı.style.visibility="visible"
```

Katmanı gizleyebilmek için:

```
katman_adı.style.visibility="hidden"
```

- **Netscape Navigator için Görünebilirlik özelliği**

Katmanı görünebilir kılmak için:

```
document.katman_adı.visibility="show"
```

Katmanı gizleyebilmek için:

```
document.katman_adı.visibility="hide"
```

Şimdi de bununla ilgili bir örnek yapalım.

```
<html><head><title>Katman</title>
<script language="javascript1.2">
<!—
function sakla()
{ var tarayici= navigator.appName
if (tarayici=="Netscape") document.katman.visibility="hide"
if (tarayici=="Microsoft Internet Explorer")
katman.style.visibility="hidden" }
function goster()
{ var tarayici= navigator.appName
```

```
if (tarayici=="Netscape") document.katman.visibility="show"
if (tarayici=="Microsoft Internet Explorer")
katman.style.visibility="visible" }
-->
</script></head>
<body>
<div id="katman" style="position:absolute ; left:400px; top:10px">
Bu katmanin tikladiginizda yok olacak
</div><p><p><p>
<a href="javascript:sakla()">Burayi tiklayin ve katmaniniz yok olsun</a><br>
<a href="javascript:goster()">Burayi tiklayin ve katmaniniz geri gelsin</a>
</body></html>
```

Örnekte de görüldüğü gibi onMouseOver ve onMouseOut olay yönlendiricilerini de kullanarak çok daha güzel şeyler üretebilirsiniz.