JAVASCRIPT

Önsöz

Eskiden, medrese zamanı, o zamanki öğrenciler de bizden farklı olmayıp, dersi kaynatmak istedikleri zaman müderrise, "Hocam; hangi velinin kızını dağda kurt yedi?" diye sorarlarmış. Hoca da başlarmış anlatmaya: "Bir kere veli değil, nebi idi; dağda değil çölde idi; kızı değil, oğlu idi. Kurt demedi, 'Yedi!' dediler.." dermiş.

Son bir yıldır en sık karşılaştığım soru: "Web siteme veya filanca sayfama bir Java koydum; ama çalışmıyor! Bir bakar mısınız?" Müderrisin hesabı, bu soruyu baştan yanıtlasak iyi olacak:

Bir kere Java değil, Javascript. Çalışmıyor değil, sayfada bu Script'i çalıştıracak olay ya da komut yok! Ayrıca başka bir Web sayfasında çalışan bir programın zorla yamandığı her sayfada çalışması mümkün olamaz. İşin temizi şu işi baştan öğrenmek! Sonuç itibariyle Web siteniz varsa veya açmayı düşünecek kadar ilerlediyseniz, HTML'i öğrendiniz demektir. Javascript'in, programlama dili sayarsanız, HTML'den zor bir tarafı yok! Bütün mesele, işi yarım bırakmamak; iki satır komut öğrenip, eski usul, başkalarının Javascript metinlerini kesip-kesip yapıştırma yoluna gitmemek.

Bu kitapçığı elinize alıp, bilgisayarın karşısına geçip, çok değil, bir haftanızı verirseniz, "Javascript biliyorum!" diyecek kadar Javascript öğrenebilirsiniz. Gerçi Javascript'in temeli olan ECMAScript ile ilgili kurallar kitabı 1.500 sayfa ama! Varsın olsun. Niyetiniz Javascript ile yeni bir oyun programı yazmak değilse, bu kitapçıktaki bilgiler işinize yarayacak kadar bu dili öğrenmenizi sağlayacaktır.

Konuya girmeden bir iki noktaya açıklık kazandıralım. "Javascript, Java değildir," dedik. Peki nedir? Javascript, Netscape firması tarafından hep ürküntü ile duya geldiğimiz C dilinden türetilmiştir. Java ise, Sun firması tarafından Pascal ve Delphi dillerinden

esinlenerek yazılmıştır. Javascript, HTML gibi, Visual Basic Scripting Edition (VBScript) gibi, kodları (komutları) düz yazı dosyası olarak kaydedilen ve işleyebilmesi için bir yorumlayıcıya ihtiyacı olan bir dildir. Java ile yazdığınız program ise (en azından teoride) işletme sisteminden bile bağımsız olarak, kendi başına çalışabilir.

Netscape, HTML'in dil olarak yetersizliğini Web Browser programının ikinci sürümüne hazırlanırken gördü. Bir kere bir Web sayfasına bir "iş yaptırmak" istediğiniz zaman bunu HTML ile yaptıramıyordunuz. Ayrıca HTML yeteri kadar dinamik değildi. Mesela, Web ziyaretçisi radyonuzun program listesinin yer aldığı sitenize bağlandığında kendisine programları güne göre sunmak istediğinizde, bir CGI programı yazmanız veya yazdırmanız gerekiyordu. CGI (Common Gateway Interface) Web ziyaretçinin Browser'ı ile sizin sitenizin bulunduğu Web Server'ın buluştuğu nokta. Bu noktada yapılacak işler, çalıştırılacak programlar Web Server'da çalışıyor demektir. Her yeni ziyaretçi, her yeni CGI programı Web Server'ınıza ek yük yüklenmesi, yani Web Server'ın giderek yavaşlaması demek. Siteleri dinamik hale getirmeyi amaçlayan ve Web sayfasının "eğer.. ise.." sorusunu sormasını gerektiren böyle küçük programları Server'da değil de, ziyaretçinin bilgisayarında çalıştırabilmek, Server'ın yükünü azaltabilirdi. Bu düşünceden yola çıkan Netscape, LiveScript adıyla bir "Düz yazı dili" geliştirdi. O sırada Sun firmasının geliştirdiği, PC veya Mac farkı gözetmeden, Windows, Unix veya başka bir işletim sistemine tabi olmadan çalışacak bir dil üzerinde çalışıyor, ve böyle bir dilin geliştirildiği haberleri bile bilgisayar programcıları arasında heyecan fırtınalarına sebep oluyordu. Ortalığı bir "Java" fırtınası kaplamıştı. Netscape, akıllıca bir davranışla 1995 Aralık'ında bu dili piyasaya Javascript adıyla sürdü. Netscape 2.0, "Javascript uyumlu" olarak çıktı. O sırada Microsoft da Internet Explorer adlı Browser'ına bu imkanı kazandırmak istiyordu, ama Netscape bir yıla yakın Javascript'i paylaşmaya yanaşmadı. Microsoft, Netscape'in Javascript yorumlama kodunu almadan, IE'i Javascript'i anlar hale getirdi; fakat ortaya çıkan dile JScript adını verdi. Jscript 1.0 ile Javascript 1.0 tümüyle aynı idi ve her iki program da hem Javascript, hem de JScript

anlayabiliyordu. Fakat bu durum çok uzun sürmedi; her iki firma da kendi Script diline diğer programın tanımadığı özellikler kazandırdılar. Bu arada Microsoft, IE'ye, JScript'te de, Javascript'te de olmayan becerilerle donattığı VBScript'i tanıma özelliği kazandırdı.

Şu anda Netscape firmasının Netscape Navigator'ı üçüncü sürümüne varan JScript'i ve VBScript'i tanımıyor, yorumlayamıyor. Buna karşılık IE, her üç Script dilini (Javascript'in bazı komutları hariç) tanıyabilir, yorumlayabilir. VBSCript, günümüzde daha çok Microsoft'un Web Server programı olan Internet Information Server (IIS) ile çalışan Web Server'larda, Server için yazılan programlarda kullanılıyor. Bu tür programlarda, mesela Active Server Pages teknolojisi ile yazılacak Server uygulamalarında da Javascript kullanılabilmesi, Javascript dilini bir Web tasarımcısının öncelikle öğrenmesini gerekli kılıyor.

Bunu gerekli kılan başka gelişme daha var. Avrupa Birliği standart kurumu, daha sonra kavgayı bir kenara bırakıp, Script dilinde ortaklığa gitmenin daha doğru olduğunu gören Microsoft ve Netscape firmalarının ortaklaşa sunduğu öneriye dayanarak. Javascript ve JScript'i ECMAScript adıyla birleştirecek bir standart yayınladı. Kısa bir süre sonra, Javascript'e yüzde 99 benzeyen ortak bir Script diline kavuşacağız. Bu dili çıktığı gün bilir olabilmek için bugünden Javascript öğrenmek gerekir.

Özetlersek:

1. Javascript, HTML'in içine gömülür; gerçek Java programları veya programcıkları ise yani Java Applet'ler HTML içinde sadece zikredilir, HTML'in parçası olmazlar.

2. Javascript Browser'a bağımlı olarak yorumlanır, yani Javascript'in bazı komutları Netscape'te başka, IE'de başka sonuç verebilir veya hiç sonuç vermeyebilir.

3. Javascript ile, ortaya uzatması .EXE olan bir dosya çıkartmazsınız; Javascript kodları HTML'in içine, kendisini HTML'den ayırteden <SCRIPT>...</SCRIPT> etiketlerinin arasına gömülür (isterseniz, içinde Javascript kodları bulunan bölümü ayrı bir düzyazı dosyası ve adının uzatmasını da ".js" yaparak HTML sayfasına LINK edebilirsiniz). Javascript,

bir Script dili olduğu için illa şu ya da bu kurala uygun yazılmaz; son derece esnek bir dildir. Javascript yorumlanan bir dildir: kendisi bir takım nesneler oluşturmaz; kendisini yorumlayacak çerçeveye programa (yani Netscape Navigator'ın veya Internet Explorer'ın nesnelerini kullanır. Bu, Javascript öğrenmenin asıl büyük bölümünün Browser programının ne gibi unsurları, bu unsurların ne gibi özellikleri olduğunu ve bu özelliklere hangi olay sırasında nasıl hitap edileceğini bilmek anlamına gelir.

4. Ve son nokta, Javascript, ancak bir olay halinde işler: bu olay ziyaretçinin Browser programının bir iş yapması, bir işi bitirmesi veya ziyaretçinin bir yeri tıklaması ya da klavyesinde bir tuşa basması demektir.

Peki, Javascript'i tanıdığımıza göre, biraz da onunla neler yapabileceğinizden söz edelim. Gerçi su kitapçığa ilgi gösterdiğinize göre, Javascript'in kullanım alanlarına o kadar da yabancı olmadığınız anlaşılıyor; fakat kısaca "client-side application" (istemci-tarafındaki uygulama) programlarına değinmemiz gerekir. Yukarıda dedik ki, Javascript bir Web sayfasında oluşturulmak istenen bazı etkileri yerine getirecek işlerin Web Server'da değil, ziyaretçinin bilgisayarında yapılması maksadıyla geliştirildi. O halde Javascript programları Web ziyaretçisinin bilgisayarında çalışacaktır. HTML kodlarınızın arasına gömdüğünüz Javascript kodları, kendilerini harekete geçirecek işareti görünce, ne yapmaları gerekiyorsa, o işi yaparlar. Bu işler neler olabilir? Çok çerçeveli bir sayfada belirli çerçevelerin içeriğini belirleme işi Javascript ile yapılabilir. Ziyaretçilerinizden sayfalarınızda bazı formlar doldurmalarını istiyorsanız, bu formların kutularına yazılan bilgilerin, arzu ettiğiniz türden olup olmadığı Javascript ile anında denetlenebilir. (Hayır; Javascript, kendiliğinden, sitenizi ziyaret edecek kişilerin kredi kartı numarasını edinemez!)

Javascript öğrendiniz diye her yere Javascript kodları serpiştirmeli misiniz? Hayır. HTML ile yapabileceğiniz bir işi asla Javascript ile yapmamalısınız. Bu iyi tasarım ilkelerine aykırıdır. Bir kere, sayfanıza koyacağınız her Javascript kodu, sizin kendi yükünüzü arttırır; belirttiğimiz gibi Javascript henüz ve muhtemelen epey bir zaman için Browser-bağımlı

olarak kalacak, yani kodlarınızın IE ve Netscape'de yorumu daima yüzde 100 aynı olmayacaktır. Bu yüzden yazacağınız her Javascript kodunu her iki Browser'da ve farklı sürümlerinde sınamak zorundasınız. İkincisi, Javascript kodunuz, ne kadar kısa olursa olsun, ziyaretçinin Browser'ının yükünü arttıracaktır. Hiç kimsenin sizin programcılık becerisi gösterinizi izleyecek zamanı yok! Javascript veya başka herhangi bir Script programına, ancak ve sadece HTML'in yetersiz kaldığı durumlarda başvurabilirsiniz.

Sonuç: Javascript, Java değildir, ama ona yakın beceriklilikte bir programlama dilidir. Programlama dilleri arasında öğrenme kolaylığı bakımından en ön sırada yer alır. Ve en önemlisi, Javascript olağanüstü zevklidir. Gerçek hikayeyi anlattıkça, Javascript ile neler yapabileceğini gördükçe, bir daha başkasının Javascript kodunu alıp, "Neden işlemiyor?" diye üzülmektense, kendi Javascript'inizi yazmanın daha kolay olduğunu da göreceksiniz.

Bu kitapçığı, okurun iyi, hem de bayağı iyi düzeyde HTML bildiği varsayımıyla ve programlama konusunda hiç deneyimi olmayan bir Web tasarımcısını dikkate alarak kaleme aldık. Eğer HTML konusunda kendinizi hazır hissetmiyorsanız, söz gelimi, Byte dergisinin Eğitim Dizisi'nde çıkan HTML Rehberi kitapçığını okumanızı öneririm. Bir programlama dili biliyor ve şimdi Javascript'i öğrenmek için bu kitapçığı okuyorsanız, programlamanın temel ilkeleriyle ilgili cümleleri veya bölümleri okumak zorunda değilsiniz! Yine bu kitapçıkta hemen her fırsatta Javascript'e ilişkin bütün terimlerin İngilizcesini göstermeye ve kullanmaya çalıştık. Bunun amacı, ilerde karşılacağınız kullanılmaya hazır ve kullanma izni verilmiş Javascript kodlarını kendi sayfalarınıza uyumlu hale getirirken, program yazıcısının neyi nasıl ifade ettiğine aşina olmanıza yardımdır. Bu kitapçıkla giriş yaptığınız Javascript'i daha başka kaynaklardan derinlemesine öğrenmek istediğiniz zaman da, İngilizce terimler kaynaklar arasında paralellik kurmanıza yardımcı olacaktır.

Buraya kadar dersi iyi kaynattık. Haydi kolları sıvayalım.

Not: Bu kitapçıktaki örnek kodları PC World-Türkiye'nin Internet sitesinde (http://www.pcworld.com.tr) bulabilirsiniz. Bu kodları istediğiniz gibi kendi sitenizde kullanabilir ve dağıtabilirsiniz. Ancak bu kodların satılması yasaktır.

///////////////NOT İÇİN NOT://///////////////

Eğer bu kodlar Internet'te değil de CD'de yer alacaksa, not şöyle olabilir:

Not: Bu kitapçıktaki örnek kodları, CD-Magazin'in ---------- tarihli --------- sayısında bulabilirsiniz. Bu kodları istediğiniz gibi kendi sitenizde kullanabilir ve dağıtabilirsiniz. Ancak bu kodların satılması yasaktır.

////////////////////////////NOTUN SONU/////////////////

Javascript'le Tanışalım

Javascript programı yazmak için ihtiyacınız olan alet-edavat, bu kitapçığın yanı sıra, iyi bir bilgisayar ve bir düz yazı programıdır. Bilgisayarınız ecza dolabı kılıklı bir PC ise Notepad, bilgisayarınız mandalina, çilek veya şeftali renkli bir iMac ise SimpleText bu iş için biçilmiş kaftan sayılır. Eğer bu amaçla bir kelime-işlem programı kullanacaksanız, oluşturacağınız metnin dosyasını diske veya diskete kaydederken, düz yazı biçiminde kaydetmesini sağlamalısınız. Çalışmaya başlamadan önce örnek kodlarınızı bir arada tutabilmek ve gerektiğinde gerçek sayfalarınızda yararlanabilmek için sabit diskinizde bir dizin açmanız yerinde olur.

Javascript, HTML'in bir parçasıdır ve içinde bulunduğu HTML ile birlikte Web Browser programı tarafından yorumlanır. Dolayısıyla, Javascript programı yazmak demek, bir Web sayfası hazırlamak ve bu sayfadaki HTML kodlarının arasına Javascript kodları gömmek demektir.

O halde, kolları sıvayıp, ilk Javascript programımızı yazalım. Şu kodu yazıp, merhaba1.htm adıyla kaydedin:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Merhaba Dünya!</TITLE>
</HEAD>
<BODY>
<B>Merhaba Dünya</B><br>
<FORM>
<BUTTON onclick="alert('Merhaba Dünya!')">TIKLAYINI!</BUTTON>
</FORM>
</BODY>
```

```
</HTML>
```

Sonra ya bu dosyayı iki kere tıklayın; ya da favori Browser programınızı çalıştırın ve bu dosyayı açın. Karşınıza çıkacak düğmeyi tıklayın. İşte sonuç:

<JS001.TIF>

Şimdi kendi kendinizi kutlayın: Birazcık yardımla da olsa, Javascript programcısı olarak dünyaya merhaba demiş bulunuyorsunuz. Yazdığınız koddan tek kelime bile anlamıyorsanız da hiç önemli değil. Önemli olan ilk adımı atmaktı.

Bu adımı attığınıza göre şimdi yazdığınız kodu biraz irdeleyelim. Bütün Web sayfaları gibi, Javascript kodunuzun yer aldığı sayfa da tipik bir HTML kod kümesi:

1. Browser programına kendisinin bir HTML dosyası olduğunu birinci satırda beyan ediyor; ve bittiği yer açıkça gösteriliyor.

2. HTML kodu iki bölüm içeriyor: Başlık (Head) ve gövde (Body) bölümleri. Her bölümün nerede başladığı ve nerede bittiği açıkça belirtiliyor.

3. Gövde bölümünde bir Form unsuruna yer veriliyor; fakat bu formun tek ögesi var: Düğme (Button).

4. Daha önce Web sayfalarınıza düğme koyduysanız, bu düğmenin onlardan farklı bir tarafı var: Türü, değeri, vs. belirtilmiyor; sadece "onclick="alert('Merhaba Dünya!')" şeklinde bir ifadeye yer veriliyor.

5. Ve ilk Javascript programınızda ne Javascript'in adı geçiyor; ne de HTML ile Javascript bölümlerini birbirinden ayırteden, <SCRIPT>..</SCRIPT> etiketine yer veriliyor!

Özellikle bu son husus, size şunu gösteriyor: Günümüzde Netscape ve IE Javascript ile o kadar içiçe geçmiş ve Javascript özellikle 1.2'nci sürümü ile o kadar Browserların Belge Nesnesi Modeli (Document Object Model) ile kaynaşmıştır ki, kimi zaman HTML'in işlevi nerede bitiyor, Javascript'in işlevi nerede başlıyor, kolaylıkla ayırt edilemez.

Javascript'ten söz ederken, bu dilin imla kuralları olmakla birlikte bu kurallara yüzde 100 uymanın zorunlu olmadığını ifade ettik. Kural olarak, Javascript bölümü, HTML'in içine <SCRIPT>..</SCRIPT> etiketlerinin arasına gömülür. İşte size bütün kuralları yerine getirilmiş bir Javascript bölümü yazma örneği:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Merhaba Dünya!</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!--Javascript kodunu eski sürüm Browserlardan gizleyelim
function merhaba() //merhaba isimli fonksiyonu deklare ediyoruz
{ //bu, fonksiyonun başlama işareti
alert ("Merhaba Dünya!") //fonksiyonun komutu ve komutun gerektirdiği metin
} //bu fonksiyonun bitme işareti
// kod gizlemenin sonu -->
</HEAD>
<BODY>
<B>Merhaba Dünya</B><br>
<FORM>
<BUTTON onclick=merhaba()>TIKLAYINI!</BUTTON>
</FORM>
</BODY>
<HTML>
```

Bu metni merhaba2.htm adıyla kaydeder ve Browser'ınızda açarsanız, görüntünün ve işlevin tıpatıp aynı olduğunu göreceksiniz. İki metin arasındaki fark, ikincisinin Javascript yazma kurallarına yüzde 100 uymasından ibaret. Bununla birlikte Javascript ile HTML'i birbirinden dikkatlice ayırmanız gereken durumlar, ikisinin birbiriyle kaynaştığı noktalardan daha çoktur. Hatta o kadar ki, Javascript'in ileri sürümüne ilişkin komutlar kullanıyorsanız, eski sürüm Browser'ların kafasının karışmaması ve dolayısıyla ziyaretçinizin bilgisayarının kilitlenmemesi için bunu bile belirtmeniz gereken durumlar olabilir.

Bir de şunu deneyin: önce merhaba.js adıyla şu metni kaydedin:

```
function merhaba()
      {
alert("Merhaba, Dünya!")
      }
```

Sonra, merhaba2.htm dosyasında şu değişikliği yapın ve merhaba3.htm adıyla kaydedin:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Merhaba Dünya!</TITLE>
<SCRIPT SRC="merhaba.js" LANGUAGE=”JavaScript”>
</SCRIPT>
</HEAD>
<BODY>
<B>Merhaba Dünya</B><br>
<FORM>
<INPUT TYPE="SUBMIT" NAME="BUTTON1" VALUE="TIKLAYIN!" onclick="merhaba()">
</FORM>
</BODY>
</HTML>
```

Ve merhaba3.htm’i açtığınızda yine aynı görüntü ve aynı işlevle karşılaştığınızı göreceksiniz. Bu üçüncü yöntemde sadece Javascript kodlarını içeren ve dosya adının uzatması “.js” olan bir düzyazı dosyasını HTML’e dışardan da eklemiş olduk.

Bu esnekliğe bakarak, Javascript diline, “Ne yapsam olur! Nasıl yazsam işler!” anlayışıyla yaklaşamazsınız. Javascript, bütün bilgisayar dilleri gibi yorumlanabilmesi (çalışabilmesi) için kendi imla ve dilbilgisi kurallarına son derece bağlı kalınmasını ister. Bunu sınamak isterseniz, şu ana kadar oluşturduğunuz herhangi bir HTML dosyasında Javascript bölümündeki bir parantezi veya süslü parantezi kaldırın; dosyayı başka bir isimle kaydedin ve Browser’ınıza açtırmaya çalışın! Merhaba3.htm dosyasında “onclick=”merhaba()” komutunun bir parantezini kaldırdığımızda, Netscape ve IE’nin şu hata mesajlarını verdiğini görüyoruz:

<JS001.TIF>

HTML sayfalarınının Web Browser programlarında yapabileceği bir başka hata ise, kodu icra etmek yerine içerik gibi görüntülemektir. Web ziyaretçilerinin halâ kullandığı eski sürüm Browser programları, Javascript programlarını ya hiç anlamazlar, ya da eski sürümlerini anlarlar. Netscape 2.0 öncesi ile IE 3.0 öncesi Browser programlarının, Javascript kodlarını icra etmek yerine sayfanın içeriği imiş gibi görüntülemesini önlemek için Script bölümlerini eski sürüm Browser'lardan, merhaba2.htm sayfasında yaptığınız gibi gizlemeniz gerekir. Merhaba2.htm'e bakarsanız:

```
<!-- Javascript kodunu eski sürüm Browserlardan gizleyelim
function merhaba()
      {
alert ("Merhaba Dünya!")
      }
// kod gizlemenin sonu -->
```

şeklinde bir bölüm göreceksiniz. Bu bölümün başında ve sonunda yer alan "<!--" ve "-->" işaretlerinin arasındaki herşey, eski sürüm Browserlar tarafından HTML dili kurallarına göre "yorum" sayılacak ve görmezden gelinecektir. Javascript dilinin yorumları ise "//" işaretiyle başlar ve satır sonunda sona erer. Merhaba2.htm'de Javascript kodlarının bütün satırlarında böyle yorumlar koydunuz. Eğer yorumlarınız bir satırdan uzun olacaksa, bunu şöyle belirtebilirsiniz:

```
/* yorumun birinci satırı
yorumun ikinci satırı
yorumun üçüncü satırı */
```

Javascript veya başka bir dille program yazarken, iyi programcılar, programlarını kendilerinden başkası denetlemeyecek ve yeniden kullanmayacak da olsa, önemli işleri yaptıkları satırlara mutlaka yorum koyarlar. Bunun yararını, kendi yazdığınız bir programı bile bir yıl sonra yeniden açtığınızda görürsünüz!

Ziyaretçinin Web Browser programı Javascript anlıyor ise sayfanızdaki Javascript kodları, ya ilk yazdığınız programda olduğu gibi, ziyaretçinin sayfanızda bir yeri tıklaması veya klavyede bir tuşa basmasıyla harekete geçer; ya da HTML sayfası ziyaretçinin Browser'ında görüntülendiği anda otomatik olarak çalışmaya başlar. Otomatik çalışan Javascript kodu ise iki ayrı yöntemle çalıştırılabilir: HTML kodları icra edilmeden önce (yani sayfanız ziyaretçinin Web Browser'ında görüntülenmeden önce, veya sayfa görüntülendikten sonra.

Davranın düz yazı programınıza ve şu kodları girin:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Merhaba Dünya!</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!--
alert("Merhaba Dünya!")
// -->
</SCRIPT>
</HEAD>
<BODY>
<B>Merhaba Dünya</B><br>
</BODY>
<HTML>
```

Bu dosyayı da merhaba4.htm adıyla kaydedin ve Browser'ınızda açın. Sayfanızın içeriğini oluşturan "Merhaba Dünya" yazısı ile Javascript programınızın icra ettirdiği uyarı kutusunun görüntülenmelerindeki sıraya dikkat edin.

Bu dosyada, küçük bir değişiklik yapalım ve kodlarımıza şu şekli verelim:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Merhaba Dünya!</TITLE>
```

```
<SCRIPT LANGUAGE="Javascript1.2">
<!--
function merhaba()
      {
alert("Merhaba Dünya!")
      }
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="merhaba()">
<B>Merhaba Dünya</B><br>
</BODY>
<HTML>
```

merhaba5.htm adıyla kaydedeceğiniz bu dosyayı Browser'ınızda açtığınızda, sayfa metni ile uyarı kutusunun görüntülenme sırasına dikkat edin. Merhaba4'de önce uyarı kutusunun görüntülenmesinin sebebi, Browser'ın HTML kodundaki komutları yukarıdan aşağı sırayla yerine getirmesi ve bu arada karşısına çıkan Javascript komutunu da icra etmesidir. Oysa merhaba5'te, Javascript komutunu bir fonksiyon olarak tanımladınız ve bu fonksiyonun icrasını HTML'in BODY etiketinin içeriğinin Browser'ın penceresine yüklenmesine bağladınız. Bunu Javascript'in "onLoad" komutuyla yaptık. (onLoad'ın nasıl kullanıldığını ve nerelerde kullanıldığını daha sonra ayrıntılı ele alacağız. Şimdiden telaşa kapılmaya gerek yok!)

Buraya kadar gördüğümüz bir kaç örneği, Javascript kodlarının ne zaman ve nasıl işlediğini göstermek amacıyla verdik. Fakat bu örneklerden çıkartacağımız başka bir sonuç daha var: Dikkat etti iseniz Javascript komutları, daima HTML sayfasının bir unsuruna bağlı olarak veriliyor. Yani Javascript'in konusu veya hedefi, Web Browser programının (Netscape veya IE'nin) bilgisayarın ekranında oluşturduğu pencerenin içindeki belgenin bir ögesi. Bu öge, şimdilik ya bu belgenin içindeki FORM unsurunun bir bölümü (örneğin, düğmesi), ya da sayfanın BODY bölümünün bizzat kendisi oldu.

Bu ögeler, unsurlar, unsurların bölümlerine programcılık dilinde "Nesne" (Object) denir. Ya kendisi bir nesne oluşturabilen veya içinde çalıştığı programın ya da işletim sisteminin nesnelerini konu alan ve bu nesnenin unsurlarını, ögelerini, parçalarını hedef olarak alan dillere Object-oriented (nesne-yönelimli) diller denir. Bu dillerle yazılan programlara (isterse bir Script dilinin düz yazı şeklindeki komutları olsun) Object-oriented Program (OOP) adı verilir.

Javascript programları da OPP sınıfına girdiği için, Javascript programcısı olacaksanız, programlarınızın mutlaka Web Browser'ın bir nesnesinin bir özelliğini belirlemeye veya değiştirmeye ve Web Browser'ı bir nesne oluşturmaya ve nesnenin şu özelliğini şöyle veya böyle yapmaya yönelik olması gerektiğini akıldan hiç çıkartmamalısınız. Bu nesne, Browser'ın penceresinin menü çubuğunun olması ya da olmaması, pencerenin bilgisayar ekranında ne büyüklükte olması, pencerenin alt çerçevesindeki statü mesajının içeriğinin değiştirilmesi bile olabilir. Javascript programı açısından HTML etiketleri ile oluşturacağınız hemen herşey nesnedir. Bu nesnelerin nasıl etkileneceği ise Javascript programının içeriğini oluşturur.

Özetlersek, Javascript ile Web Browser programını penceresini ve Browser programının yorumladığı HTML etiketlerinin oluşturacağı sayfa ögelerini, bu ögelerin biçimini, değerini belirler, sorgulayabilir ve değiştirebilirsiniz.

Şimdi kısaca bu işlevselliğin temeli olan ve adına Document Object Model (DOM, Belge Nesne Modeli) denen kurallar demetinden söz edelim. Javascript kodu, DOM'u hedef alır: bu bağlamda DOM, bir Browser'ın penceresi ve penceresinin içindeki herşey demektir. Ama henüz herşey için bir olay ve metod standardı geliştirilmiş değil.

"Olay" (Event) ve "Metod" (Method) burada dikkat etmeniz gereken iki kelime. Olay, Web Browser'ın veya kullanıcının yaptığı bir iş, eylem, hareket demektir; "Metod" ise programcı olarak sizin bu "Olay"ı veya nesnenin bir özelliğini (Property) kullanarak,

Belge'nin bir unsuruna yaptırtabileceğiniz iş, eylem, hareket veya değişikliktir. Bu "teori" ilk bakışta sanıldığı kadar kavranması zor değil. Bir örnekle açıklayalım:

Diyelim ki, HTML sayfanızda şöyle bir bölüm var:

```
<FORM NAME=form01> WEB KİTAPÇISI
<INPUT TYPE="checkbox" NAME="roman">Roman
<INPUT TYPE="checkbox" NAME="hikaye">Hikaye
<INPUT TYPE="checkbox" NAME="biyografi">Biyografi
<TEXTAREA NAME="sonuc" ROWS="6" COLS="6">Burada sonuç gösterilecek</TEXTAREA>
</FORM>
```

Şimdi bu HTML kodu sayfanızda şu Nesne'leri oluşturur:

1. Bir adet Form

2. Dört adet "checkbox" türü INPUT alanı

3. Bir adet metin alanı

Şimdi dikkat: Sizin açınızdan "form01" isimli bu formu oluşturan FORM ögesi, Javascript açısından "document.form01" adlı nesnedir. Ziyaretçinin içine işaret koyduğu kutu, Javascript tarafından "document.checkboxSelected" diye tanınır. Sizin "sonuc" isimli metin alanınızın içinde yazılı olacak metin ise Javascript bakımından "document.form01.sonuc.value" (value=değer) diye bilinir.

HTML dilini geliştirenler, türü "checkbox" olan INPUT nesnesinin başına gelebilecek "olay" türlerinin neler olabileceğini düşünürken, "Mesela," demişler, "Kullanıcı bu kutuyu tıklayabilir!" Yani, bu nesnenin Browser açısından yol açabileceği "olay"lardan biri tıklanması halinde olan olaydır. Tıklamanın İngilizcesi ne? "click!" O halde bu olayın adı "Click," bu olayı yönlenrdiren metodun adı ise "onClick"tir (on, İngilizce -de, -da eki veya üzerinde, halinde kelimesinin karşılığıdır). Bu durumda "onClick," olayı, bu kutu açısından bir olaydır ve Javascript prnogramcısı olarak bu olayı yakalayabilir, kullanabiliriz. Başka bir deyişle, Browser'a, Javascript yoluyla "Ey Browser; kullanıcı benim INPUT nesnelerimden "roman"

adlı olanı tıkladığı zaman şu, şu işi yap!" emrini verebiliriz. Bu noktada bilgisayar programının, hangi dille yazılırsa yazılsın, alt alta gelmiş komutlar listesi olduğunu hatırlatalım.

Peki, yine aynı örnekte, Browser'a Javascript yoluyla ne gibi bir iş yaptırtabiliriz? Mesela, "Ey Browser, kullanıcı benim INPUT nesnelerimden 'roman' adlı olanı tıkladığı zaman benim 'sonuc' adlı TEXTAREA nesnemin değerini 'Roman' yap!" diyebiliriz. Biliyorsunuz ki, TEXTAREA nesnelerinin değeri, ekranda çizdikleri kutunun içine koydukları içerik demektir. Tabiî aynı mantıkla, "hikaye" kutusu işaretlenince Sonuç kutusuna "Hikaye," "biyografi" kutusu işaretlenince de Sonuç kutusuna "Biyografi" yazdırtmak elimizde.

Kısaca, Javascript herşeyi, bir HTML ögesine göre tanımlar, adlandırır ve bilir. Unutmayacağınız kural: "Javascript, sayfanın bir şeyini alır, sayfanın bir şeyine hitabeder!"

Bu gayet anlamlı şekilde ifade ettiğimiz kuralı yerine getirmekte sorun, sadece Javascript'in on-onbeş kelimesini, yedi-sekiz işlemini öğrenmek değil, aslında HTML belgesinin nesnelerinin olaylarını ve metodlarını öğrenmektir.

Dolayısıyla, önce Javascript'in on-onbeş kelimesi ve yedi-sekiz işleminden ibaret temel kurallarını ele alacağız; sonra da bunlarla Belge Nesneleri'ne hangi olay halinde ne yapabileceğimizi ve bunu nasıl yapabileceğimizi göreceğiz.

Javascript'in Temel İlkeleri

Web Browser programları, Javascript komutlarını yorumlayabilmek için, HTML'in içinde <SCRIPT LANGUAGE="Javascript"> etiketini ararlar ve </SCRIPT> etiketini gördükleri anda bu yoruma son verir; HTML yorumuna dönerler. Bu etikette, kullanacağınız Javascript sürümünü de belirtebilirsiniz. Javascript, şu anda 1.3'ncü sürümünde. Ancak Netscape 2.0 ve 2.2 ile IE 3.0, Javascript'in 1.0 ve 1.1 sürümlerini tanıyabilir. Her iki Browser'ın 4'ncü sürümleri ise Javascript 1.2 ile uyumludur. Script etiketinde sürüm yazmazsanız, her iki Browser da 1.0 sürümünü kullandığınızı varsayar. Netscape ileri sürümlere ait komutları (eğer tanıyorsa) icra eder, tanımıyorsa görmezden gelir. IE ise belirttiğiniz sürümden ileri komutları tanımaz ve hata mesajı verir. Kullanılan Browser türleri ve sürümlerine ilişkin istatistikler dikkate alınırsa, bu satırları kaleme aldığımızda en güvenli sürüm 1.2'dir. Biz de örneklerimizde bunu belirteceğiz.

Hemen bir uyarı Javascript'in birinci sürümünü kastetmek amacıyla <SCRIPT LANGUAGE="Javascript1.0"> veya "<SCRIPT LANGUAGE="Javascript1"> yazmayın; IE hata mesajı verir, Netscape ise Javascript bölümünü görmezden gelir!

Javascript Yazım Kuralları

Browser'ların Javascript yorumlama bölümleri, kodların arasında yer alan yorum ifadeleri ile boşlukları atar; geri kalan kelimeleri beşe ayırarak ele alırlar:

1. Belirleyiciler (Identifier):

Javascript dilinin değişkenleri, metodları ve nesnelerini belirleyen isimlere Belirleyiciler denir. Bu sınıfa giren bütün kelimeler ya harfle ya da alt çizgi (_) ile başlar. Rakam veya diğer işaretler birinci karakter olarak kullanılamaz, fakat daha sonra

kullanılabilir. Javascript, aynı kelimenin büyük harfle yazılanı ile küçük harfle yazılanını farklı isimler olarak görür. Bu sınıfta giren kelimelerin içinde boşluk olamaz.

Javascript kodlarınız sizin bilgisayarınızda değil, ziyaretçinin bilgisayarında çalıştırılacağına göre, kullandığınız karakterlerin ziyaretçinin bilgisayarında nasıl bir değer taşıyacağını düşünmeniz gerekir. Bu bakımdan güvenli yol, Bu sınıfa giren kelimelerde, İngilizce alfabede bulunmayan, Türkçe ve diğer dillerdeki high-ASCII karakterleri (ı, İ, ğ, Ğ, Ş, Ş ile ü, Ü, ö, Ö, ç ve Ç) kullanmamaktır.

Aşağıda doğru ve yanlış belirleyici kelime örneklerini bulacaksınız:

Doğru	Yanlış
sonucgoster	sonuc goster
ikincidegisken	2ncidegisken
_gelen	#gelen

Anahtar kelime grubundaki kelimeler de bu sınıfta kullanılamaz.

2. Anahtar Kelimeler (Keyword):

Javascript dilinin önceden tanımlanmış ve programın yorumunda özel anlam kazandırılmış kelimelerine Anahtar Kelime denilir. Aşağıda bu kelimelerin yanında kelime anlamlarını göreceksiniz; fakat ilerledikçe bu kelimelerin Javascript'te kullanıldıkları yere göre anlamlarını ve nerelerde kullanıldıklarını göreceğiz.

//////////QUARK İÇİN NOT/////////////////

Kelime listelerini Word içinde sütun haline getirdim. Böylece Quark'a giderse alfabetik sıra bozulabilir. Word'de iken kelime listesini sütun'dan çıkartmak gerekir.

////////////////QUARK NOTU BİTTİ////////////////////

Javascript 1.0'deki anahtar kelimeler:

break (kes)	continue (devam et)

else (başka bir durum)

false (yanlış)

for (için)

function (işlev)

if (eğer)

in (içinde)

int (integer, tam sayı)

new (yeni)

null (boş değer)

return (dön)

this (bu)

true (doğru)

var (variable, değişken)

while (... iken)

with (ile)

Javascript 1.i ile eklenen anahtar kelimeler:

typeof (türü)

void (geçersiz)

Javascript 1.2 ile eklenen anahtar kelimeler:

do (yap)

labeled (etiketli)

switch (değiştir)

3. Ayrılmış Kelimeler (Reserved):

İkinci gruba girsin-girmesin bazı kelimeler, ilerde Javascript programlama ve yorumlama işlerinde kullanılabileceği düşüncesi ile, bir kenara ayrılmıştır; Javascript kodlarında kullanılamazlar. Bu listede yer alan ve halen Anahtar Kelime listesine girmiş bir kelime değişken, fonksiyon, nesne veya metod adı olarak kullanılırsa, program hata verir; henüz anahtar kelime listesine alınmamış olmakla birlikte "rezerv edilmiş" bu kelimelerin geçtiği komut satırı ise görmezden gelinir. Javascript programlarınızda kullanmayacağınız kelimelerin listesi şöyledir:

abstract (soyut)

boolean (Boolean Mantığı)

break (kes)

byte (bayt)

case (hal)

catch (yakala)

char (karakter)

class (sınıf)

const (sabit)

continue (devam)

default (varsayılan)

delete (sil)

do (yap)

double (çift)

else (başka bir durum)

extends (uzanır)

false (yanlış)

final (sonuncu)

finally (sonunda)

float (kesirli)

for (için)

function (işlev)

goto (--ya git)

if (eğer)

implements (uygular)

import (ithal et)

in (içinde)

instanceof (--nın oluşumu)

int (integer, tam sayı)

interface (arayüz)

labed (etiketli)

long (uzun)

native (kendinden olan)

new (yeni)

null (boş değer)

package (paket)

private (özel)

protected (korunmuş)

public (genel)

return (dön)

short (kısa)

static (sabit)

super (kuvvet)	true (doğru)
switch (değiştir)	try (dene)
synchronized (uyumlu)	typeof (türü)
this (bu)	var (değişken)
throw (içine kat)	void (geçersiz)
throws (içine katar)	while (iken)
transient (geçici)	with (ile)

4. Değerler (Literal):

Javascript kodu icra edildiği sırada değişmeyen rakam veya metinlere Değer denir. Javascript kodlarında beş tür değer bulunur:

a. Tamsayı Değerler (Integer Literal):

Tamsayılar, 10 tabanlı (ondalık, decimal), 8 tabanlı (octal) veya 16 tabanlı (hexadecimal) olabilir. 8 tabanlı sayıları belli etmek için sayıdan önce sıfır, 16 tabanlı sayıları belli etmek için sıfır ve x harfi kullanılır. Hexadecimal sayılarda 9'dan büyük sayılar A, B, C, D, E ve F harfleri ile yazılır. (Sayının basamaklarını okuma kolaylığı sağlamak için, nokta değil, virgül ile ayırmalısınız.) Örnek: Decimal 46,789; Octal 072,7898; Hexadecimal: 0x7B8.

b. Kesirli Değerler (Floating-point literal):

Tam sayı bölümünü bir ondalık nokta (virgül değil) ile kesir bölümü izleyen sayılar. Örnek: 3987.786, -1.1.

c. Boolean Mantık İfadeleri (Boolean Literal):

Javascript dilinde, Boolean Mantığı, iki sonuç verir: True (Doğru) ve False (Yanlış). Javascript, True (Doğru) değerini 1, False (Yanlış) değerini 0 rakamıyla tutar. Bir denklemin sonucunun doğru veya yanlış olduğunu irdelerken, Javascript metninde bu iki kelime küçük

harfle yazılmalıdır. Dolayısıyla büyük harfle yazılmış TRUE ve FALSE kelimeleri, değişken, fonksiyon, nesne ve metod adı olarak kullanılabilir. Tabii, bir süre sonra kafanız karışmazsa!

d. Alfanümerik (Karakter) Değerler (String literal):

İki adet çift-tırnak (") veya tek-tırnak (') içine alınan her türlü ifade, Javascript için String değeridir. (Çeşitli Türkçe bilgisayar kaynaklarında "String literal" terimi "karakter değişken" olarak belirtilmektedir. "Karakter" bu değerlerin örneğin sayı değerlerden farkını anlatmıyor. Bazı kaynaklarda ise hem rakam, hem de harf içerdikleri, buna karşılık sayı olmadıkları gerçeğini belirtmek amacıyla, daha aşina olduğumuz alfanümerik terimini görmek mümkün. Ben bu ikincisini benimsiyorum.) Bir Javascript metninde alfanümerik değerleri bir tek tırnakla, bir çift tırnakla gösteremezsiniz. Başta nasıl başladıysanız, programın sonuna kadar bütün alfanümerik değerleri aynı tür tırnak içinde göstermelisiniz. Alfanümerik değer olarak verdiğiniz karakterler, daha sonra bir HTML sayfada bir nesnenin bir unsurunun değeri (örneğin bir TEXTAREA'nın içeriği) olarak kullanılacaksa ve HTML sayfa, meta etiketlerinde kendisinin Türkçe olarak yorumlanmasını sağlayacak ifadeye sahipse, Türkçe karakter ve diğer yüksek ASCII kodlardaki karakterleri içerebilir. Örnekler: "Bugün hava çok güzel" 'Bugün hava güzel değil' "123 adet yumurta" '23,234,678.987'

e. Özel Karakterler

Özellikle alfanümerik değerleri verirken, Browser'a metin görüntüleme konusunda biçim veya hareket komutları da vermek isteyebilirsiniz. Bunu, bazı kod harflerin önüne ters-bölü işareti koyarak yapabilirsiniz. Bu tür özel karakterler şunlardır:

\b - Klavyede Geri (backspace) tuşunun görevini yaptırır.

\f - Yazıcıya sayfayı bitirmeden çıkarttırır (formfeed).

\n - Yazı imlecini yeni bir satırın başına getirir (new line)

\r - Klavyede Enter-Return tuşunun görevini yaptırır.

\t - Sekme (tab) işaretini koydurur.

\\ - Yazıya ters-bölü işareti koydurur.

\’ - Yazıya tek-tırnak işareti koydurur.

\” - Yazıya çift-tırnak işareti koydurur.

Javascript’e bu tür özel karakterlerle HTML sayfasına bir metin yazdıracağınız zaman, bu yazının <PRE>..</PRE> etiketleri arasında olması gerekir. Aksi taktirde Javascript ne yazdırırsa yazdırsın, HTML bu özel karakterleri dikkate almayacaktır.

Javascript Değişkenleri (Variable)

Değişken, adı üstünde, Javascript yorumlayıcısı tarafından bilgisayarın belleğinde tutulan ve içerdiği değer programın akışına göre değişen bir unsurdur. Değişkenlerin bir adı olur, bir de değeri. Program boyunca beğişkenin adı değişmez; fakat içeriği değişebilir.

Değişkenlere isim verirken Belirleyici isimleri kurallarına riayet etmeniz gerekir. Yani bir değişkenin adı rakamla veya alt çizgi (_) dışında bir işaretle başlayamaz. Javascript, büyük harf-küçük harf ayırt ettiği (case-sensitive olduğu) için, örneğin SONUC ve sonuc kelimeleri iki ayrı değişken gösterir. Bir değişkeni tanımlarken, büyük harf kullandıysanız, program boyunca bu değişkeni büyük harle yazmanız gerekir.

Değişken tanımlamak, bilgisayar programcılarına daima gereksiz bir yük gibi görünür. Birazdan göreceğiz, Javascript sadece değişkenleri tanımlamayı zorunlu kılmakla kalmaz, fakat nerede tanımlandığına da özel bir önem verir. Javascript’e bir Belirleyici’nin değişken olarak kullanılacağını bildirmek için “var” anahtar-kelimesini kullanırsınız:

```
var sonuc

var adi, soyadi, adres, siraNo

var i, j, k
```

var mouseNerede, kutuBos, kutuDolu

Gördüğünüz gibi, bir "var" satırında birden fazla değişken tanımlayabilirsiniz.

Bazı diğer programlardan farklı olarak Javascript, size değişkenleri hem beyan, hem de içeriğini belirleme işini aynı anda yapma imkanı veriyor (initialization):

var sonuc = "Merhaba Dünya!"

var adi = "Abdullah", soyadi = "Aksak"

var i = 100, j = 0.01, k = 135

var kutuBos = false, kutuDolu = true

Gerekiyorsa, bir değişkeni önce tanımlar, sonra değerini belirleyebilirsiniz:

var kutuBos = false, kutuDolu

kutuDolu = true

Javascript, programcıya bir değişkeni tanımlamadan "initalize etme" (içine doldurma) imkanı da verir; bu durumda o değişken genel (global) nitelik kazanır. Buna birazdan değineceğiz; ama iyi programlama tekniği ve özellikle daha sonra hata arama zorunluğu sebebiyle, değişkenlerin açık-seçik tanımlanması en doğru yoldur.

Bir değişkeni tanımlayarak içini doldurmadan (initialization'dan) önce içindeki değeri sorgulamaya kalkarsanız, Browser'ın Javascript yorumlayıcısı o noktada durur ve tanımlanmamamış (undefined) değişken hatası verir.

Javascript programlarında beş tür değişken bulunabilir:

1. Sayı (number): -14, 78, 87678

Sayı türündeki değişkenler, tam sayı, ondalık sayı, pozitif sayı veya negatif sayı olabilir. Sayı değişkenlerle aritmetik işlemler yapılabilir.

```
var eni = 9, boyu = 4
```

```
var alani = eni * boyu
document.writeln(alani)
```

Bu kod örneği ile Javascript, Browser penceresi içindeki belgeye "alani" adlı değişkenin değerini (36) yazdıracaktır.

2. Boolean değişken: true, false

Javascript, tanımlanırken değeri "true" (doğru) veya "false" (yanlış) olarak belirtilen değişkenleri otomatik olarak Boolean değişken olarak sınıflandırır ve sorguladığınızda "true" için 1, "false" için 0 değerini verir.

3. Alfanümerik (String) Değişken: "Merhaba Dünya!"

Alfanümerik değişken, kendisine tek veya çift tırnak içinde verilen bütün karakterleri tutar, ve sorulduğunda aynen bildirir. Alfanümerik değişkenin içi boş olabilir (var adi = "", soyadi = ""). Alfanümerik değişkenler, tabir yerinde ise "toplandığında" değişkenlerin değerleri sırayla birbirine eklenir:

var adi = "Osman", soyadi = "Hömek"

var adisoyadi = adi + soyadi

document.writeln(adisoyadi)

Bu kod örneği ile Javascript, Browser penceresi içindeki belgeye "adisoyadi" adlı değişkenin değerini (OsmanHomek) yazdıracaktır. Araya boşluk koymanın tekniğine ilerde değineceğiz!

4. İşlev (Function) Değişken:

Javascript'in hayatî noktası fonksiyonlardır. Javascript'e fonksiyonlarla iş yaptırırız. Kimi fonksiyonu, Javascript'i tasarlayanlar bizim için tanımlamışlardır; bunlara metod denir. (Kimi metod, ne yapacaksa bizim hiç bir katkımızı beklemeden yapar; kimi metod mutlaka bizden bir katkı bekler.) Kimi fonksiyonları biz tanımlarız ve komutlarını biz veririz. Bunlara "Programlanan Fonksiyonlar" denir. Başlarda yazdığınız ve merhaba2.htm adıyla

kaydettiğiniz dosyada biz "merhaba()" adlı bir fonksiyon oluşturduk; sonra bu fonksiyona bir otomatik fonksiyon olan alert() metodunu kullanmasını bildirdik; bu otomatik fonksiyona da görevini yaparken kullanmasını istediğimiz değeri verdik!

5. Nesne (Object) değişkenleri: window, document

Bu tür değişkenlere değişken adını vermek bile gerekmez; çünkü bunlar Browser'ın nesneleridir. Fakat Javascript kodlarımızda bu nesneleri de değişken gibi kullanabiliriz. Bu sınıfa giren özel bir değişken ise değerini "null" (içi boş) kelimesiyle belirlediğiniz değişkenlerdir.

Javascript, değişkenlerini bu beş sınıfa ayırmakla birlikte sizden değişkenlerinizini sınıflamanızı beklemez. Sınıflamanın önemi, daha sonra, programın ileri bir aşamasında bir değişkenin değeri ile yaptıracağınız işlemde ortaya çıkacaktır. Bir değişken, kendi sınıfının yapamayacağı bir işleme veya sorgulamaya tabi tutulursa, Javascript size pek de kibarca olmayan bir hata mesajı ile değişkeni doğru kullanmadığınızı hatırlatacaktır.

Bununla birlikte bir fonksiyonda sayı olarak tanımlanmış ve doldurulmuş bir değişken, aynı fonksiyonda daha sonraki bir işlemde veya başka bir fonksiyonda alfanümerik değişken olarak tanımlanabilir ve doldurulabilir. Bu değişken sorgulandığında değerini, yetki alanı (scope) çerçevesinde bildirir.

Değişkenlerin yetki alanı veya geçerli olduğu alan (scope), oluşturulmuş ve değeri belirlenmiş olduğu sahayı, yani kendisine atıfta bulunulduğu zaman bildireceği değerini ve bu değişkene nerelerden atıfta bulunulabileceğini gösterir. Şimdi, HTML dosyasının baş tarafında (HEAD etiketi içinde) bir değişkeni tanımladığınızı ve ona bir değer verdiğinizi düşünün. Daha sonra yazacağınız bütün fonksiyonlarda veya değerini belirleyebileceğiniz otomatik fonksiyonlarda (metod'larda) bu değişkeni bir daha tanımlamaya ve değerini belirlemeye gerek kalmadan kullanabilirsiniz; çünkü Javascript açısından bu değişken genel (global) değişken sayılır. Daha sonra ayrıntılı olarak göreceğimiz gibi, HTML'in gövde

kısmında (BODY etiketi içinde) bir fonksiyon yazdığımızı ve bu fonksiyonun içinde bir değişken tanımladığımızı düşünün. Daha sonra yazacağınız bir fonksiyonda bu değişkeni kullanamazsınız; çünkü Javascript bir fonksiyonun içindeki değişkeni yerel (local) değişken sayar ve kendi yetki alanı (scope'u) dışında kullanmanıza izin vermez. Bir yerel değişken, ait olduğu fonksiyon çağrıldığı anda oluşturulur ve fonksiyonun icrası bittiği anda yok edilir. Dolayısıyla bir fonksiyon, başka bir fonksiyonun yerel değişkinini kullanmaya kalktığında "undefined" (tanımlanmamış) değişken hatasıyla karşılaşırsınız.

Bir değişkeni bütün HTML sayfası boyunca kullanmayı düşünüyorsanız, bu değişkeni, HTML sayfasının başında içinde fonksiyon bulunmayan bir SCRIPT bölümünde tanımlamalı ve doldurmalısınız. İyi bir programlama tekniği, genel değişkenleri, açık-seçik tanımlamak ve bunu yorum satırıyla belirtmektir:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Merhaba Dünya!</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// genel değişken tanımları
var metin1 = "Merhaba Dünya!"
var metin2 = "Vayy, merhaba canım! N'aber?"
var metin3 = "Bu samimiyet nereden? Lütfen biraz ciddi olur musunuz?"
// -->
</SCRIPT>
</HEAD>
<P>BURAYA ÇEŞİTLİ AMAÇLARLA HTML KODLARI GİREBİLİR!! SONRA TEKRAR SIRA JAVASCRIPT'E
GELEBİLİR</P>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
//bu fonksiyon uyarı kutusu açar
function merhaba()
      {
```

```
alert(metin1)
      }
// -->
</SCRIPT>
</HEAD>
<BODY>
<B>Merhaba Dünya</B><br>
<FORM>
<INPUT TYPE="SUBMIT" NAME="dugme1" VALUE="TIKLAYIN!" onclick="merhaba()">
</FORM>
</BODY>
</HTML>
```

Bu kodu da merhaba6.htm adıyla kaydedebilirsiniz. Daha sonra merhaba fonksiyonunda "alert(metin1)" satırındaki 1 rakamını 2 ve 3 diye değiştirirseniz, merhaba6.htm'in uyarı kutusunun içeriğini değiştirdiğini göreceksiniz. Şimdi, merhaba6'da merhaba() fonksiyonunu şöyle değiştirin ve şu eki yapın:

```
function merhaba()
      {
var metin1 = "Sana da merhaba!"
alert(metin1)
      }
function hello()
      {
alert(metin1)
      }
```

Sonra, aynı dosyada mevcut düğme oluşturan INPUT satırının altına şu satırı ekleyin; sayfanızda iki düğme görünecektir:

```
<INPUT TYPE="SUBMIT" NAME="dugme2" VALUE="TIKLAYIN!" onclick="hello()">
```

Bu düğmelerden birincisi, merhaba() fonksiyonunu, ikincisi hello() fonksiyonunu çağıracak. Genel olarak tanımladığınız ve değer verdiğiniz metin1 değişkeninin değerini merhaba() fonksiyonunda değiştirdiniz; bu değişkenin yeni değeri "Sana da merhaba!" şeklini aldı. Bu metni, merhaba7.htm adıyla kaydedip, Browser'da açtığınızda ve birinci

düğmeyi tıkladığınızda, uyarı kutusunun içeriğinin metin1 değişkeninin genel değerini değil yerel değerini içerdiğini göreceksiniz. Yani metin1'in değeri değişmiş oldu. Peki, ikinci düğmeyi tıklayın! metin1'in değeri değişmemiş gibi, uyarı kutusunun genel tanımlamadaki değeri kullandığını göreceksiniz. Oysa biraz önce, birinci düğmeyi tıkladığımızda yani merhaba() fonksiyonu çalıştığında metin1'in değeri değişmemiş miydi? Şimdi neden eski metin1'in eski değeri karşımıza çıkıyor?

Bu alıştırmanın öğrettiği kural şöyle özetlenebilir: Bir genel değişken, bir fonksiyon içinde yerel olarak değiştirilebilir; ama onun genel değeri diğer fonksiyonlar için geçerli kalır. Javascript programlarınızı yazdığınız zaman genel değişkenleriniz beklediğiniz değeri vermiyorsa bu değeri bir fonksiyonun yerel olarak, sırf kendisi için, değiştirip değiştirmediğine bakmalısınız. Bu tür hatalardan kaçınmanın bir yolu, yerel değişkenlerinizle genel değişkenlerinize farklı isimler vermektir.

Javascript, değişkenlerinizi isimlendirmede rakamla ve işaretle başlamamak dışında kural koymuyor; ama iyi programcılık tekniği, değişkenlere anlaşılır ve kullanıldığı yeni belli eden isimler vermektir. Örneğin, "adi" şeklindeki bir değişken adı, çok değişkenli bir Javascript programında yanıltıcı olur. "musteri_adi" veya "musteriAdi" çok daha uygun olabilir. Ayrıca değeri "Evet" (=doğru, true) veya "Hayır" (=yanlış, false) olan Boolean değişkenlerin adlandırılmasında, örneğin, "doluMu," "tamamMi," "acikMi," "bittiMi" gibi cevabı çağrıştıran değişken adları kullanabilirsiniz.

Dizi-Değişkenler (Array)

Javascript'in şu ana kadar ele aldığımız bütün değişkenleri, tekildir. Yani bu değişkenlerden Javascript bakımından sadece bir adet vardır; dolayısıyla bir adet de değeri bulunur.

Oysa şimdi şöyle bir durum düşünün: Evcil kuşları tanıttığınız bir site yapıyorsunuz. Ele alacağınız bütün kuşları ve özelliklerini bir tablo olarak düşünebilir misiniz? Kuş-1:

Kanarya, Kuş-2: Bülbül, Kuş-3: Muhabbet Kuş-4: Papağan, gibi. Bunların hepsini tek tek tanımlamaya kalkarsak, daha sonra kullanım zorluğu doğabilirdi. Javascript, bize "dizi-değişken" (Array) yoluyla, örneğin bir "kuş" değişkeni oluşturup, bunun içine istediğimiz kadar değer koyma imkanı veriyor. Bir dizi-değişkenin bireylerinin değerlerini dizi-değişkenin endeksi ile çağırabilir veya belirleyebilirsiniz.

Bir dizi-değişken oluşturmanın formülü şudur:

var dizinin_Adı = new Array(unsur1, unsur2, unsur3...unsurN)

Bu formüle göre, kuş dizisini şöyle oluşturabiliriz:

var kusDizi = new Array(bülbül, kanarya, muhabbet, papagan)

Diziler (nedendir, bilinmez) sıfırdan itibaren numaralanır; yani bizim örneğimizde dizinin birinci üyesi kusDizi[0], ikinci üyesi kusDizi[1], üçüncü üyesi kusDizi[2], dördüncü üyesi ise kusDizi[3] diye anılır. Burada örneğin kusDizi[2] değişkeni "muhabbet" değerini taşır. İlerde bu değeri değiştirmek ve diyelim ki "Saka" yapmak istersek, Javascript kodumuzun uygun yerinde şunu yazarız:

kusDizi[2]="saka"

Dizi-değişkenleri, özellikle ele almamızın bir nedeni, dizilerin Javascript açısından nesne sayılmasıdır. Bunun önemine ve kullanıldığı yerlere daha sonra değineceğiz; şimdilik şunu belirtelim: dizi-değişkenlerin, nesne oldukları için özellikleri (property) vardır. Örneğin, oluşturduğum dizi-değişkenin kaç üyesi olduğu, "kusDizi.length" özelliğinin içinde yazılıdır. İlerde dizi-değişkenlerimizin üye sayısından çok yararlanacağız.

Javascript'te İşlemler (Operator)

Program yazmanın amacı, bilgisayara işlem yaptırtmaktır. Javascript de bu açıdan farklı değildir: bir değişkenin değerini alırsınız, bir işlemden geçirirsiniz; sonra ortaya çıkacak yeni değeri ya bildirirsiniz, ya da yeni bir işlemde kullanırsınız. Tabiî herşey bu kadar kolay

olsa idi, herkes programcı olurdu. Herkes programcı olmadığına göre, söz gelimi "bir değişkeni almak" için önce değişkenin değerinin nereden ve nasıl elde edileceğini bulmanız gerekir. Ayrıca yukarıda öğrendik ki, Javascript programları Browser'ın pencere nesnesine veya penceresinde de görüntülediği belge nesnesinin bir unsuruna hitaebeder. Dolayısıyla hangi nesnenin ne gibi nitelikleri olduğunu öğrenmemiz gerekir ki, işleme tabi tuttuğumuz değerleri bu unsurların işine yarayacak şekle getirelim.

Javascript, eline verdiğiniz değişkenin türüne ve değişkenin değerine göre dört tür işlem yapar. Şimdi bunları sırasıyla ele alalım:

Atama (Assignment) İşlemleri:

Javascript'te en sık yapacağınız işlem, bir değişkene bir değer atama işlemidir: "Adı" adlı değişkenin değeri "Hakkı" olsun! "Soyadı" adlı değişkenin değeri "Öcal" olsun, gibi. Ve tahmin ettiğiniz gibi bunu eşittir (=) işaretiyle yaparız:

var adi = "Ahmet", soyadi = "Arif"

var siraNo = 123, sigortaNo = "A123-2345-234"

Javascript, bir eşittir işaretiyle karşılaştığında, önce işaretin sağına bakar ve değeri okur; sonra işaretin soluna bakar ve bu adın, meşru bir değişkene ait olup olmadığını ve bu adla bilgisayarın belleğinde bir yer açıp açamayacağını belirler. Eğer işaretin solundaki isim, meşru, yani önceden tanımlanmış veya değeri belirlenmiş bir değişkene ait ise, o değişkenin bilgisayarın belleğindeki adresine işaretin sağındaki değeri yazar. Değişken ismi yeni bir meşru isme aitse, Javascript bu değişkeni "initalize" eder, yani hem yer açar, hem de içine değerini yazar. Değişken ismi kabul edilebilir nitelikte değilse, yani söz gelimi altı çizgi (_) dışında bir işaretle veya rakamla başlıyorsa, içinde boşluk varsa, veya yasaklı kelimelerden biri ise, Javascript hata mesajı verir.

Bu noktada sık yapılan bir hata, başka bir programlama dilinden kalma alışkanlıkla değişkeni sağa, değeri sola yazmaktır: hakki = adi, gibi. Javascript bu durumda da hata

mesajı verir; çünkü "hakki" değişkenine "adi" değerini atamak için, sayı olmayan bu alfanümerik değerin ya tek, ya da çift tırnak içine yazılması gerekir.

Javascript'te bir değişkene değer atarken, bu değeri mevcut bir veya daha fazla değişkenden de alabilirsiniz:

```
var i = j + k

var indexNo = siraNo + kategoriNo

var tutariTL = birimFiyatTL * adet
```

Aritmetik İşlemleri:

İşte lisede matematik dersinde kaytarmamış olduğunuza şükredeceğiniz noktaya geldik: Javascript (veya başka herhangi bir programlama dili) siz formülünü vermedikçe hiç bir hesabı yapamaz; ama dünyanın en çapraşık, en karmaşık formülünü de verseniz, sonucu (bilgisayarınızın hızı ile orantılı olarak) hesaplar.

Tabiî, Javascript, dört temel işlemi yapabilir. toplama işlemini artı işaretiyle (+), çıkartma işlemini eksi işaretiyle (-), çarpma işlemini yıldız (asterisk, *) işaretiyle, ve bölme işlemini düz bölü işaretiyle (/) yaptırırsınız.

Javascript, artı işaretini gördüğü zaman, işaretin hem sağına, hem de soluna aynı anda bakar ve bulduğu iki değeri, sayı ise toplar, alfanümerik ise sağdakinin soldakinin arkasına ekler:

```
x = 3 + 4
```

Javascript, bu işlemin sonucu olarak x adlı değişkene 7 değerini atar. Fakat kimi zaman, bir değişkenin mevcut değerini (sıfır bile olsa) belirli bir miktarda arttırmak isteyebilirsiniz. Yani Javascript'e "x'in mevcut değerine 3 ekle!" demek isteyebilirsiniz. Bunu, "x = x + 3" diye ifade ederiz. Eğer arttırılacak miktar 1 ise, bunu daha kısa yazmak da mümkündür: ++x.

Javascript çıkartma işleminde de aynı kuralları uygular:

x = 3 - 4

x = x - 3

--x

Değişkenleri 1 arttırma veya 1 eksiltmeye yarayan kısaltmanın üzerinde biraz duralım. İlerde bol bol örneklerini göreceğimiz gibi, Javascript programlarınızda bazı değişkenleri sayaç olarak kullanmak isteyebilirsiniz. Bir işin sonuna geldiğimizi bilebilmek için, diyelim ki 12 kişilik bir adres listesini Javascript'e okutup, sayfaya bir tablo çizdiriyorsak, Javascript'e listenin sona geldiğini belirtebilmek için sayaç rakamı 12 veya 0 oldu ise durmasını emretmek zorundasınız. Bu durumda sayaç olarak kullandığınız değişkenin değerini, her adres okunduğunda 1 arttırmak veya 1 eksiltmek, en kestirme yoldur. Diyelim ki, x'in orijinal değeri 12. Javascript x-- veya --x işlemini gördüğü anda, 12'den 1 çıkartacak ve x'in yeni değeri olarak 11'i atayacaktır. Aynı şekilde ilk değer sıfır ise, x++ veya ++x işlemini gördüğü anda x'in yeni değerini 1 yapacaktır.

Toplama ve çıkartma işlemlerinde yapabileceğiniz başka bir kısaltma ise şöyle yazılır:

x = x + y işlemini kısaltmak için x += y

x = x - y işlemini kısaltmak için x -= y

Bu bölümü bitirmeden önce, bir sayının değerini negatif yapmaktan da söz edelim. Bir değişkene negatif değer atamak için, sayının önüne eksi işareti koymanız yeter: x = -5 gibi. Bir değişkene başka bir değişkenin değerini negatif olarak atamak istiyorsanız, x = -y gibi yazabilirsiniz. y değişkeninin değeri zaten negatif ise, x'e atanan değer pozitif olur.

Belki belirtmek bile gerekmez ama, Javascript, bölü işaretinin solundaki sayıyı, sağındaki sayıya böler; yıldız (asterisk) işaretinin sağındaki ve solundaki sayıları birbiri ile çarpar.

Javascript, alfanümerik değerlerle çıkartma, çarpma ve bölme işlemleri yapmaz; sonucun yerine "NaN" (Not a Number, Sayı Değil) yazar.

Karşılaştırma İşlemleri:

Javascript dahil, bütün bilgisayar programlarının ortak özelliği, programın bir noktada karar vermesidir: "Filanca değişken şu değerde ise, falanca değişken ile şunu yap; o değerde değil de, bu değerde ise bunu yap!" gibi. Bu tür komutlarda Javascript, sözünü ettiğiniz değişkenin değerini bulup, onu verdiğiniz bir başka ölçütle karşılaştıracak ve varacağı sonuca göre emrettiğiniz işi yapacaktır. Dolayısıyla Javascript'te bir takım karşılaştırma işlemlerine ihtiyacınız var demektir.

Javascript'in karşılaştırma operatörleri genellikle "if" (eğer..ise) ifadesiyle birlikte kullanılır; ve bu soruyu soran satıra "true" (doğru) veya "false" (yanlış) sonucunu verir. Önce, bu işlemleri yaptırtan operatörleri ve işlevlerini sıralayalım:

== Eşit operatörü. İşaretin sağında ve solundaki değerlerin eşit olması halinde true (doğru) sonucunu gönderir.

!= Eşit değil operatörü. İşaretin sağında ve solundaki değerlerin eşit olmaması halinde true (doğru) sonucunu gönderir.

> Büyüktür operatörü. Soldaki değer, dağdaki değerden büyük ise true (doğru) sonucunu gönderir.

< Küçüktür operatörü. Soldaki değer, dağdaki değerden küçük ise true (doğru) sonucunu gönderir.

>= Büyük veya eşit operatörü. Soldaki değer, dağdaki değerden büyük veya bu değere eşit ise true (doğru) sonucunu gönderir.

<= Küçük veya eşit operatörü. Soldaki değer, dağdaki değerden küçük veya eşit ise true (doğru) sonucunu gönderir.

Daha sonra örneklerini göreceğiz; ve karşılaştırma işleminden sonra ne olduğunu, Javascript'in nasıl bir yol izlediğini ele alacağız. Şimdi sadece böyle bir karşılaştırma işleminde nasıl sonuç verdiğini görmekle yönetinelim. Düz yazı programınızda şu HTML kodunu yazın ve karsilastirma1.htm adıyla kaydedin:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te Mukayese</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// degisken tanimlari
var x, y, z
x = 7
y = 7
z = 13
// -->
</SCRIPT>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
//mukayese sonuclarını gösterelim
document.writeln("x = " + x)
document.writeln("y = " + y)
document.writeln("z = " + z)
document.write("x değişkeni y değişkenine eşit mi, (x==y)? ")
document.writeln(x==y)
document.write("y değişkeni z değişkenine eşit mi, (y==z)? ")
document.writeln(y==z)
// -->
</SCRIPT>
</PRE>
</BODY>
```

```
</HTML>
```

Bu kod bir tarafta açıkken, oluşturduğunuz dosyayı Browser'da açın ve sonuca bakın; sonra kodunuzda x, y ve z değişkenlerinin değerini değiştirerek, Browser'a sayfayı yenilettirin; sonucun nasıl değiştiğini inceleyin. Javascript'in sonuç true (doğru) ise nasıl ilerlediğini, sonuç false (yanlış) ise nasıl ilerlediğini daha sonra ayrıntılı olarak göreceğiz. Fakat derhal öğrenmeniz gereken kuralı burada belirtelim: Javascript, karşılaştırma sonucu doğru ise, karşılaştırma komutundan sonraki ilk emri (veya emir grubunu), değilse bir sonraki emri (veya emir grubunu) icra eder. Bunun ayrıntılarını da ele alacağız. Şimdilik sadece bu kuralı bir kenara yazın!

Bu konuyu bitirmeden, bir de ne ile neyi karşılaştırabileceğinizden söz edelim. Javascript 1.0'de elmalarla (sayılarla) armutları (alfanümerik değişkenleri) karşılaştırabilir ve doğru sonucunu verebilirdi. Yani, Javascript 1.0 açısından 7 ile "7" eşitti; Javascript önce karşılaştırılan değişkeni sayıya çevirip çeviremeyeceğine bakar; çevirebiliyorsa, karşılaştırmayı yapardı. Daha sonraki sürümlerinde durum değişti; Javascript'i tasarlayanlar bu çevirme işlemini programcıya bıraktılar; dolayısıyla Javascript elmalarla armutları ayırteder oldu!

Alfanümerik İşlemleri:

Javascript'in alfanümerik değişkenlerin değerleri ile sadece toplama işlemi yaptığını söylemiştik. Bu durumda buna toplama değil birleştirme, ekleme işlemi denir.

Aşağıdaki kodu düz yazı programınızla oluşturup, birlestir.htm adıyla kaydederseniz ve Browser'ınızda incelelerseniz, Javascript'in alfanümerik değerleri nasıl birleştirdiğini görmüş olursunuz:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te Ekleme</TITLE>
</HEAD>
```

```
<BODY>
<PRE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
//değiskenlerimizi tanımlayalım
var a, b, c, d
a = "www"
b = "pcworld"
c = "com"
d = "tr"
//sonuçlarımızı görüntüleyelim
document.writeln("a değişkeninin değeri: \""+a+"\".")
document.writeln("b değişkeninin değeri: \""+b+"\".")
document.writeln("c değişkeninin değeri: \""+c+"\".")
document.writeln("d değişkeninin değeri: \""+d+"\".")
document.writeln("\nŞimdi bunları birleştirelim:\n")
document.write("a + b = ")
document.write(a + b)
document.write("\nAralarına nokta koyalım: ")
document.write(a + "." + b)
document.write("\nŞimdi c değişkenini, noktasıyla birlikte ekleyelim: ")
document.write(a + "." + b + "." + c)
document.write("\nŞimdi de d değişkenini, noktasıyla birlikte ekleyelim: ")
document.write(a + "." + b + "." + c + "." + d)
document.write("\nİşte a + b + c + d'nin sonucu: ")
document.write(a + "." + b + "." + c + "." + d)
// -->
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

Bu dosyada, iki tür yazdırma komutu kullandığımıza dikkat ediyor musunuz: "document.writeln()" ve "document.write()". Bu iki komut arasındaki farkı bulabilir misiniz? (İpucu: "document.write()" yönteminde satırbaşı yaptırtmak için "\n" kullanıyoruz!)

Javascript, alfanümerik değerlere diğer aritmetik işlemleri yapamaz. Buna karşılık, Javascript sayılarla yaptığı bütün karşılaştırma işlemlerini alfanümerik değişkenlerle de yapar ve doğru veya yanlış sonucunu bildirir.

Daha önce kaydettiğiniz karsilastirma1.htm dosyasını düz yazı programında açın ve değişkenleri tanımladığınız bölümünü şöyle değiştirin:

```
x = "Ali"
y = "Ali"
z = "Veli"
```

Sonra bu dosyayı, karsilaştirma2.htm adıyla kaydedip, Browser'da inceleyin; değişkenlerin değerini değiştirip, yeniden inceleyin ve ne sonuç aldığınıza bakın.

Şartlı İşlemler:

Javascript'te karşılaştırma yaparken şartlı (..ise ..yap!) işlemler de yaptırabilirsiniz. Şartlı işlemlerde ? (soru işareti) ve : (iki nokta üstüste) işaretlerini kullanırsınız.

Karsilastirma2.htm dosyasının bütün "document.write.." kodlarını silin ve yerine şu iki satırı yazarak, karsilastirma3.htm adıyla kaydedin.

```
sonucMsg = (y==x)?"y degiskeni x degiskenine esit!" : "y degiskeni x degiskenine esit degil!"
alert(sonucMsg)
```

Kod dosyası düz yazı programında açıkken, karsilastirma3.htm'i Browser'da açın. Uyarı kutusunda, iki değişkenin eşit olması şarti halinde görüntülenmesini istediğiniz mesajı göreceksiniz. Dosyada değişkenlerin değerini değiştirerek, sonucu inceleyin. Bunu sağlayan şartlı işlem komutu üç bölümden oluşuyor: şartın doğru olup olmadığını sorguladığınız, soru işaretine kadar olan karşılaştırma bölümü; iki nokta üstüste işaretine kadar olan şart doğru ise uygulanacak bölüm, sonra satır sonuna kadar olan şart yanlış ise uygulanacak bölüm. Yukarıda, "Javascript bir karşılaştırma yaptıktan sonra karşılaştırma sonucu doğru ise, karşılaştırma işleminden sonraki ilk komutu icra eder" demiştik. Burada da, x değişkeni ile y değişkeni aynı olduğu zaman, soru işaretiyle biten karşılaştırma işleminden hemen sonraki

ilk komut (veya ifade) seçiliyor; değişkenler aynı olmadığı taktirde, iki nokta üstüste işaretinden sonra gelen komut (veya ifade) seçiliyor.

Mantıksal İşlemler:

Javascript, karşılaştırma işlemini Boolean (.. ve .. doğru ise ... yap!) mantıksal işlemlerini kullanarak da yapabilir. Burada, Boolean mantığından kısaca söz edelim.

Şimdi, bir baba, kızı Ayşe ile oğlu Ali'ye diyor ki, "İkiniz de sınıfını geçerseniz, bu yıl sizi Antalya'ya deniz kenarında bir ay geçirmek üzere teyzenizin evine göndereceğiz!" Ali sınıfını geçer, Ayşe geçemezse iki çocuk da teyzelerine gidemeyecekler; Ali sınıfını geçer, Ayşe de geçerse, iki çocuk da Antalya'ya gidecek; Ali de, Ayşe de sınıflarını geçemezse, iki çocuk da yaz tatilini oturdukları yerde geçirecekler.

Javascript'in Boolean mantığını sorgulama işaretleri şunlardır:

&&	Mantıksal Ve: iki koşul da doğru.
||	Mantıksal Veya: ya birinci, ya da ikinci koşul doğru.
!	Mantıksal Değil: Tek koşula uygulanır; koşul doğru ise sonuç false (yanlış), koşul yanlış ise sonuç true (doğru) olur.

Bu mantığı, şimdi Boolean ifadesi haline getirelim:

Ali=geçti && Ayşe=geçti : çocukların ikisi de tatile gidiyor.

Ali=geçti || Ayşe=geçti : çocukların ikisi de tatile gidemiyor.

!(Ali=geçti) : çocukların ikisi de tatile gidemiyor.

!(Ayşe=geçti) : çocukların ikisi de tatile gidemiyor.

İşlemlerde Sıra:

Javascript'te işlemler yukarıdan aşağı ve soldan sağa yapılır, ama aritmetik işlemlerde bu kuralın bazı istisnaları vardır. Javascript kodlarınızda beklediğiniz sonuçları alamıyorsanız, önce işlemlerinizi işlem sırasına uygun yazıp yazmadığınızı kontrol edin.

Javascript'te işlemlerin genel sırası şöyledir: Atama işlemleri, şartlı işlemler, mantıksal ve/veya/değil işlemleri, karşılaştırma işlemleri, aritmetik işlemler, fonksiyonlar.

Javascript'te aritmetik işlemler ilke olarak soldan sağa doğru yapılır. Örneğin

x = a * b + c

denklemi çözülürken, önce a ile b çarpılır, elde edilecek sayıya c eklenir.

Fakat bir denklemde parantez içine alınmış ifade varsa, Javascript önce parantezin içini halleder. Örneğin

x = a * (b + c)

denklemin önce b ile c toplanır, elde edilecek sayı a ile çarpılır.

Eğer bir işlemde Javascript aritmetik işlemleri sıraya sokmak zorunda kalırsa, toplama, çıkartma, çarpma ve bölme sırasıyla yapar. İşlemlerin sırası konusunda kuşkunuz varsa, mümkün olduğu kadar çok parantez kullanın.

Program Akış Denetimi

Javascript veya bir başka programlama dili ile yazacağınız programda, zaman zaman programın akışını, programa bırakmanız gerekir. Programcı olarak siz, programın izleyeceği alternatif yolları belirlersiniz; fakat bu yollardan hangisini izleyeceğine, sizin koyduğunuz şartları değerlendirerek program kendisi karar verir. Programcılığın gerçek anlamda zevki de bu noktada başlar. Şimdi Javascript'in programın akışını kontrolde kullanacağınız unsurlarını tanımaya başlayalım.

Aşağıda yazacağımız iki Javascript kodunda Browser'a, ziyaretçinin ekranındaki bir form unsurunun içini doldurtma tekniğini kullanacağız. Henüz ele almadığımız bu tekniğin ayrıntılarını FORM nesnesinin özelliklerine değinirken ele alacağız.

If (Eğer .. ise)

Javascript programının, bir değişkenin değerine göre yönlenmesini sağlayan ilk aracı, "if" (eğer) ifadesidir. İşte size program akışını "if" yoluyla kontrol eden bir HTML sayfası:

```
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te Mukayese</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
function goster(secme)
      {
var endeks, secilenYemek
endeks=secme.selectedIndex
secilenYemek=secme.options[endeks].text
if (secilenYemek == "Adana Kebap")
             {
        document.tarifForm.tarif.value=secilenYemek + " seçtiniz! Bravo!"
             }
if (secilenYemek == "Yogurtlu Iskender")
             {
document.tarifForm.tarif.value=secilenYemek + " seçtiniz. Sizi tebrik ederiz!"
             }
if (secilenYemek == "Biber Dolma")
             {
document.tarifForm.tarif.value=secilenYemek + " seçtiniz. Zevkinize hayranız!"
             }
if (secilenYemek == "Yaprak Sarma")
             {
       document.tarifForm.tarif.value=secilenYemek + " yapması biraz zordur; ama çok
lezizdir"
             }
      }
```

```
</SCRIPT>
</HEAD>
<BODY>
<BR>
<FORM NAME="tarifForm">
<P>Hangi yemeğin tarifini istiyorsunuz? Lütfen istediğiniz yemeği seçin:</P>
<SELECT NAME="yemeklistesi" onChange="goster(this)">
<OPTION SELECTED> Bir yemek seçiniz
<OPTION> Adana Kebap
<OPTION> Yogurtlu Iskender
<OPTION> Biber Dolma
<OPTION> Yaprak Sarma
</SELECT>
<P></P>
<INPUT NAME="tarif" VALUE="" SIZE=55>
</FORM>
</BODY>
</HTML>
```

Bu sayfayı denetim1.htm adıyla kaydedin ve Browser'inizde açın. Şimdilik, yemek tariflerini vermediğimize aldırmayın. Daha sonra bunu yapmanın yolunu da göreceğiz. Bu sayfadaki Javascript kodunda "goster()" isimli bir fonksiyon var; bu fonksiyon kendisini harekete geçmeye çağıran nesnenin kendisine "secme" adı altında bir takım bilgiler göndermesini şart koşuyor. Sayfamızda yer alan ve adı "tarifForm" olan FORM nesnesi, kendi unsurlarından SELECT nesnesindeki her değişiklikte, "goster" fonksiyonunu harekete geçiriyor. "yemeklistesi" isimli SELECT etiketinin içindeki onChange (değişiklik halinde) isimli Javascript komutu, bir metod'tur ve burada kendi unsurlarını ve değerlerini topluca "gonder" fonksiyonuna gönderecektir. (Şimdilik bu metod ve metodu kullanırken yazdığımız THIS ifadesi üzerinde durmayın. Sırası gelecek.)

Fonksiyonun başında iki değişken tanımlıyoruz: endeks ve secilenYemek. Endeks değişkeninin değeri, fonksiyona gönderilecek nesnelerden birinden alınıyor. Bu nesne "selectedIndex" adını taşıyor. "selectedIndex" denen şeyin, HTML belgesinin Form

nesnesinin bir özelliği (property) olduğunu belirtmekle yetinelim; ayrıntılarını Form nesnesinin özelliklerini incelerken göreceğiz. Bir Form'un içinde SELECT nesnesi varsa, Browser mevcut SELECT ögelerini dizi-değişken yaparak 0'dan itiberen numaralar ve kullanıcının seçtiği SELECT ögesinin numarası "selectedIndex" adıyla bir kenara kaydeder.

Fonksiyonumuzdaki ikinci değişken olan "secilenYemek" değişkeni de değerini, SELECT nesnesinin seçilen ögesinin metninden alıyor. Şimdilik bu tanımlar sizin için anlam ifade etmiyorsa, üzerinde durmayın; iki değişkenimize ziyaretçinin sayfada yapacağı tercihe bağlı olarak değerler kazandırdığımızı düşünün. Şimdi dikkatimizi şu satıra verelim:

```
if (secilenYemek == "Adana Kebap")
```

Yukarıda öğrendiğimize göre, bu satırdaki ifadede Javascript'e ne demiş oluyoruz? "Ey Javascript, 'secilenYemek' adlı değişkenin değerine bak; orada gördüğün metin, 'Adana Kebap' alfanümerik değeri ile aynı ise, bu satırdan hemen sonraki işi yap! aynı değilse, ber sonraki satıra git, oradaki işi yap!" diyoruz. Buradaki "aynı ise" ifadesini, "==" işaretleriyle sağladığımızı biliyorsunuz; "hemen sonraki iş" ifadesinin de bir karşılaştırma komutu doğru sonuç verirse, Javascript'in bu karşılaştırma komutundan hemen sonra gelen satırdaki işi yapmasından kaynaklandığını hatırlıyorsunuz. Ziyaretçi "Adana Kebap" seçeneğini seçti ise, "Seçim, Adana Kebap ise" sorgulamasının sonucu doğru olacak, ve Javascript

```
document.tarifForm.tarif.value=secilenYemek + " seçtiniz! Bravo!"
```

komutunu icra ederek, ekrana "secilenYemek" değişkeninin değeri yazacak ve buna "seçtiniz! Bravo!" kelimelerini ekleyecektir. Sorgulamanın sonucu doğru değil ise, yani ziyaretçi Adana Kebap'ı seçmemişse, Javascript bir sonraki satırına atlayacaktır. Burada "bir sonraki satır" ile şu anda içinde bulunduğumuz "if" ifadesinin açılan ve kapanan süslü parantezle belirlenen sınırının bittiği yeri kastediyoruz. Yazma kolaylığı bakımından ve açtığımız parantezi kapattığımızdan emin olmak için süslü parantezleri ayrı satırlara yazarız. Ama bu şart değildir. Yani istese idik, burada üç satıra yayılan "sonraki satır" aslında tek satırda yazılabilirdi.

Javascript'in, ziyaretçinin Adana Kebap'ı seçmemesi halinde atlayacağı bir sonraki satırda da bir "if" ifadesi var. Javascript bu kez bu sorgulamanın sonucunu alacak ve bu sorgulama doğru sonuç verirse (yani ziyaretçi Yoğurtlu İskender'i seçmişse), hemen sonraki satırı, seçmemişse, bir sonraki satırı icra edecektir. Böylece dört sorgulamayı da bitiren Javascript, çalışmasına son verecektir. Taa ki, ziyaretçi, yeni bir tercih yapıncaya kadar. SELECT etiketinin içindeki onChange (değişiklik halinde) yönlendiricisi, bu kutuda olacak her değişiklikte, formun SELECT nesnesinin ögelerini topluca "gosder" fonksiyonuna gönderecektir.

Bir "if" sorgusunun içinde bir veya daha fazla "if" bulunabilir. Böyle iç-içe "if" ifadesi yazarsanız, herbirinin icra alanının başladığı ve bittiği yeri gösteren süslü parantezlerine dikkat edin.

If ... Else (Eğer .. Diğer)

Kimi zaman bu kadar çok seçeneğe tek tek "if" ifadesi yazmak gerekmeyen durumlar olabilir. Javascript'e "Eğer şu değişkenin değeri şu işe şunu yap, diğer bütün durumlarda ise şunu yap!" demek istiyebilirsiniz. "If... Else" ifadesi ile bunu sağlarız.

denetim1.'in dört "if" ifadesini de silin ve yerine şunları yazın:

```
if (secilenYemek != "Adana Kebap")
      {
document.tarifForm.tarif.value=secilenYemek + " tarifini veremiyoruz!"
      }
   else
      {
document.tarifForm.tarif.value="Adana Kebap Tarifini asağıda sunuyoruz:"
      }
```

Bu sayfayı denetim2.htm adıyla kaydedin ve Browser'inizde açın. Şimdi dikkatimizi şu satıra verelim:

```
if (secilenYemek != "Adana Kebap")
```

Yukarıda öğrendiğimize göre, bu satırdaki ifadede Javascript'e "Ey Javascript, 'secilenYemek' adlı değişkenin değerine bak; orada gördüğün metin, 'Adana Kebap' alfanümerik değeri ile aynı değil ise, bu satırdan hemen sonraki işi yap!" demiş oluyoruz. Buradaki "aynı değil ise" ifadesini, "!=" işaretleriyle sağlıyoruz; "hemen sonraki iş" ifadesinin de bir karşılaştırma komutu doğru sonuç verirse, Javascript'in bu karşılaştırma komutundan hemen sonra gelen satırdaki işi yapmasından kaynaklandığını hatırlıyorsunuz. Ziyaretçi "Adana Kebap" seçeneğini seçmedi ise, "Seçim, Adana Kebap değil ise" sorgulamasının sonucu doğru olacak, ve Javascript

```
"document.tarifForm.tarif.value=secilenYemek + " tarifini veremiyoruz!"
```

komutunu icra ederek, ekrana "secilenYemek" değişkeninin değeri yazacak ve buna "tarifini veremiyoruz!" kelimelerini ekleyecektir. Sorgulamanın sonucu doğru değil ise, yani ziyaretçi Adana Kebap'ı seçmişse, Javascript sorgulamanın sonucunun "diğer" sınıfına girdiğini anlayacak ve "Else" ifadesini icra edecektir. Yani ekrana "Adana Kebap Tarifini aşağıda sunuyoruz:" yazacaktır. (Tarifi vermediğimizin üzerinde de durmayın. İlerde bu tür uzun ifadeleri Javascript ile sayfaya yazdırmanın yollarını da göreceğiz.)

Özetlersek, "if" ifadesinde ileri sürdüğünüz şart doğru ise ilk komut dizisi, doğru değilse "Else" bölümündeki komut dizisi icra edilir.

Program Döngü Denetimi

Javascript'te bazen işler yukarıdaki örnekte olduğu gibi açık ve belirgin şartlara bağlı ve bir kere icra edilecek türden olmayabilir. Javascript, bir işi belirli (veya kendi belirleyeceği) kere yapmasını isteyebilirsiniz. Bu tür tekrarlanan işleri Javascript'e bir kaç satırlık bir kodla yaptırtmak çoğu kez bizi sayfalar dolusu kod yazmaktan kurtarır.

"for" döngüsü

Diyelim ki, sitenizde santigrad dereceyi fahrenhayt dereceye çeviren tablo sunmak istiyorsunuz. 0'dan 100'e kadar bütün santigrad derecelerin (fahrenhayt = santigrad * 9 / 5

+ 32) formülüyle karşılığını bulup, bir tabloya yazabilirsiniz. Hesabı yapmanın güçlüğünün yanı sıra, HTML kodunuz tam 100 adet <TR><TD>..</TD><TD>..</TD></TR> kodu içerecektir. Oysa bunu iki satırlık bir Javascript kodu ile Browser'a yaptırabilirsiniz. Aşağıdaki kodu, dongu1.htm adıyla kaydedin ve Browser'ınızda açın:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te döngü</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// santigrad-fahrenhayt tablosu
document.writeln("<TABLE Border=\"1\">")
document.writeln("<TR><TD>Santigrad</TD><TD>Fahrenhayt</TD></TR>")
for (santigrad = 0; santigrad <= 100; ++santigrad)
      {
fahrenhayt = santigrad * 9 / 5 + 32
document.writeln("<TR><TD>" + santigrad + "</TD><TD>" + fahrenhayt + "</TD></TR>")
      }
document.writeln("</TABLE>")
// -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Burada "for" ifadesiyle tekrarlanan bir döngü oluşturuyoruz. "for" döngüsüne geçmeden Javascript'e, sayfaya bir kaç HTML komutu göndermesini bildiriyoruz; bu komutlar sayfada bir tablo başlatıyor; tablonun birinci sırasına iki kutu oluşturarak, içlerine "Santigrad" ve "Fahrenhayt" yazıyor. Şimdi asıl, "for" ifadesi üzerinde duralım:

"for" ile oluşturacağımız döngü, bir sayaca göre ilerler. Burada sayaç olarak "santigrad" adında bir değişken oluşturuyoruz, Javascript'e "Santigrad adında bir değişken

yap, içine 0 değerini koy; ve bu değişken 100'den küçük veya 100'e eşit ise bazı işler yap!" demiş oluyoruz. Bu arada Javascript, diğer dillerin benzeri komutuna göre bize bir kolaylık sağlıyor: Javascript'in her döngü yapacağı işten sonra santigrad'ın mevcut değerine 1 eklemesini aynı satırda ve kolayca sağlıyoruz. "for" işleminin başladığını belirten açılan süslü parantezden sonra Javascript'e her döngüde yapacağı iki işi bildiriyoruz:

```
fahrenhayt = santigrad * 9 / 5 + 32
```

Bu komutla Javascript, "fahrenhayt" adını verdiği bir değişken oluşturuyor ve ilk değer olarak, santigrad değişkeninin değerini 9 ile çarpıp, 5'e bölüyor, bulduğu sayıya 32 ekliyor. ilk turda, santigrad değişkeninin değeri 0 olduğuna göre, fahrenhayt değişkeninin değeri 32 olarak bulunuyor; ve Javascript ikinci işe geçiyor:

```
document.writeln("<TR><TD>" + santigrad + "</TD><TD>" + fahrenhayt + "</TD></TR>")
```

Bu iş ise, sayfaya bir tablo sırası ile bir tablo kutusu göndermek; bu kutunun içine santigrad değişkeninin değerini yazmak; sonra bu kutuyu kapatıp yeni bir kutu açmak; bu kutunun içine fahrenhayt değişkenini değerini yazarak, kutuyu ve tablo sırasını kapatmaktan ibaret.

Javascript, birinci turu bitirdiğinde, "for" komutunun tayin ettiği ölçüde sayaç değişkeninin arttırıp veya azaltıp, ikinci tura başlayacaktır. Bu ne zamana kadar böyle devam edip gidecektir? Sayaç değişkeninin değeri, tayin ettiğiniz sınına ulaşıncaya kadar. Nedir bu sınır: santigrad değişkeninin değerinin 100'e eşitlenmesi. Bu kadar uzun bir tablo vermek istemiyorsanız, bu değeri istediğiniz bir sayıya indirebilirsiniz ve sayacı birer birer arttırma yerine, mesela beşer beşer attırabilirsiniz. Sayacı beşer-beşer arttırmak için "for" komutunun son argümanını "santigrad +=5" şeklinde yazmalısınız.

Şartlı döngü: while

Javascript kodu yazarken öyle durumlar olabilir ki, programın sayaç değişkeninin her değeri için yapmasını istediğiniz işi yapmasındansa, sadece belirli bir şart karşılanıyorsa

yapmasını tercih edebilirsiniz. Bunu "while" (...iken) ifadesiyle sağlarız. Bu yöntemle Javascript'e "filanca işi yap, ama falanca şart varsa (veya yoksa)" demiş oluruz.

"while" ile kuracağınız döngünün şartı rolünü oynayacak değişkenin daha önceden oluşturulması, içeriğinin başka bir yöntemle belirlenmesi ve değiştirilmesi gerekir; "for" döngüsünün aksine "while" şartlı döngüsü bunu sizin için otomatik olarak yapmaz. Ayrıca koyduğunu şarta bağlı olarak, "while" döngüsü sırasında yapılmasını istediğiniz iş hiç yapılmayabilir.

Diyelim ki, bir elektronik ticaret siteniz var; ve müşteri adayı ziyaretçileriniz, bazı malları Web sayfanızda toptan ısmarlayabilirler. Böyle bir siparişi alabilmek için, o maldan stokunuzda yeterli miktarda bulunması gerekir. Aşağıdaki kodu dongu2.htm adıyla kaydederseniz, ısmarlanan bir miktar üzerinden kaç sipariş karşılayabileceğini sayfanız otomatik olarak hesaplayacaktır.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te döngü</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// envanter kontrolü icin degisken
var stok = 700
// -->
</SCRIPT>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE="Javascript1.2">
function hesapla(miktar)
      {
var endeks, talepMiktar, satis
```

```
endeks=miktar.selectedIndex
talepMiktar=miktar.options[endeks].text
satis = 0
while (talepMiktar < stok)
            {
        stok -= talepMiktar
       satis++
            }
document.miktarForm.sonuc.value=talepMiktar + " ton itibariyle "+ satis + " adet
sipariş için stokumuz var!"
      }
</SCRIPT>
</HEAD>
<BODY>
<BR>
<FORM NAME="miktarForm">
<P>Alacağınız miktarı lütfen seçiniz:</P>
<SELECT NAME="miktar" onChange="hesapla(this)">
<OPTION SELECTED> Miktarı seçiniz
<OPTION>100
<OPTION>200
<OPTION>300
<OPTION>400
</SELECT>
<P></P>
<INPUT NAME="sonuc" VALUE="" SIZE=55>
</FORM>
</BODY>
</HTML>
```

Bu kodun da bir çok unsuru üzerinde durmayalım; bizi döngü açısından ilgilendiren bölümlerini ele alalım. Gördüğünüz gibi üç değişken tanımlıyoruz; "stok" değişkeninin değerini biz veriyoruz; miktarTalep değişkeni ise sayfamızdaki Form nesnesinin "miktar" adlı SELECT nesnesinden alınıyor. Bu arada, işlemi yapan fonksiyonun içinde "satıs" adlı bir değişken oluşturuyoruz ve değerinin, döngünün içinde birer birer artmasını sağlıyoruz. Bu

kodun mantığı, müşterinin sipariş vereceği miktarda, stokumuzdan kaç adet toptan satış yapılacağını bilmek olduğuna göre, Javascript, döngünün içinde çalışırken şöyle düşünecektir:

"Benden, stok değişkeninin değerinden talepMiktar değişkeninin değeri kadar çıkartmam; sonra da satıs adlı değişkenin değerini bir adet arttırmam isteniyor. Ama bunu ancak ve sadece talepMiktar değişkeninin değeri stok değişkeninin değerinden az iken yapabilirim!"

Nitekim Javascript, talepMiktar değişkeninin değeri stok değişkeninin değerine eşit veya fazla hale geldiği anda, döngüye son verecek ve "while" ifadesinin dışına çıkacaktır. Döngüden çıktığı anda Javascript'in yapacağı iş ise ekrandaki Form'un sonuc adlı kutusuna kendisine bildirilen metni yazmaktan ibaret!

do...while

"while" döngüsünün, koyduğunuz şarta göre bazen hiç icra edilmeyebileceğini söyledik. Yukarıdaki örnekte müşteri öyle bir rakam verir ki, sipariş miktarı stoktan yüksek olduğu için, hesaplama hiç yapılmaz ve müşteriye kaç parti sevkiyat yapacağımızı değil, sevkiyat yapamayacağımızı bildiririz. Fakat bazı durumlarda döngünün bir kere işlemesi, fakat ikinci ve daha sonraki seferler için bir değişkenin değerine bakması gerekebilir. Akla gelecek ilk örnek, bir elektronik ticaret sitesinde, müşteriye bir malı almak isteyip istemediği sorusudur. Bu soruyu bir kere sormak zorundayız; müşteri bu soruya "Hayır!" yanıtını verirse, alış-veriş sorusunu soran döngüyü durdurmamız gerekir; müşteri alışverişe devam etmek istediğini bildirirse, döngü devam edecek ve müşteriye alış-veriş sorusu yeniden sorulacaktır.

Bunu, do...while (Yap ... iken) komut grubuyla yapabiliriz. Do (Yap) komutu bir kere icra edilir ve ondan sonra while (..iken) şartı aranır. Bu şart yerine geliyorsa, Do'daki komut yeniden icra edilir, şartın değişip değişmediğine yeniden bakılır.

Düz yazı programında aşağıdaki kodu yazın ve dongu5.htm adıyla kaydedin:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te Switch</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// gecici bir degiskene ihtiyacımız var
gecici = "Bir"
// müsteriye soruyu bir defa soralım
     do
      {
              //buraya gerçek alış veriş halinde gerekli işlemler girecek
       yanit = prompt(gecici + " gömlek almak ister misiniz?", "Evet")
       gecici = "Bir başka"
      }
while (yanit == "Evet")
// -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Bu programda Browser'ın kullanıcıdan bilgi alma aracı olan Prompt kutusunu kullanıyoruz; müşteri adayına "bir" gömlek almak isteyip istemediğini soruyoruz. Buradaki "bir" kelimesi, geçici değişkenimizden alınıyor; müşteri ikinci alış-verişini yapmaya karar verirse, bu değişkenin değerini "Bir başka" olarak değiştiriyoruz. Bu programda, müşteri-ziyaretçinin ilk "Evet" yanıtı halinde sayfamızda yapılabilecek siparişi alma, fiyatı bildirme, onay isteme ve benzeri işlerin hiç biri yok; Javascript sadece aynı soruyu yeniden soruyor; taa ki ziyaretci Prompt kutusuna "Evet" dışında bir şey yazıncaya kadar. Do komutu, while bölümüne yazdığınız şart değişinceye kadar icra edilecektir. (Tabiî, buradaki örnekte bütün süreç, müşteri-ziyaretçinin Prompt kutusunda Cancel (İptal) düğmesine basmasıyla da sona erecektir; ama bu Prompt kutusunun bir özelliğidir.)

"Break" ve "Continue"

Şartlı döngüde, tekrar eden iş, şartın yerine geldiği noktada kendiliğinden kesilecektir. Aynı otomasyonu "for" döngüsünde break (kes) ve continue (devam et) komutlarıyla biz de sağlayabiliriz.

Javascript, Break ile karşılaştığı anda döngüyü keser ve icraata döngüden sonraki ilk ifadeden devam eder. Continue ise Javascript'in döngünün o andaki adımını durdurup, döngünün başına dönmesini sağlar; döngü baştan devam eder.

dongu1.htm adıyla kaydettiğiniz dosyada, fahrenhayt'ı hesaplattırdığınız komutla, bunu HTML belgesine yazdırdığınız komut arasına şu satırı koyarak, dosyayı, dongu3.htm adıyla saklayın.

```
if (santigrad == 13) continue
```

Browser'da incelediğinizde, Fahrenhayt derecenin 13 santigrad derece için hesaplanmadığını göreceksiniz. Eklediğiniz satırdaki "continue" komutu, santigrad 13'e eşitlendiğinde, döngüyü olduğu yerde keserek, bir sonraki adımdan yeniden başlattı.

Eklediğiniz bu satırdaki "continue" kelimesini "break" diye değiştirir, dosyayı dongu4.htm adıyla kaydeder ve sayfayı Browser'da açarsanız, bu kez listenin 12'de kesildiğini göreceksiniz; çünkü break komutu döngüyü olduğu yerde durdurup, Javascript'i bir sonraki adımdan devama zorladı.

"Switch"

Javascript'in switch (değiştir) komutu, programın bir değişkenin değerine göre, belirli bir işi yapmasını sağlar. Sözgelimi, elektronik satış sitenizde, müşteri-ziyaretçiniz, alacağı gömleğin beden ölçüsünü seçecek ve siz de bu ölçüye göre, siparişi kaydedecek ve müşteriye siparişin alındığını bildireceksiniz. Javascript programının mantığı şöyle olacak: "Müşteri büyük beden'i seçti ise Büyük bedenle ilgili işleri yap; küçük bedeni seçti ise küçük bedenle ilgili işleri yap; orta bedeni seçti ise orta bedenle ilgili işleri yap!"

Javascript'e switch komutu, bir veri kümesi gösterilerek verilir ve bu veri kümesi içinde hangi değere göre ne yapacağı case (halinde) ifadesiyle bildirilir: "Değerin A olması halinde şu işi yap; değerin B olması halinde şu işi yap!" gibi. Şimdi şu kodu düzyazı programında yazın ve dosyayı dongu6.htm adıyla kaydedin:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te Switch</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// degiskeni tanımlayalım
var olcu
// degiskeni ziyaretciden alalım
olcu = prompt("Ölçünüzü belirtin: B-Büyük K-Küçük O-Orta", "O")
// bu degiskene göre yanıt verelim
switch (olcu)
      {
case "O" :
case "o" :
       alert ("Orta boy gömlek siparişiniz alındı!")
         break
case "B" :
case "b" :
       alert ("Büyük boy gömlek siparişiniz alındı!")
         break
case "K" :
case "k" :
       alert ("Küçük boy gömlek siparişiniz alındı!")
         break
default: alert ("Lütfen beden ölçünüzü belirtiniz!")
      }
// -->
</SCRIPT>
</HEAD>
```

```
<BODY>
</BODY>
</HTML>
```

Bu sayfayı Browser'da açtığınızda, karşınıza bir diyalog kutusu gelecektir.

<js003.tif ve js004.tif -- Resimaltı: Javascript'in "prompt" komutu ile IE'nin ve Netscape'in açtığı diyalog kutuları, başlıkları dışında, aynıdır.>

Bu kutuda varsayılan değer olarak Orta Boy ölçü için O harfi bulunuyor. isterseniz, komutta değişiklik yaparak. bu kutuya istediğiniz değeri girebilirsiniz. Javascript, bu kutunun sağlayacağı değeri, "olcu" adı altında, switch komutuna iletiyor; switch komutu da "olcu" bilgisinin üç ayrı halinde üç ayrı yanıt veriyor. Böyle bir programı gerçek hayatta uygularken, yapılacak iş olarak sadece bir Alert kutusuyla yetinmeyerek, söz gelimi bu noktada Javascript'i farklı fonksiyonlara sevkederek, diyelim ki bir sipariş belgesi hazırlatabilirsiniz. Bu örnekte Javascript, olcu bilgisinin farklı hallerinde sadece farklı uyarı kutuları vermekle yetiniyor. Bu programda dikkatinizi çeken bir diğer nokta, her bir hale ilişkin case grubunda yer alan break komutu olmalı. Olcu isimli switch döngüsünün işi gerekli mesajı verdikten sonra bittiğine göre, Javascript'in diğer case'leri değerlendirmesini önlememiz gerekir; bunu break ile sağlıyoruz.

Ayrıca bu programda, ziyaretçi ile ilişkilerde gözönünde tutmanız gereken bir ilkenin uygulandığını görüyorsunuz: ziyaretçinin büyük beden için büyük ya da küçük harf B tuşuna basacağını bilmediğimiz için, kendimizi her iki ihtimale karşı da hazırlıyoruz. İyi bir programcılık tekniği, özellikle ziyaretçilerden girdi beklediğiniz hallerde, büyük harf-küçük harf durumlarına karşı hazırlıklı olmaktır.

Javascript'te Fonksiyon

Javascript, klasik HTML'in yapamadığı şeyler yapabilir; ziyaretçiden veya ziyaretçinin Browser programından alacağı bir bilgiyi değerlendirerek bir dizi işlem yapabilir; ziyaretçiyi sitemizde belli bir sayfaya yönlendirebilir; kısaca sayfalarımıza ziyaretçi ile etkileşme imkanı kazandırır.

Yukarıdaki çeşitli örneklerde gördüğünüz işlerin çoğu bir kere başvurulan işlerdi; fakat çoğu zaman sayfanızdaki bir Javascript işleminin defalarca yapılması gerekebilir. Hatta öyle işlemler olur ki, Javascript, başka bir işi yapmadan önce, mutlaka bu işlemi yapmak isteyebilir.

Bu tür tekrarlanan işleri, bu işin yapılması için gerekli bilgi kümesi ve komutlarla birlikte bir grup haline getirsek ve bu gruba bir isim versek; sonra bu iş kümesine ihtiyaç olduğu yerde Javascript'e "Filanca isimli grubu al; içinde belirtilen işleri yap, sonra sonucu bu noktaya bildir!" desek; acaba işlerimiz daha kolay hale gelmez mi?

İşte bu tür, bir isim altında toplanmış işlem paketlerine Function (işlev) adı verilir; Javascript kendisine "Şu fonksiyonu yap!" dediğiniz noktada yapmakta olduğu durdurur ve fonksiyon paketindeki işleri yapar; bu paket ortaya bir değişken veya sonuç çıkartıyorsa, o bilgiyi edinmiş olarak fonksiyon için yarım bıraktığı işleme geri döner. Fonksiyon yazmanızdaki birinci sebep, büyük bir ihtimalle, Javascript'e, Fonksiyon'un sağlayacağı bilgiyi kazandırmaktır.

Genel hatlarıyla fonksiyon, şu formüle göre yazılır:

```
function fonksiyonun_adı (argüman1, argüman2, ... argümanN){
işlemler
      }
```

Fonksiyonlar mutlaka SCRIPT etiketinin içinde yer alır. Daha sonra kullanılacak (çağrılacak) olsa da bütün fonksiyonlarınını HTML dosyasının HEAD bölümüne koymanız, Browser'ın Javascript yorumlayıcısı bakımından sürat sağlar. İyi bir programlama tekniği, bir sayfada gerekli bütün fonksiyonları, altalta, HTML'in HEAD bölümünde yer alacak bir SCRIPT etiketi içinde beyan etmek; sonra ihtiyaç olduğu yerde yeni bir SCRIPT etiketi koyarak bu fonksiyonu göreve çağırmaktır.

Bir fonksiyon, kendisini göreve çağıran komuttan veya fonksiyondan veri kümesi (argument) alabilir. Buna bir değerin fonksiyona geçirilmesi, ulaştırılması, verilmesi denilir. Bir fonksiyon, bir ya da daha fazla argüman alabilir. Fonksiyonun argümanları, bir isim altında toplanır ve bu bilgi kümesinin bölümlerine bu isimle atıfta bulunulur.

Fonksiyona değer gönderme ve değer alma

Bir fonksiyon ile Javascript programının diğer işlemlerinin ilk ilişkisi fonksiyona bir değer gönderme ve ondan bir değer almaktır. Bir fonksiyon, yaptığı işin sonucu olarak, kendisini göreve çağıran komuta veya fonksiyona kendi adıyla bir değer verebilir.

Bir örnek üzerinde düşünmeye başlayalım. Ziyaretçinizden telefon numarasını yazmasını istiyorsunuz. Ziyaretçi de sayfada bir INPUT kutusuna veya PROMPT diyalog kutusuna telefon numarasını yazıyor. Bu numarayı, sayfada başka bir programda veya bölümde kullanmadan önce, gerçekten doğru yazılıp yazılmadığını irdelemek istiyorsunuz. Ziyaretçinin verdiği telefon numarasını bir fonksiyona havale edebilirsiniz; bu fonksiyon telefon numarası arzu ettiğimiz biçimde yazılmışsa olumlu, yazılmamışsa olumsuz yanıt verebilir. Bütün yapacağımız şey, ziyaretçiden alacağımız bilgiyi, bu fonksiyona argüman olarak geçirmekten ibaret; fonksiyonun sonuç olarak verdiği değer doğru ise işleme devam edeceğiz, değilse ziyaretçiyi uyararak, doğru bilgiyi girmesini isteyeceğiz.

Şimdi düz yazı programınızda şu kodu yazıp, fonksiyon1.htm adıyla kaydedin:

```
<HTML>
```

```
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te Fonksiyon</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// fonksiyonu tanımlayalım:
function dogruMu(numara)
      {
var karakter = null
var durum = true
if(numara.length != 13)
            {
        durum = false // durum değişkenin değeri değişti
            }
        else
            {
            for(var i = 0; i <= 12; i++) {
            karakter = numara.charAt(i)
        if ( i == 0 && karakter == "(" )
            continue //birinci karakter parantez ise başa dön
        else //değilse devam et
            if( i == 4 && karakter == ")" )
                  continue //beşinci karakter parantez ise başa dön
            else //değilse devam et
                  if( i == 8 && karakter == "-" )
                  continue //dokuzuncu karakter çizgi ise başa dön
                  else //değilse devam et
            if( parseInt(karakter,10) >= 0 &&
                  parseInt(karakter,10) <= 9 )
                  continue //1, 4 ve 9 dışındaki karakter sayı ise devam et
                  else //değilse dur
                  {
            durum = false //değişkenin değeri değişti
            break //fonksiyon kesildi
            }
```

```
            }
        }
        return(durum) //çağıran işleme durum'un değeri bildirildi
        }
// -->
</SCRIPT>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE = "Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// Ziyaretciden telefon numarasını isteyelim
numara = prompt("Lütfen telefon numaranızı yazınız [(333)111-2222 gibi]", "")
if(dogruMu(numara))
        {
        document.writeln("Telefon numaranızı verdiğiniz için teşekkür ederiz.")
        document.writeln("Müşteri temsilciniz size ilk fırsatta arayacaktır")
        }
        else
        {
        document.writeln("Numaranızı örnekte görüldüğü şekilde yazmanızı rica ederiz")
        }
// -->
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

Bu sayfada, önce "if(dogruMu(numara))" şeklindeki ifadeye dikkat edelim. Javascript, bu noktada "dogruMu" fonksiyonu çağıracak, bunu yaparken de fonksiyona "numara" adı altında bir değer geçirecektir. (Bu değeri, çeşitli şekillerde elde edebiliriz; bir Form nesnesinde INPUT kutusundan alabileceğimiz gibi, burada olduğu gibi PROMPT diyalog kutusuyla da alabiliriz.) Fonksiyondan bütün beklediğimiz "if" sorgulamasına "true" (doğru) veya "false" (yanlış) şeklinde bir karşılık vermesidir. Yukarıda "if" döngüsünü ele alırken,

yanıtın doğru olması halinde ilk bölümdeki, yanlış olması halinde "else" bölümündeki komutların uygulandığını görmüştük. Şimdi burada dogruMu() fonksiyondan "true" sonucu dönerse, sayfaya teşekkür mesajı ve bilgi notu iki satır halinde yazılacak; "false" sonucu dönerse, ziyaretçi numarayı doğru yazması için uyarılacaktır. (Bu örnekte program bu noktada kesiliyor, oysa gerçek hayatta bu noktada ziyaretçiye bilgi girme aracının yeniden sunulması gerekir.)

Şimdi fonksiyona dönebiliriz. dogruMu() fonksiyonu, kendisine görev verecek işlemden veya fonksiyondan mutlaka bir argüman verilmesini istiyor. "numara" adıyla işleme sokulacak olan bu argüman, fonksiyon için bir nesnedir. Bu arada fonksiyonumuz kendi işleri için bir değişken oluşturuyor (karakter), ve içine boş değer koyuyor (null); daha sonra kendisinden beklenen değeri içine koyacağı bir değişken oluşturuyor ve içine varsayılan değer olarak "true" kelimesini koyuyor. Aksi kanıtlanmadığı taktirde, fonksiyonu göreve çağıran "if" döngüsüne bu değer bildirilecektir.

Fakat önce fonksiyonun gerçekten kendisine verilen bilgilerin, arzu edilen biçime uygun olup olmadığını sınaması gerekir. Bu bilgiyi tutan "numara" nesnesi, nesne olduğu için bazı özelliklere sahiptir; bu özelliklerden biri de uzunluğudur. Önce bu uzunluğun 13 karakter olup olmadığına bakıyoruz. Ziyaretçi, telefon numarasını yazdığı halde parantezleri veya kesme çizgisini unuttuysa, verdiği bilgilerin uzunluğu 13 karakterden az olacaktır, yani "numara.length" 13 olmayacak, "durum" değişkenin değeri "false" olarak değiştirilecektir. Yok, numaranın karakter sayısı 13 ise, fonksiyonumuz 13 karakteri, bir "for" döngüsü çerçevesinde tek tek irdelemeye başlayacaktır.

charAt(i)

Fonksiyonlarda ve diğer tür işlemlerde kullanabileceğimiz bir Javascript kısaltması, "filanca yerdeki karakter" diyebileceğimiz "charAt(i)" ifadesidir. Burada i yerine herhangi bir rakamı yazabilirsiniz. Bu kısaltma ancak sayı ya da alfanümerik değer içeren değişkene veya nesneye uygulanabilir.

Buradaki örneğimizde, Javascript, karakter adını verdiği ve içini boş bıraktığı değişkenin değeri olarak, "numara" nesnesinin o anda, o adımda irdelediği karakterini atamaktadır. Şimdi kendimizi Javascript'in yerine koyalım, ve buradaki "for" döngüsünün birinci adımında ne yapacağımızı belirtelim:

"Hımm.. karakter değişkeninin değeri olarak numara nesnesinin i'nci karakterini atamam isteniyor. Peki i nedir? 0. O halde karakter, nesnenin 0 numaralı ögesi olacaktır. Nedir bu: Açılan parantez. Güzel. O halde, karakter eşittir açılan parantez. Sonra, eğer i değişkeni sıfır ve karakter açılan parantez ise devam etmeliymişim! İkisi de doğru; o halde devam. O da ne? Devam etmem gereken yerde "continue" komutu var. Yani, işi burada kes, "for" döngüsünün başına git ve yeniden başla diyorlar. Peki, hadi en başa dönelim. i'yi bir adet arttıralım, 2 yapalım; 2, 12 olmadığı ve 12'den küçük olduğuna göre devam edelim.."

İkinci adımda Javascript, karakter değişkeninin değeri olarak numara nesnesinin ikinci karakterini atayacak ve, i artık sıfır olmadığı için ilk continue'den geri dönmeyecek ve birinci Else'i izleyen işi yapacaktır. Bu işin ilk şartı i'nin dört olmasıdır. i bu anda 2 olduğuna göre, Javascript bu kez hiç bir "continue"dan geri dönmeyecek ve en sondaki parseInt() işlemlerine ulaşacaktır. Ne zaman 1'nin değeri dört olursa, ikinci if'in ikinci şartı olan dördüncü karakterin kapanan parantez olup olmadığı irdelenecektir. Ziyaretçinin verdiği telefon numarasında dördüncü karakter kapanan parantez değil ise, fonksiyon kesilecek ve durum değişkeninin değeri "false" olacak ve bu değer, fonksiyonu göreve çağıran if döngüsüne bildirilecektir. Fonksiyon aynı sorguyu, dokuzuncu karakterin kesme çizgisi olup olmadığını belirlemek için de yapacaktır.

parseInt(i,n)

Fonksiyona dikkat ederseniz, 0, 4 ve 9'ncu karakterlerdeki açılan ve kapanan parantez ve kesme çizgisi irdelemelerinde başa dönmeyen döngülerde, Javascript en sonda if ile başlayan parseInt() şeklinde bir ifadeyi icra ediyor.

Javascript, kendisine alfanümerik olarak verilen bir değişken değerini tamsayı (integer) veya kesirli sayıya (float) çevirebilir. Alfanümerik değerleri tam veya kesirli sayıya çevirmek için, parseInt ve parseFloat ifadelerini kullanırız. (Javascript'in sayı olarak verilen değişkeni alfanümerik değere (string) çevirmesi için kullanılmaya hazır bir metod yoktur; bunun için toplama işlemi içeren bir fonksiyon yazmanız gerekir.) Her iki metotta da, çevirme işleminin 10 tabanlı (decimal), 8 tabanlı (octal) veya 16 tabanlı (hexadecimal) olmasını istediğinizi, çevrilmesini istediğiniz alfanümerik değerden (veya bu değeri temsil eden değişkenin adından) sonra bir virgül koyarak, 10, 8 veya 16 rakamı ile belirtmeniz gerekir.

parseInt() metodu ile bazı çevirme işlemlerine örnek verirsek, parseInt("123456.3333"), 123456 sonucunu verir. Çünkü bu metod kendisine verdiğiniz alfanümerik değerin içinde rakam ile artı ve eksi işaretlerinden başka bir şey gördüğü noktada durur; o noktaya kadar gördüğü unsurları bir tamsayı değişkeni olarak size geri verir.

parseFloat() metoduna da bir örnek verirsek, parseFloat("123456.3333"), 123456.3333 sonucunu verir. Çünkü bu metod kendisine verdiğiniz alfanümerik değerin içinde rakam ile artı ve eksi işaretlerinin yanı sıra nokta işaretinden başka bir şey gördüğü noktada durur; o noktaya kadar gördüğü unsurları bir kesirli sayı değişkeni olarak size geri verir.

Her iki metod da, çevirme işlemini başarıyla yaparlarsa, kendilerini görevlendiren işleme (bizim örnekte bunu bir if döngüsü yapıyor) 1, başarıyla yapamazlarsa 0 değeri döndürürler. Yani, if döngüsü, parsInt() metodu başarılı olursa 1, başarılı olamazsa 0 sonucunu alacaktır. Biliyoruz ki, if açısından 1, true-doğru, 0 ise false-yanlış demektir.

Buradaki örnek kodda, 1, 2, 3, 5, 6, 7, 8, 10, 11, 12 ve 13'ncü karakterler, parseInt() metodu ile sayıya çevriliyor ve elde edilen sayının 0 ile 9 arasında olup olmadığına bakılıyor; sonuç doğru ise döngü bir sonraki karakteri almak üzere başa dönüyor; sonuç yanlış ise

döngü o naktada break ile kesiliyor, durum değişkenin değeri "false" yapılıyor ve fonksiyon bu sonucu kendisini görevlendiren işleme teslim ederek, kapanıyor.

Fonksiyon ile HTML unsarlarının ilişkisi

Javascript açısından, yukarıdaki gibi uzun fonksiyon ile tek satırlık bir metod'un farkı yoktur: ikisi de kendisinden beklenen bir değeri, bu değeri bekleyen işleme teslim etmek zorundadır. Bunu ister 150 satırlık komut kümesiyle yapsınlar, isterse tek satırlık bir komutla, Javascript, isteyen bütün fonksiyonlara ve metodlara kendi nesnelerini ve onların metodlarını kullanma imkanı verir; hatta bilgisayarın kaynaklarından yararlanmalarını bile sağlar.

Böylece Javascript açısından Browser ve onun görüntülediği HTML belgesinin nesne olarak özelliklerine ve bilgisayarın kaynaklarını kullanmaya biraz daha yaklaşmış oluyoruz.

Şimdi yeni bir fonksiyon yazarak, Browser'ın, bilgisayarın işletim sisteminden saat ve tarih bilgisi edinerek, Javascript'e ulaştırmasını sağlayalım. Bunun için düz yazı programınızda şu kodu yazın ve fonksiyon2.htm adıyla kaydedin.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te Fonksiyon</TITLE>
<SCRIPT LANGUAGE="Javascript1.2">
<!--Javascript kodunu eski sürüm Browserlardan saklayalim
// fonksiyonu tanimlayalim:
function tarihsaat()
      {
var saat = new Date()
saat.getTime()
document.saatForm.saatkutusu.value = saat.toString()
kronometre = setTimeout("tarihsaat()",1000)
      }
//-->
```

```
</SCRIPT>
</HEAD>
<BODY onLoad="tarihsaat()">
<FORM method="POST" name="saatForm">
Şu anda saat: <input type=text size=50 maxlength=50 name="saatkutusu">
</FORM>
</BODY>
</HTML>
```

Bu Javascript kodunda tanımladığımız değişkenin, Javascript'in otomatik bir metodundan değişken elde ettiğini görüyorsunuz.

Date(): getYear(), getMonth(), getDate(), getDay(), getTime(), getHours(), getMinutes(), getSeconds()

Bu metod, Javascript'in Browser'dan, Browser'ın işletim sisteminden, işletim sisteminin de bilgisayarın temel girdi/çıktı işlemlerini yapan (BIOS) çipinden, saati ve tarihi içeren bilgiyi almasını sağlar. Gerçi metod Date (günün tarihi) adını taşıyor, ama Date(), Javascript açısından class (sınıf) sayılır ve edinilen bilgi sadece ay, gün ve yıl bilgisini değil, o andaki saati, dakikayı ve saniyeyi de içerir.

"saat" adını verdiğimiz bu değişken, Date() metodunun sağladığı bütün bilgileri içerdiği için, içinden saate ilişkin bilgiyi söküp almamız gerekir. Date() sınıfının sağladığı bilgiyi, get... yoluyla alırız.

Burada önce biraz, hem İngilizce'den hem de Javascript dilinden Türkçe'ye tercüme yapalım:

Year: Yıl (1900'den sonra)

Month: Ay (0=Ocak - 11=Aralık)

Date: Gün (1-31)

Hours: Saat (0-23)

Minutes: Dakika (0-59)

Seconds: Saniye (0-59)

Time: Zaman (Bu bilgilerin tümü: Gün, Ayın Adı, günün tarihi, saat:dakika:saniye, Yaz saati/kış saati ve yıl)

Bu durumda getYear(), size bilgisayarın BIOS'nun ayarlı oluduğu yılı (örneğin 1999) değerini; getHours() ise bu komutun icra edildiği andaki saati (örneğin, 19) kazandıracaktır. Gördüğünüz gibi, getTime() metodu, bize olağanüstü zengin bilgi kazandırabilir.

Bizim örnek kodumuzda, "saat" değişkeninin içinden getTime() yöntemi ile, hemen her türlü tarih ve saatle ilgili bilgi demetini alıyoruz; ve bunu bir başka metod'un işlemine tabi tutuyoruz.

toString(), toLowerCase(), toUpperCase()

Javascript'in kullanılmaya hazır işlevleri (metodları) arasında Date (Tarih) nesnesine uygulanabilecek olanlarından biri toString() metodudur. Kelime anlamı String'e çevir, alfanümerik'e çevir olan bu metodla, saat nesnesinin tamamen kendine özgü biçimi, Javascript tarafından HTML'in anlayabileceği şekle çevrilmiş olur. Buradaki örnek kodda, "saat" değişkeninin içeriğini toString() metoduna tabi tutarak, alfanümerik değere çeviriyoruz ve bunu, HTML sayfasındaki Form nesnesinin "saatkutusu" isimli INPUT ögesinin içeriği haline getiriyoruz.

Javascript'te, çevirme amaçlı iki metod daha vardır. toString() metodu sadece Date nesnesine uygulanabilir, ama toLowerCase() (küçük harfe çevir) ve toUpperCase() (büyük harfe çevir) metodları bütün alfanümerik değerlere uygulanabilir. Nesneleri daha ayrıntılı ele aldığımız bölümde, alfanümerik değerlere uygulanabilecek, kullanılmaya hazır fonksiyonlar diyebileceğimiz başka metodlar bulunduğunu da göreceğiz ve hepsini birarada inceleyeceğiz.

setTimeout("fonksiyonun_adı", milisaniye)

Örnek fonksiyonumuza dönersek; çevirme ve bulunan alfanümerik değeri HTML'in Form nesnesinin bir ögesinin içeriği olarak atama işleminden sonra, "kronometre" adını

verdiğimiz bir değişken görüyoruz; fakat bu değişkenin değeri oldukça farklı bir ifade içeriyor: setTimeout().

setTimeout(), Javascript'in 1.2 sürümüyle gelen ve kendi kendisinin zamanlamasını belirleyen bir metoddur. Buradaki örnekte, "kronometre" değişkeninin değeri olarak, "tarihsaat" fonksiyonumuzun, her 1000 milisaniyede bir yenilenmesini istiyoruz. Bunun sonucu, Javascript, fonksiyonu her bin milisaniyede bir çalıştıracak, ve bunun sonucu olarak HTML sayfasındaki formda saatkutusu isimli kutunun değeri sürekli yenilenecektir.

Bu metoddan yararlanarak, Javascript'e, sözgelimi her sabah 6:30'da bir iş yaptırabiliriz. Bunun için, söz gelimi setTimeout metodunu içeren değişkeni, şöyle bir döngü içine koyabiliriz:

```
      if ((saat.getHours() == 6) && (saat.getMinutes() == 30))
            {
//burada ne gibi işler yapılacaksa o işe ait komutlar olabilir, örneğin:
      document.saatForm.saatkutusu.value = saat.toString()
//sonra setTimeout metodunu içeren zamanlama ifadesi gelir:
      kronometre = setTimeout("tarihsaat()",10000)
            }
```

Böyle bir döngü içine alındığı zaman kronometre sadece sabah 6:30'da saati ve tarihi gösterecektir.

Değişkenleri Değerlendirme (Eval)

Buradaki örneğimizde yer almamakla birlikte bu bölümü bitirmeden bir metoddran daha söz edelyim. Javascript'in kendi hazır fonksiyonları içinde en az kullanılanı hangisidir diye sorarlarsa Değerlendirme Fonksiyonu olan eval()'dir derseniz, yanılmamış olursunuz. Buna karşılık, Javascript'in hemen hemen en kullanışlı fonksiyonlarından biri de eval() fonksiyonudur.

Bir kere, eval(), programcıyı satırlarca kod yazmaktan kurtarır. Diyelim ki, Javascript programınızın bir yerinde

```
var toplamTutar = "((siparisMiktarı * fiyatTL) * vergi)"
if (vergiYok) {
toplamTutar = "(siparisMiktarı * fiyatTL)"
      }
.......
```

şeklinde bir kodunuz var. Daha sonra programın herhangi bir yerinde,

```
document.write(eval(toplamTutar))
```

demekle, bütün bu üç-dört satırlık kodu yazmış olursunuz. eval() sizin için değişkenlerin değerlerini bulup, sonucu değerlendirip, arzu ettiğiniz yere yazacak veya iletecektir. Bu fonksiyondan ilerde yararlanacağız.

Javascript'in Nesneleri, Olayları ve Özellikleri

Bilgisayar programcılığı da giyim-kuşam dünyası gibidir: modayı takip eder. Günümüzde programcılığın en moda unsuru Nesneye Yönelik programcılıktır dersek, gerçeği ifade etmiş oluruz.

Javascript programcılığında nesne (object), ve nesnenin özellikleri (properties), genellikle HTML belgesinin adı (name) ve değeri (value) olan herşeydir. Bir HTML unsurunun etiketinde NAME ve VALUE bölümleri varsa, bu unsur, Javascript için nesne sayılır.

Bu tanıma göre Form, Javascript için bir nesnedir. Ayrıca Form nesnesinin bir ögesi olan INPUT, kendisi de ad ve değer alabildiğine göre, Javascript için bir nesne sayılır; fakat bu nesneye daima içinde bulunduğu nesne "dolayısıyla" atıfta bulunabilirsiniz. Bu tür atıflarda bulunurken, şu kurala uymanız gerekir:

nesneAdı.özellikAdı

Bir nesnenin durumunu, değerini veya bir özelliğini değiştiren Javascript'in kendi içinde kullanılmaya hazır işlevleri; tarih gibi, Browser'ın masaüstündeki penceresinin bir özelliği gibi değerleri belirleyen otomatik işlevleri; nesnelerin değerlerini belirli bir düzen içinde arttıran veya azaltan süreçleri; ve Javascript'in hazır şablonlarından yeni bir nesne üreten işlemleri, metod adı altında toplarız. Her nesnenin kendine ait bir metodu olabilir; bir metod birden fazla nesne ile birlikte kullanılabilir. Bu gibi ifadeleri şöyle yazarız:

nesneAdı.metodAdı (argüman)

Javascript ve niteliklerini etkilediği HTML, bir işletim sistemi ortamında, Grafik Kullanıcı Arayüzü (GUI) ile çalışan bir Browser programının içinde yer alırlar. Browser programları kendiliklerinden veya GUI sonucu, öyle bazı olaylara (örneğin Mouse

işaretçisinin bir nesnenin üzerine gelmesi veya bilgisayar kullanıcısının Mouse'un veya klavyenin bir düğmesini tıklaması gibi) yol açarlar ki, bu olay işletim sistemi-GUI-Browser yoluyla HTML belgesi (ve dolayısıyla Javascript) açısından önem taşıyabilir. Bunlara Event (olay) denir, ve Javascript'e bu olayın olması halinde icra edilmek üzere özel emirler verilebilir. Bu tür komutların yazılmasında şu yöntem izlenir:

event="fonksiyon_veya_metod (argüman)"

Aslında yukarıda özetlediğimiz üç unsuru, nesne, nesne özelliği ve Browser olaylarını buraya kadar bir çok örnekte gördük ve kullandık. Bu bölümde, bu kitapçığın elverdiği ölçüde her üçünü ayrı ayrı ve ayrıntılı ele alacağız.

Nesneler

Javascript ile yazacağımız programlar, Netscape veya Internet Explorer programlarının belge nesne modeli (Document Object Model) denen kurallar içinde hareket etmek zorundadır. Daha yüksek programlama dillerine, örneğin C++, Delphi veya Java gibi dillerle program yazmışsanız, programcı olarak başka bir programın modeli ile sınırlı olmadığınızı, hatta işletim sisteminin programlar için sağladığı arayüzle (API) kendinizi sınırlı hissetmeyebileceğinizi bilirsiniz. Fakat Javascript dahil tüm Script dilleri, Browser'ın sunduğu hiyerarşik nesne modeli ile sınırlıdır. Aşağıda Netscape'in modelini görüyorsunuz. Gerçi Internet Explorer çok daha fazla nesneyi kontrol etmenize imkan veriyor; fakat IE'nin geniş alanına giren nesnelere yönelik Javascript programı, Netscape kullanan ziyaretçilerin bilgisayarında işleyemeyebilir:

<DOM.TIF>

Javascript'te "document" nesnesi kavramını öğrenirken, ilk kavranması gereken unsur, "container" (içinde tutan, kap) meselesidir. Hatırlayacağınız gibi, yukarıdan beri Javascript'e ekrana bir şey yazdıracağımız zaman, şuna benzer bir komut veriyoruz:

document.write("<H1>Merhaba Dünya!</H1>"

Burada kullandığımız write()metodu, "document" nesnesine, o da Browser'ın "window" (pencere) nesnesine aittir. "window" nesnesini zikretmiyoruz; çünkü zaten onun içindeyiz. Ama yeni bir Browser penceresi açtırmak istersek, bunu da açıkça belirtmemiz gerekir. Aşağıdaki kodu, düz yazı programınızda yazarak, yenipencere.htm adıyla kaydederseniz ve Browser'ınızda açarsanız, size yeni bir "window" nesnesi oluşturma imkanı verecektir:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>Javascript'te Yeni Pencere</TITLE>
<SCRIPT LANGUAGE = "Javascript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalım
var windowNesnesi
function yeniPencere()
      {
windowNesnesi = window.open("", "YeniPencere", "toolbar=0, width=300, height=150,
resizable=1")
windowNesnesi.document.write("<H2>İşte yeni Pencereniz.. Tebrikler!</H2>")
      }
function kapaPencere()
      {
windowNesnesi.close()
      }
// -->
</SCRIPT>
<BODY OnUnload="kapaPencere()">
<H1>Bu düğmeyi tıklarsanız, yeni pencere açılır.</H1>
<FORM>
<INPUT Type="button" Value="Yeni Pencere Aç" OnClick="yeniPencere()" </INPUT>
</FORM>
</BODY>
</HTML>
```

Burada "windowNesnesi" adlı değişkinin değerine dikkat ediyor musunuz: Browser'ın kendi içinde bulunan "window.open()" metodunu kullanarak, araç çubuğu bulunmayan, eni 300 yüksekliği 150 piksel ve ziyaretçi tarafından ölçüleri değiştirilebilen bir yeni Browser penceresi açıyor. Aslında bu noktada windowsNesnesi adlı değişken, bu nesnenin buradaki temsilcisi sayılır; yeni nesneye ne yaptırtmak isterseniz, burada, windowNesnesi değişkinini kullanarak yapabilirsiniz. Bir başka dikkat edilmesi gereken nokta, yeni Window nesnesini oluşturduktan sonra bir daha "window" nesne adını kullanmıyoruz, sadece document adlı container'in (kabın) adına referans yapıyoruz.

Yukarıdaki şemada gördüğümüz nesnelerin HTML karşılıkları şöyledir:

Javascript Nesnesi	HTML Karşılığı
Pencere (Window)	Yok
Çerçeve (Frame)	<FRAME>
Belge (Document)	<BODY>
Form	<FORM>
Düğme	<INPUT TYPE="button">
İşaretleme Kutusu	<INPUT TYPE="checkbox">
Gizli (Hidden)	<INPUT TYPE="hidden">
Dosya Çıkart (file upload)	<INPUT TYPE="file">
Parola (password)	<INPUT TYPE="password">
Radyo düğmesi (Radio)	<INPUT TYPE="radio">
Başa dön (Reset)	<INPUT TYPE="reset">
Seç (Select)	<SELECT>
Gönder (Submit)	<INPUT TYPE="submit">

Metin (Text)	<INPUT TYPE=”text”>
Metin Alanı (Textarea)	<TEXTAREA>
Bağ (Link)	<A HREF=””>
Bağlantı (Anchor)	<A NAME=””>
Applet	<APPLET>
Resim (Image)	<IMG>
Plugin	<EMBED>
Alan (Area)	<MAP>

Bu nesneleri ve metodları ile yanıt verdikleri Window olayları (event) konusuna geçmeden önce bir iki noktayı belirtelim:

Window nesnesinin HTML karşılığı bulunmamakla birlikte BODY etiketinin onLoad ve unonLoad olaylarında işleyen bir fonksiyonla yeni window nesnesi oluşturmanız (yukarıdaki örnekte olduğu gibi) daima mümkündür. Ayrıca, içinde bulunduğunuz veya yeni açmak istediğiniz bir Browser penceresinin alt çervesindeki Status (durum) mesajı alanına istediğiniz mesajı yazdırabilirsiniz:

window.status = ’Benim sayfama hoş geldiniz!’

Internet, HTML sayfası ve bağlı belgelerle bilgisayar dosyalarını alıp vermek demektir. Mahal (Location) nesnesi, Browser’ın görüntülediği belgenin nereden geldiğini gösterir. Gerçi, HTML size görüntülediği belgenin nereden geldiğini bildirmez, ama Browser, bunu “window.location.protocol” nesnesi olarak tutar. Bu değer ya “http:” ya da “file:” (ikisi de küçük harf ve sonlarında iki nokta üstüste var) olur. Örneğin:

```
if (window.location.protocol == “http:” {
alert (“Bu belge Internet’ten geliyor.”) }
Else { alert (“Bu belge sabit diskten geliyor.”)}
```

şeklinde bir kodla, belgenin mahallini araştırmak mümkündür.

Tarih (History) nesnesi ise, her iki Browser'ın, ziyaret edilen siteler ve sayfalar listesinde ileri veya geri gitmekte kullandığı nesnedir. HTML'de bu listeye erişmeyi sağlayan etiket bulunmamakla birlikte Javascript bu listeyi kullanabilir. Örneğin:

function gerigit() { window.history.go(-1) }

şeklindeki bir fonksiyon ve bu fonksiyonu çağıran bir düğme (onClick=gerigit() gibi) Tarin nesnesinden yararlanmasını sağlayabilir.

Olaylar

Browser programları, olaylı bir yaşama sahiptir! Ya bilgisayar sistemi, ya da kullanıcı (ziyaretçi), sürekli bir takım olayların (Event) oluşmasına sebep olurlar. Ya Mouse'un bir düğmesine basılır "Click" adı verilen olay oluşur; ya klavyede bir tuşa basılır, "KeyDown" olayı oluşur. Bir HTML sayfasının yüklenmesi biterse "Load," sayfadan çıkma işlemi tamamlanırsa "unLoad" olayı olur.

Her Pencere, Browser veya Kullanıcı olayına karşılık, Javascript'in bir Olay Yönlendiricisi (Event Handler) vardır. Olay Yönlendirici, olayın adını taşır. Örneğin, kullanıcının Mouse'un bir düğmesini tıklaması, Click, bu olayı karşılayan ve yönlendiren metod ise onClick (tıklandığında, tıklama halinde) adını taşır. Javascript programcısı olarak bu olayların tümü bizi ilgilendirmediği gibi, nesnelerimiz de çoğunlukla bu olaylara ilgi göstermezler. Bizi yakından ilgilendiren nesneler ve onların olaylarla ilgisini ..... tabloda görebilirsiniz.

//////////////NOT////////////////////////////////

Boş yere tablonun bulunduğu yeri belirten ifade gidecek:

karşı sayfadaki

önceki sayfadaki

arkadaki sayfada bulunan

gibi.

//

<EVENTS.TIF --- RESİMALTI: Javascript açısından Browser Nesneleri ve karşılık verdikleri Browser ve GUI Olayları (Event).>

Event Handler'ları, bir tür adlandırılmış ama içi boş fonksiyonlar gibi düşünebilirsiniz. Programcı olarak bize düşen, bu olay karşısında olay yöndendiricisinin ne yapmasını istediğimi belirtmekten ibarettir. Event Handler'lar, daha önceden hazırladığımız bir fonksiyonu göreve çağırabilir; veya hemen oracıkta bir Javascript metodunu da devreye sokabiliriz. Mesela:

```
<INPUT TYPE="text" SIZE=50 MAXLENGTH=100 NAME="soyadi" onChange="denetle(this)">
```

Ziyaretçi bu INPUT kutusuna soyadını yazdığında, kutunun içeriği değişmiş olacak ve bu Change (değişiklik) olayı, kendisini yöndendiren onChange sayesinde, önceden hazırladığımız "denetle()" isimli fonksiyonu çağıracaktır. Burada gördüğünüz "this" (bu) kelimesi, Javascript'e fonksiyonun istediği değer kümesi olarak bu nesnenin içeriğini vermesini bildiriyor. İlerde örneklerde bunu sık sık kullandığımızı göreceksiniz.

Şimdi sık kullandığımız olayları ve yönlendirilcilerini daha yakından tanıyalım:

onClick

Ziyaretçinin, Mosue işaretçisi tıklanabilir bir nesne üzerinde iken Mouse'nun düğmesine basması, Click olayını doğurur; bu olayı onClick yönlendirir. Düğmeler, işaretleme kutuları, radyo düğmeleri ve bağlantılar tıklanabilir. Olayın oluşması için varsayılan Mouse düğmesinin basılması ve bırakılması gerekir. Düğme basılı durursa, Click olayı gerçekleşmiş olmaz.

onClick yönlendiricisine, "Click" olayı halinde ne yapmasını istediğinizi ait olduğu nesnenin HTML etiketi içinde bildirmeniz gerekir. Aşağıdaki kodu düz yazı programınızda yazar ve onClick01.htm adıyla kaydederseniz, onClick yönlendiricisi için program yazmış olacaksınız:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>JavaScript'te onClick</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
//onClick icin fonksiyonlar
function resim1()
      {
resim1Pencere = window.open("resim1.htm", "Resim1", "toolbar=0, width=200,
heigth=400,resizeable=0");
      }
function resim2()
      {
resim2Pencere = window.open("resim2.htm", "Resim2", "toolbar=0, width=200,
heigth=400,resizeable=0");
      }
// -->
</SCRIPT>
</HEAD>
<BODY>
<H1>Düğmelerden birini tıklayınız!</H1>
<INPUT TYPE="button" NAME="Resim1" VALUE="Resim 1" onClick=resim1()>
<INPUT TYPE="button" NAME="Resim2" VALUE="Resim 2" onClick=resim2()>
</BODY>
</HTML>
```

Bu kodda adı geçen resim1.htm ve resim2.htm adını vereceğimiz ve iki küçük resim içeren sayfaları da oluşturalım (Elinizde hazır bir grafiğin adını verebilirsiniz).

```
<html>
```

```
<head>
<title>Resim 1[veya Resim 2]</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>
<body>
<h1>Resim 1[veya Resim 2]</h1>
<p><img src="resim1.gif" width="151" height="278" alt="uzayli.tif (4942 bytes)"></p>
</body>
</html>
```

Bu üç sayfayı oluşturduktan sonra onClick.htm'de sırayla her iki düğmeyi de tıklayın; her Click olayında onClick yönlendiricisinin farklı fonksiyonları çağırdığına dikkat edin.

Kimi zaman onClick yönlendiricisi, bağlantı nesnesi ile doğrudan kullanılabilir. Bağlantı nesnesi olan <A HREF=....> etiketi, bir bağlantı protokolü adresi (URL) verilmesini ister. Ancak HTTP veya FILE protokolü yerine buraya doğrudan JavaScript'in adını ve icra etmesi istenen fonksiyonu yazabilirsiniz. Örneğin, <A HREF="JavaScript:resim1()"> şeklinde bir ifade, protokol olarak Javascript'i göstermemizi sağlayabilir.

onClick01.htm'de INPUT etiketleri yerine, şu iki satırı yazarak, onClick02.htm adıyla kaydedip, Browser'da açın. Düğmelerin yerini bağlantılı iki kelimenin aldığını göreceksiniz; fakat sayfanın işlevi değişmeyecek. onClick, bu kez düğmenin Click olayını değil, bağlantının Click olayını karşılıyor:

```
<A HREF="JavaScript:resim1()">Resim 1</A>
<A HREF="JavaScript:resim1()">Resim 1</A>
```

onSubmit

Javascript'in, HTML'i tamamladığı noktaların başında Form ve Form yoluyla elde edilecek bilgilerle ilgili işlemler gelir. Bu işlemlerin başında ise Form'daki bilgilerin Server'a gönderilmeden önce, daha Browser'da iken doğrulanması ve hatalardan arındırılması vardır. Javascript, bunu ziyaretçinin formda, Submit (Gönder) olayına yol açan herhangi bir düğmenin tıklanması halinde yapar. Bir form, ziyaretçi tarafından türü Submit olan bir

düğmenin tıklanması halinde, Browser tarafından HTTP protokolü ile Server'a gönderilir. Javascript, onSubmit yönlendiricisi ile bu olaya ilişkin işler yapabilir, fonksiyonlar icra edebilir. Fakat burada dikkat edeceğiniz nokta, "Submit" olayına engel olmamak veya bu işlemi şarta bağladı iseniz, bu şartın yerine gelmesini sağlamaktır. Bunu bir örnek üzerinde görelim. Diyelim ki, HTML sayfanızda, şuna benzer bir FORM etiketi var:

```
<FORM ACTION="form_islem.pl" METHOD="post" onSubmit="onayla()">
```

Burada adı geçen "onayla()" fonksiyonu HTML'in HEAD bölümünde yer alıyor ve şuna benzer komutlar içeriyor:

```
//onSubmit olayını yönlendirme fonksiyonu
function onayla() {
confirm('Bu siparişi vermek istediğinizden emin misiniz?');
      }
```

İlk bakışta böyle bir "tuzak," formun bir kere daha onaylandıktan sonra gönderileceğini düşündürebilir. Burada gördüğünüz "confirm" komutu, Javascript'e, Browser'ın kullanıcıdan OK (Tamam) ve Cancel (İptal) yazılı bir kutu ile onay almasını sağlar. Kullanıcı OK düğmesini tıklarsa, Browser, Javascript'e "true" (doğru), kullanıcı Cancel (İptal) düğmesini tıklarsa "false" (yanlış) değeri gönderecektir; sanılabilir ki, Javascript'e "true" değeri dönerse formu gönderilecek, fakat "false" değeri dönerse form gitmeyecektir.

Bu doğru değil! Her iki değerde de form gidecektir; çünkü, yukarıdaki kodda Browser'ın Confirm kutusunun oluşturacağı değerin türü ile onSubmit olay yönlendiricisi arasında bir bağ kurulmuyor; başka bir deyişle onSumbit'e kendisine dönmesi gereken bir sonuç olduğu bildirilmiyor. Bunu "return" anahtar kelimesiyle sağlarız. "return" (anlamı: geri dön) kelimesi, bir fonksiyona kendisini göreve çağıran unsura mutlaka bir değer göndermesi gerektiğini bildirir. "return" kelimesini kullanarak, Form etiketini şöyle yazabiliriz:

```
<FORM ACTION="form_islem.pl" METHOD="post" onSubmit="return onayla()">
```

Dolayısıyla "onayla()" fonksiyonuna da kendisini göreve çağıran komuta bir sonuç ulaştırması gerektiğini bildtirmemiz gerekir:

```
//onSubmit olayını yönlendirme fonksiyonu
function onayla() {
return confirm('Bu siparişi vermek istediğinizden emin misiniz?');
      }
```

Bu durumda kullanıcı/ziyaretçi "confirm" kutusunun OK (Tamam) düğmesini tıklarsa onSubmit yönlendiricisi Form'u belirtilen CGI programına, işlenmek üzere, gönderecek; Cancel (İptal) düğmesi tıklanırsa, onSubmit yönlendiricisi submit olayını durduracaktır.

onReset

Formlara koyduğunuz Reset (Başa dön) düğmesi, bir Form'daki bütün bilgilerin silinmesini sağlar; Browser'ın Geri düğmesini tıklamak dahil, hiç bir eylem ziyaretçinin yanlışlıkla bastığı Sil düğmesinin sildiğini geri getiremez. Fakat programcı olarak siz böyle bir hatayı önleyebilirsiniz. Çünkü Sil düğmesinin oluşturduğu "reset" olayını yönlendiren onReset, size bu imkanı verir. Tıpkı onSubmit'teki gibi FORM etiketinize onReset tarafından devreye sokulacak bir doğrulama fonksiyonu koyabilirsiniz. Form etiketiniz şöyle olabilir:

```
<FORM ACTION="form_islem.pl" METHOD="post" onReset="return sahiMi()">
```

Bu fonksiyonu da HTML'in HEAD bölümünde şöyle yazabilirsiniz:

```
//onReset olayını yönlendirme fonksiyonu
function sahiMi() {
return confirm('Bu formdaki büktün bilgileri silmek istediğinizden emin misiniz?');
      }
```

Ziyaretçiniz doldurduğu formda, Gönder yerine yanlışlıkla Sil'i tıkladıysa, böylece bir şansa daha kavuşmuş olur.

onChange

HTML sayfanızda olabilecek tek değişiklik, ziyaretçinin içine bir şeyler yazabildiği veya bazı işaretler koyabildiği üç alandır: Text (Metin), Textarea (Metinalanı) ve Select (Seç) etiketleri. Metin ve metinalanın içini tıklayarak Browser Penceresi'nin odak noktasını buraya getiren ziyaretçi klavyesine basarak, bu alanlara metin girebilir; bu HTML sayfasında

"change" (değişiklik) olması anlamına gelir Select etiketinin sağladığı şıklardan birini seçen ziyaretçi de sayfada değişiklik olayına yol açar. "change" olayı halinde, onChange bu olayı yönlendirir. Javascript programcısı olarak siz bu yönlendiriciye belirli bir fonksiyonu çalıştırmasını bildirebilirsiniz.

Bunu da bir örnekle ele alalım. Diyelim ki Frame (Çerçeve) esasına dayanan bir ana sayfanız var; ziyaretçinin bir çerçevedeki "select" etiketinin seçeneklerinden birini seçmesi halinde bir diğer çerçevenin içeriği değişsin istiyorsunuz. Ana sayfanızda bir FRAMESET var ve bu üç çerçeveli bir sayfa oluşturuyor. Burada Browser'ın "document" (sayfa) hiyerarşisi içinde Frameset "Parent" (ebeveyn; çünkü diğer bütün Frame'leri o doğuruyor!) ve "Frame 0" (1'nci çerçeve) "Frame 1" (2'nci çerçeve) ve "Frame 2" (3'ncü çerçeve) vardır. "document" nesnesi çerçeveleri bir dizi-değişken içinde tuttuğu ve Array (dizi değişken) numaralanmaya 0'dan başladığı için çerçevelerimiz 0'dan itibaren numaralanıyor. Buna göre sağ Frame'in içeriğini değiştireceğimiz zaman bu çerçeveye "Frame 2" adıyla gönderme yapmalıyız. Frame 1'e şöyle bir FORM koyarsak, daha sonra bu formun değişiklik olayını onChange ile sağ çerçevenin içeriğini değiştirmeye yönlendirebiliriz:

```
<FORM METHOD="post">
<P><SELECT NAME="hedef" SIZE="1"
onChange="parent.frames[2].location.href=this.form.hedef.options[this.form.hedef.selected
Index].value">
<OPTION>Seç bakalım
<OPTION VALUE="test1.htm">Test 1
<OPTION VALUE="test2.htm">Test 2
<OPTION VALUE="test3.htm">Test 3
</SELECT>
</FORM>
```

Burada, SELECT nesnesinin içeriği, ziyaretçinin herhangi bir seçeneği seçmesi halinde değişmiş oluyor; bu "change" olayı üzerine bu olayı yönlendirmek üzere kullandığımız onChange ise karşısındaki Javascript komutunu icra etmeye başlıyor. Bu uzun kodda, "this" (bu) anahtar kelimesinin yeniden kullanıldığını görüyorsunuz. Bu kelime, herhangi bir

Javascript fonksiyonuna (adı fonksiyon olmamakla birlikte burada yazdığımız kod da bir fonksiyonun görevini yapıyor) ihtiyacı olan bilgileri bu nesneden almasını bildiriyor. "Bu" diye nitelediğimiz nesne, içinde bulunduğumuz FORM'un SELECT ögesidir. SELECT ögesinin OPTION'ları, bu nesnenin OPTIONS dizi-değişkeninde numaralanır ve seçilen OPTION'ın numarası "selectedIndex," içeriği ise ".value" adıyla adlandırılır. Nesneleri ve özelliklerini ele aldığımız bölümde bu konuya yeniden döneceğiz.

onFocus, onBlur

Bilgisayarda bir program açıkken, Mouse ile programın başlık çubuğunu tıklayın; çubuğun renginden işletim sisteminin dikkatinin bu programın penceresinde yoğunlaştığını göreceksiniz. Bir de masaüstünde boş bir yeri tıklayın; bu pencere sistemin odak noktası olmaktan çıkacaktır. HTML sayfasında da Browser'ın Focus'unu (dikkat odağını) üzerinde topladığı veya odağın çekildiği üç nesne olabilir: Text (Metin), Textarea (Metinalanı) ve Select (Seç) etiketleri. Çünkü ziyaretçinin sadece bu etiketlerin oluşturduğu nesnelere klavyenin dikkatini çekme hakkı vardır. Bu nesnelerden biri tıklanınca Browser'ın focus'u bu nesneye dönmüş, yani "focus" olayı olmuş demektir; bu durumda, biz de bu olayı yönlendiren onFocus'u kullanabiliriz. Aynı mantıkla, ziyaretçi Browser'ın focus'unu bu nesneden başka bir yere çektiği zaman bu nesne focus'u kaybeder, "blur" (netlikten çıkma, flulaşma) olayı olur. Bunu da onBlur ile yönlendiririz. Bu iki yönlendiricinin birlikte kullanıldığı şu örnek kodu, onFocus.htm adıyla kaydederseniz ve Browser'da ekrandaki talimatları izlerseniz, her iki yönlendiriciyi de kullanmış olacaksınız:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>JavaScript'te onFocus ve onBlur</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// fonksiyonları tanimlayalim
function goster()
```

```
      {
document.isimForm.ad.value="Adınız kayda geçmiştir."
      }
function uyar()
      {
document.isimForm.ad.value="Lütfen yanlışlık yapmayınız!"
      }
// -->
</SCRIPT>
</HEAD>
<BODY>
<BR>
<FORM NAME="isimForm">
<P>Lütfen adınızı yazın ve sayfa üzerinde başka bir yeri tıklayın:
<BR><INPUT TYPE="text" NAME="ad" VALUE="Adınızı buraya yazınız" SIZE=25
onBlur="goster()">
<P>Şimdi de aşağıdaki kutuya mesleğinizi yazın:
<BR><INPUT TYPE="text" NAME="meslek" VALUE="Mesleğinizi buraya yazınız" SIZE=25
onFocus="uyar()">
</FORM>
</BODY>
</HTML>
```

Javascript, üstteki kutuya adınızı yazdıktan sonra başka bir yeri tıkladığınızda olan iş, "ad" isimli metin kutusunun "blur" olayına onBlur'ün çağırdığı goster() fonksiyonu ile; alttaki kutuyu mesleğinizi yazmak üzere tıkladığınız anda "meslek" kutusunda olan "focus" olayına da uyar() fonksiyonu ile karşılık veriyor.

onMouseOver, onMouseOut

Javascript dünyasında en çok kullanılan iki "event" varsa, birincisi Mouse işaretçisinin bir nesnenin üzerine gelmesi ("MouseOver" olayı), ikincisi ise Mouse iuşaretçisinin bir nesnenin üzerinden çekilmesi ("MouseOut" olayı) sayılabilir. Bu iki olayı onMouseOver ve onMouseOut yönlendiricileri ile karşılayabiliriz. Gerçekte bu iki olayla, Mouse işaretçisi bir bağlantı satırı veya grafiğinin ve bir grafik haritasının (Map, Area) üzerine geldiği veya

üzerinden çekildiği zaman ilgilenebiliriz; çünkü diğer HTML unsurları ve Javascript nesneleri ile Mouse işaretçisinin işaretleme ilişkisi yoktur.

Bu olayı ve yönlendirilmesini, şu küçük programda görebiliriz:

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>JavaScript'te Fonksiyon</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// fonksiyonu tanimlayalim:
function faregelince()
      {
       window.status = "En güzel site için burayı tıklayın!";
       return true
      }
function faregidince()
      {
       window.status = "En güzel siteye gitme fırsatını kaçırdınız!"
       return true
      }
// -->
</SCRIPT>
</HEAD>
<BODY>
<A HREF="http://pcworld.com.tr" NAME="bag" onMouseOver="return faregelince()"
onMouseOut="return faregidince()">PC World-Türkiye</A>
</BODY>
</HTML>
```

Bu kodu onMouseOver.htm adıyla kaydeder ve Mouse işaretçisini PCWORLD-TÜRKİYE bağlantısının üzerine getirirseniz, Browser'ınızın alt çerçevesindeki statü mesajı yerinde "En güzel site için burayı tıklayın!", geri çektiğinizde de "En güzel siteye gitme fırsatını kaçırdınız!" mesajını göreceksiniz.

onLoad, onUnLoad

Javascript açısından Browser'ın bir HTML sayfasını görüntülemeyi bitirmesi sayfanın "load" (yükleme), başka bir sayfaya geçilmesi ise önceki sayfanın "unload" (yüklenmişlikten çıkması) sayılır; ve bu iki olayı onLoad ve onUnLoad yönlendiricileri ile karşılayabiliriz. HTML dosyasının iki bölüm (HEAD ve BODY) olduğunu biliyorsunuz. Browser açısından asıl sayfa, BODY bölümüdür; HEAD bölümündeki komutlar sayfanın nasıl görüntüleneceğine ilişkin komutlar içerir.

Bir sayfa Browser'a yüklenirken yapabileceğimiz bir çok işten birisi, ziyaretçinin Browser'ının türünü yoklamak olabilir. O halde, BODY etiketini şöyle yazarsak;

```
<BODY onLoad="hangiBrowser()">
```

ve HTML'in HEAD bölümüne bu fonksiyonu koyarsak, sayfamız yüklenirken bize ziyaretçinin Netcape mi, yoksa Internet Explorer mı kullandığını bildirebilir. Bu bilgiyi nasıl değerlendirebileceğimiz ise ayrı bir konu. "hangiBrowser()" fonksiyonu şöyle olabilir:

```
// hangiBrowser fonksiyonunu tanımlayalım:
var Browser=navigator.appName + " " + navigator.appVersion;
var getkey=Browser.substring(0, 12);
function hangiBrowser()
      {
if (Browser.substring(0, 8)=="Netscape")
       window.location="netscape4.htm";
if (Browser.substring(0, 9)=="Microsoft")
       window.location="ie.htm";
if ( (Browser.substring(0, 8)!="Netscape") && (Browser.substring(0, 9)!="Microsoft") )
       window.location="belirsiz.htm";
      }
// -->
```

Bu iki kodu içeren sayfanızı onLoad.htm adıyla kaydeder ve aynı dizine, içinde "Netscape kullanıyorsunuz! yazılı ve adı "netscape.htm"; içinde "IE kullanıyorsunuz!" yazılı ve adı "ie.htm", ve içinde "Ne tür Browser kullandığınızı anlamadık!" yazılı ve adı

"belirsiz.htm" olan üç HTML dosyası koyarsanız, onLoad.htm'i açmak istediğinizde kullandığınız Browser hangisi ise (onLoad.htm'in değil) diğer üç dosyadan birinin açıldığını göreceksiniz.

Bir sayfada başka bir sayfayı işaret eden bir bağlantıyı tıkladığınızda, Browser açık olan HTML kodunu silecek, yani sayfayı yüklenmemiş hale getirecektir. Bu Javascript açısından bir UnLoad olayıdır. O halde, BODY etiketimize onUnLoad yönlendiricisini harekete geçiren bir fonksiyonu çağırmasını söyleyebiliriz.

Aşağıdaki HTML konudu onUnLoad.htm, ve bir HTML sayfasına da sadece "Güle Güle.. Yine Bekleriz!" yazarak, gulegule.htm adıyla kaydederseniz sayfanız kapanırken ziyaretçinize veda etme imkanınız olur. (Ama çoğu ziyaretçinin bundan hoşlanmayacağına emin olabilirsiniz!)

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<TITLE>JavaScript'te onUnLoad</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// gulegule fonksiyonunu tanimlayalim:
function gulegule()
      {
gulegulePencere = window.open("gulegule.htm", "GuleGule", "toolbar=0, width=200,
heigth=400,resizeable=0");
}// -->
</script>
</head>
<body onUnLoad="gulegule()">
Bu sayfa size güle güle diyecek!!
<A HREF="test1.htm">Baska sayfa</A>
</body>
</html>
```

Buradaki BODY etiketi, Browser başka sayfaya geçmek üzere mevcut sayfayı sildiği anda, Browser'a yeni bir pencere açarak, bu pencerede gulegule.htm'i görüntülemesini sağlayacaktır.

onError, onAbort

Ziyaretçinin Browser'ı açmak üzere olduğu sayfayı transfer hatası yüzünden açamaz, HTML veya Javascript kodunu doğru yorumlayamazsa, Error (hata) olayı oluşur. Javascript, sadece kendisine ilişkin hata durumlarını onError ile yakalayabilir.

Javascript'in Browser penceresiyle ilgili hatasını yakalamak istediğiniz kod, HTML'in HEAD bölümünde, SCRIPT etiketleri içinde ve "window.onerror" kelimeleri küçük harfle yazılmış olarak bulunmalıdır:

```
window.onerror=hataHalinde()
        function hataHalinde (message, url, lineNumber)
        {
                    messageList = new Array()
                    urlList = new Array()
                    lineNumberList = new Array()
                    messageList[messageList.length] = message
                    urlList[urlList.length] = url
                    lineNumberList[lineNumberList.length] = lineNumber
                    alert(messageList[0] +
                    "\ndosya adı " + urlList[0] +
                    "\nsatır sayisi " + lineNumberList[0])
        return true
        }
```

Bu kodu üzerinde çalıştığınız Javascript kodu içeren bir HTML sayfasına yerleştirdiğiniz taktirde olabilecek hatalar bir Uyarı kutusuyla size bildirilir. Sayfanın ve Javascript'in hatasız çalıştığını anladıktan sonra bu kodu kaldırabilirsiniz.

Javascript, sayfalarınızdaki grafik unsurların da hata halini yakalayabilir. HTTP iletişiminde, en sık görülen transfer hatası grafik unsurların aktarılması sırasında olur; ayrıca

HTML tasarımı sırasında grafik dosyalarının sayfaya veya sitenin kök dizinine göre göreli adresinin yazılmasında sık yapılan hatalar, grafik unsurların aktarılmasını engeller. Bu tür hatalarda ziyaretçinizi şu Javascript koduyla uyararabilirsiniz:

```
<IMG NAME=”resim01” SCR=”resim1.gif” orError=”alert(‘resim1.gif aktarılamıyor; özür dileriz’)”>
```

Sayfalarınıza koyduğunuz grafik unsurlar ziyaretçilerinizin sabrını taşırır da, grafiklerin sonuna kadar aktarılmasını beklemeden Browser’ın Stop düğmesini tıklarlarsa, bunun sayfanız için kötü sonuçları olabilir. Grafik unsurlara bağlantı belirten alan haritaları (AREA, MAP) etiketleri koymuş olabilirsiniz. Ziyaretçinin Browser’ının Stop düğmesini tıklaması, Javascript açısından abort (yarıda kes) olayının doğması demektir; bunu onAbort ile yönlendirebilirsiniz. Bu yönlendiriciden yararlanarak ziyaretçiye sayfanın eksik olduğunu bildirmeniz yerinde olur. Bunun için IMG etiketini şöyle yazabilirsiniz:

```
<IMG NAME=”resim01” SCR=”resim1.gif” onAbort=”alert(‘Sayfanının bütün unsurları aktarılamadı; beklediğiniz sonucu alabilmek için bütün unsurların aktarılması yerinde olur’)”>
```

Nesneler ve Olayları Birleştirelim

Şimdi Javascript açısından Browser nesnelerinin neler olduğunu biliyoruz; Browser’ın Javascript açısından ne gibi olaylar oluşturduğunu ve bu olayları nasıl kullandığımızı küçük örneklerle gördük. Olaylardan yararlanarak Javascript’le nesneleri etkileyen programlar yazabiliriz; fakat bunun için nesnelerin ne gibi özellikleri olduğunu ve bunları nasıl etkileyebileceğimizi de bilmemiz gerekir. Bu bölümde nesnelerin Javascript ile etkilenebilir özelliklerinin neler olduğunu ele alacağız.

Nesnelerin ayrıntılarına girmeden önce bir noktaya açıklık getirelim: Javascript ile bir HTML unsurunu etkileyebilmek, örneğin bir çerçevenin (Frame) içeriğini değiştirebilmek için mutlaka o nesnenin bir olaya karşılık verir olması gerekmez. Çerçeveler HTML etiketleri ile şekillenirler; herhangi bir olaya karşılık vermezler. Dolayısıyla olaylara yanıt vermeyen

nesneleri, olaylara yanıt veren nesneler yoluyla etkileyebiliriz; bunun için de bir çok HTML nesnesinin özellikleri bilmemiz gerekir.

Bu açıdan bakınca, olaylara yanıt versin-vermesin, HTML nesnelerimizi, Browser Nesneleri, Belge Nesneleri, Form Nesneleri ve Javascript Dil nesneleri olarak dört ana gruba ayıracağız.

Browser Nesneleri

Browser'ın Javascript dilinde adı Navigator'dır. Ziyaretçinin Browser programı ister Netscape Navigator olsun, isten Internet Explorer olsun, Javascript için Navigator'dır; nesnelerine de Navigator Nesneleri denir. Bunların başında, Browser'ın pencereleri ve Çerçeveler gelir.

Pencere (window)

Şu ana kadar yazdığımız bir çok Javascript kodunda Navigator penceresinin bazı özelliklerini kullandık: width (genişlik), height (yükseklik), toolbar (araç çubuğu) gibi. Şimdi bu özellikleri toplu olarak sıralayalım:

width	Navigator'ın sayfanın görüntülenmesi için ekrandaki temiz alanının piksel olarak genişliği.
height	Navigator'ın sayfanın görüntülenmesi için ekrandaki temiz alanının piksel olarak yüksekliği.
toolbar	Navigator'ın araç çubuğunun gösterilmesi (=1) veya gizlenmesi (=0).
menubar	Navigator'ın menü çubuğunun gösterilmesi (=1) veya gizlenmesi (=0).
scrollbars	Navigator'ın sağ ve alt kenardaki kaydırma çubuklarının gösterilmesi (=1) veya gizlenmesi (=0).
resizable	Navigator penceresinin büyütülebilir-küçültülebilir olması (=1) veya olmaması (=0).

status	Navigator penceresinin alt kenarındaki mesaj çubuğunun gösterilmesi (=1) veya gizlenmesi (=0).
location	Navigator penceresinin URL adres çubunun gösterilmesi (=1) veya gizlenmesi (=0).
directories	Netscape'de ikinci araç çubunun gösterilmesi (=1) veya gizlenmesi (=0).
copyhistory	Mevcut Navigator penceresinin history kaydının (daha önce ziyaret edilen URI adreslerinin tutulduğu çizelgenin) yeni pencereye de kopyalanması.
outerWidth	Navigator penceresinin piksel olarak genişliği.
outerHeight	Navigator penceresinin piksel olarak yüksekliği.
left	Navigator penceresinin ekranın sol kenarından piksel olarak uzaklığı.
top	Navigator penceresinin ekranın üst kenarından piksel olarak uzaklığı.
alwaysRaised	Navigator penceresinin masaüstünde açık bütün pencerelerin üzerinde kalmasını sağlar. (Sadece Windows ve MacOS'de işler.)
z-lock	Navigator penceresinin içi tıklansa bile masaüstünde açık bütün pencerelerin altında kalmasını sağlar. (Sadece Windows ve MacOS'de işler.)

(Evet-Hayır anlamına değer alan özellikleri 1 ve 0 ile belirtebileceğiniz gibi yes (evet) ve no (hayır) ile de belirtebilirsiniz.)

Javascript'in yeni Navigator penceresi açma metodu, window.open() şeklinde yazılır:

```
yeniPencere = window.open("yenisayfa.htm", "YeniPencereninAdı", "toolbar=1,
menubar=yes, resizable=no")
```

Burada, yeni pencereye vermek istediğiniz adın tek kelime olduğuna, özelliklerin bir çift-tırnak içinde toplandığına dikkat edin.

Navigator pencereleri, xxxx.close() metodu ile kapatılır. xxxx yerine açık pencerenin adını yazmanız gerekir. Aksi taktirde Javascript komutunu icra ettiğiniz pencerenin kendisi (yani Browser) kapanır. (Bazı Browser'ların bazı sürümleri Javascript tarafından açılmamış Navigator pencerelerinin Javascript ile kapatılmasına izin vermezler.)

Bu nesnenin ve her iki metodun da kullanımını daha önce gördük. Mevcut bir HTML belgesinin adını vermeden, yeni pencereyi açtırıp, içine Belge nesnesini kullanarak HTML komutları yazdırmayı, aşağıda Belge Nesnesi bölümünde ele alacağız.

Bu arada, Browser'ın kendisinin bazı özellikleri ile Javascript açısından nesne olduğunu tekrarlayalım. Bu açıdan Browser'ın Javascript için şu özellikleri vardır:

appname	Browser'ın adı
appVersion	Browser'ın sürümü
appCodeName	Browser'ın kod adı
userAgent	Browser'ın Server'a kendisini tanıtırken verdiği isim.

Şu kısa kodu browser.htm adıyla kaydeder ve farklı Browser'larla açarsanız, Netscape ve IE hakkında bilgileri bir kutu içinde görebilirsiniz:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
function siziTaniyalim () {
       var BrowserAdi = ""
       BrowserAdi += "Browser: " + navigator.appName + "\r"
       BrowserAdi += "Sürümü:" + navigator.appVersion + "\r"
       BrowserAdi += "Kod Adı: " + navigator.appCodeName + "\r"
       BrowserAdı += "userAgent: " + navigator.userAgent + "\r"
       alert(BrowserAdi)
```

```
      }
</SCRIPT>
<BODY>
<H1>Browser'ınızı Tanıyor Musunuz?</H1>
<FORM>
<INPUT Type="button" Value="Browser Hakkında Bilgi Ver" OnClick="siziTaniyalim()"
</INPUT>
</FORM>
</BODY>
</HTML>
```

Mesaj Kutuları

Navigator penceresi, üç tür mesaj kutusu açabilir. Bunlar basit bilgi verme kutusu (alert), doğrulama kutusu (confirm) ve kullanıcının bilgi girme kutusudur (prompt).

Bunların genel yazım kuralı şöyledir:

window.alert(mesaj)

return window.confirm(mesaj)

return window.prompt(mesaj, varsayılanCevap)

Bu üç kutunun çeşitli kullanım şeklini, başka vesilelerle, daha önceki örneklerde gördük.

Çerçeve Nesnesi

Bir Navigator penceresinde birden fazla HTML belgesi görüntüleme imkanı veren çerçeve (Frame) tekniği, tasarım açısından kolaylık sağlar. Bununla birlikte halâ bazı Web ziyaretçilerinin çerçeveden hoşlanmadıkları da gerçektir.

Çerçevelerin, Navigator açısından nasıl bir hiyerarşı işlediğini daha önce gördük. Javascript açısından her bir çerçeve bir pencere sayılır; bunlara atıfta bulunurken şu adlandırma uygulanır:

top En üst pencere. yani Browser'ın kendisi.

parent Herhangi bir Frame'i oluşturan Frameset. Çerçeve çerçeve içinde ise, bir çerçevenin içinde bulunduğu çerçeve parent sayılır. Sayfada bir Frameset varsa, bütün çerçeveler için "top" aynı zamanda "parent" sayılır.

self Çerçevenin kendisi.

Daha önce de belirttiğimiz gibi, Javascript, bir parent (ebeveyn) Frameset'in yavru çerçevelerini 0'dan itibaren numaralar. Yani, sayfanızda iki çerçeveniz varsa, birincisi "parent.frames[0]" ikincisi ise "parent.frames[1]" adıyla bilinir. Herhangi bir çerçevenin içeriğini dinamik biçimde belirlemenin bir örneğini yukarıda gördük. Burada write() metodunu kullanarak, aynı işi nasıl yapabileceğimizi ele alalım.

Şimdi, iki çerçeveli bir Frameset için şu HTML kodunu "anacerceve.htm" adıyla kaydedin:

```
<html>
<head>
<title>Dinamik Cerceve</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>
<frameset cols="40%,*">
<frame SRC="doc1.htm" NAME="cerceve1">
<frame SRC="doc2.htm" NAME="cerceve2">
</frameset>
<noframes>
<body>
</body>
</noframes>
</frameset>
</html>
```

Sonra, bu Frameset'in çağrıda bulunduğu "doc1.htm" olarak, şu kodu kaydedin:

```
<html>
```

```
<head>
<title>Cerceve 1</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<script LANGUAGE="JavaScript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
//fonksiyonumuzu tanimlayalim
function belgeYazdir()
      {
parent.frames[1].document.clear();
parent.frames[1].document.write("<HTML><HEAD>" +
       "<TITLE>Dinamik Güncelleme Örnegi</TITLE>");
parent.frames[1].document.write(" </HEAD><BODY BGCOLOR=\"" +
       document.form1.bginput.value + "\">");
parent.frames[1].document.write("<H1>Güncellenmiş sayfa</H1>");
       parent.frames[1].document.write("Bu sayfayı güncelleştiren unsur:" +
       document.form1.input1.value);
parent.frames[1].document.write("</BODY></HTML>");
parent.frames[1].document.close();
      }
// -->
</script>
</head>
<body>
<form NAME="form1">
<input type="hidden" name="bginput" value="white">
<p><input TYPE="text" NAME="input1" size="20"></p>
<p>Bir zemin rengi seçiniz:<br>
<input TYPE="radio" NAME="radio1" VALUE="white" CHECKED
onClick="document.form1.bginput.value=&quot;white&quot;">Beyaz<br>
<input TYPE="radio" NAME="radio1" VALUE="red"
onClick="document.form1.bginput.value=&quot;red&quot;">Kırmızı<br>
<input TYPE="radio" NAME="radio1" VALUE="blue"
onClick="document.form1.bginput.value=&quot;blue&quot;">Mavi<br>
<input TYPE="radio" NAME="radio1" VALUE="green"
onClick="document.form1.bginput.value=&quot;green&quot;">Yeşil<br>
<br>
```

```
<input TYPE="button" VALUE="Çerçeve 2'yi güncelle" onClick="belgeYazdir()"> </p>
</form>
</body>
</html>
```

"doc2.htm" ise oldukça basit:

```
<html>
<head>
<title>Çerçeve 2</title>
</head>
<body>
<p>Bu sayfa güncellenecek</p>
</body>
</html>
```

Burada, document.write() metoduyla, hedef çerçevemiz olan iki numaralı çerçevenin (parent.frames[1]) birinci çerçevedeki formdan aldığımız unsurları kullanarak, zemin rengini ve içeriğini değiştiriyoruz. Buradaki örnekten hareketle, çerçevelerinizin içeriğini, bir diğer çerçevede yapılan tercihlere, yapılan arama sonuçlarına veya veri-yönlendirmeli sonuçlara ayırabilir ve dinamik olarak değiştirebilirsiniz.

Belge (Document) Nesneleri

Web sayfasının Javascript dilinde adı Belge'dir. İster metin, ister sadece Javascript kodu, ister sadece grafik içersin, <HTML> diye başlayan ve </HTML> diye biten herşey, Javascript gözüyle belgedir.

Javascript'in gözüyle dünyaya bakarsanız, dünyayı içiçe kaplardan ibaret görürsünüz. En dış kabın adı, Navigator; içindeki ikinci kabın adı Belge'dir. Belge'nin içinde ise, yine birbirinin içinde olmak şartıyla, HTML ile oluşturulan nesneler vardır. Sayfanızda örneğin faturaForm adlı bir form varsa, bu Javascript için document.faturaForm demektir, ziyaretçinin doldurduğu bilgilerle Server'a gönderilmesi için document.faturaForm.submit() metodunun kullanılması gerekir. Bu bölümde bir belgedeki Javascript kodu ile yeni bir belge

oluşturup, ya içinde bulunduğumuz sayfanın yerine, ya da yeni bir pencereye yollama yolunu ve HTML nesnelerinin özelliklerini Javascript aracılığıyla nasıl etkileyeceğimize bakalım.

Javascript, içinde bulunduğu HTML belgesi yerine sizin arzu ettiğiniz başka bir belgeyi görüntülüyebilir. Bu belgenin içeriği, ziyaretçinin bir önceki sayfada yaptığı bazı tercihleri yansıtabilir; ziyaretçinin Browser türünden veya kullandığı bilgisayar işletim sisteminin türünden kaynaklanabilir. Aşadığıdaki örnekte, Javascript ziyaretçinin Browser türünü algılayıp, ona göre bir içerik sunuyor. Bu içerik burada olduğu gibi sadece bir hoşgeldin mesajı olmayabilir, Browser türüne uygun içerik olabilirdi. Bu dosyayı, yenibelge01.htm adıyla kaydedip, Browser'ınızda açarsanız, Browser'ınıza uygun mesajı göreceksiniz:

```
<html>
<head>
<title>Yeni Belge</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>
<SCRIPT LANGUAGE = "JavaScript1.2">
var Browser = navigator.appName
document.open()
if (Browser == "Netscape") {
document.write("<h2>Sayın <a href='http://home.netscape.com'>Netscape</a>
kullanıcısı</h2><p><p>")}
else {
if (Browser == "Microsoft Internet Explorer") {
document.write("<h2>Sayın <a href='http://www.microsoft.com'>Microsoft</a> Internet
Explorer kullanıcısı</h2><p><p>") }
else {
document.write("<h1>Hoş geldiniz! Ne tür Browser kullandığınızı
anlayamadık!</h1><p><p>")
}
}
document.write("Sitemize hoş geldiniz.<p><p>")
document.write("Kullandığınız Browser türünü anladık diye şaşırdınız mı?")
document.close()
</SCRIPT>
<body>
```

```
</body>
</html>
```

Burada olduğu gibi, yeni belge mevcut belgenin yerine açılmak yerine, Frameset ile oluşturduğunuz çerçeveli bir sayfada arzu ettiğiniz bir çerçevenin içeriğini de oluşturabilir. Bunun için, yukarıda olduğu gibi document.write() metodu yerine,

```
parent.form[0].document.write()
```

metodunu kullanmanız gerekir. Burada, birinci çerçeve için köşeli parantez içine 0, ikinci çerçeve için 1, üçüncü çerçeve için 2 yazmanız gerektiğini daha önce belirtmiştik.

Javascript ile oluşturacağınız yeni belge, tamamen başka ve hatta yeni bir Browser penceresi içinde de görüntülenebilir. Bu teknikten, sayfalarınızda açıklayıcı bilgi ve yardım bilgisi vermekte yararlanabilirsiniz; hatta reklam amacıyla bile bu yöntemi kullanabilirsiniz. Aşağıdaki kodu, yenibelge02.htm adıyla kaydederseniz, ziyaretçinin Mouse işaretçisini bağlantılı gibi görünen kelimelerin üzerine getirdiğinde, Javascript anında yeni bir pencere açacak, ve içine arzu ettiğiniz metni yazacaktır:

```
<html>
<head>
<title>Yeni Belge</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<script LANGUAGE="JavaScript1.2">
function acYardim01(){
yeniPencere = window.open("", "yardim01", "height=300,width=400")
yeniPencere.document.write("<h2>Bu pencerede Konu 1 ile ilgili ek bilgiler
olacak</h2>")
yeniPencere.document.write("<p>Tabii birisi zahmet edip, yazarsa!</p>")
yeniPencere.document.write("<p><font size=-1><a
href=\"javascript:self.close()\">Mevzuya Donmek Icin Tiklayin</a></font>")
yeniPencere.document.close()
}
function acYardim02(){
yeniPencere = window.open("", "yardim02", "height=300,width=400")
```

```
yeniPencere.document.write("<h2>Bu pencerede Konu 2 ile ilgili ek bilgiler
olacak</h2>")
yeniPencere.document.write("<p>Tabii birisi zahmet edip, yazarsa!</p>")
yeniPencere.document.write("<p><font size=-1><a
href=\"javascript:self.close()\">Mevzuya Donmek Icin Tiklayin</a></font>")
yeniPencere.document.close()
}
function acYardim03(){
yeniPencere = window.open("", "yardim03", "height=300,width=400")
yeniPencere.document.write("<h2>Bu pencerede Konu 3 ile ilgili ek bilgiler
olacak</h2>")
yeniPencere.document.write("<p>Tabii birisi zahmet edip, yazarsa!</p>")
yeniPencere.document.write("<p><font size=-1><a
href=\"javascript:self.close()\">Mevzuya Donmek Icin Tiklayin</a></font>")
yeniPencere.document.close()
}
</script>
</head>
<body>
<h1>Sitemizdeki yeni konular aşağıdadır:</h1>
<h2>Konular hakkinda daha fazla bilgi almak icin Mouse isaretcisini konunun uzerine
getirin.</h2>
<p><a HREF="#" onMouseOver="acYardim01()">Yeni konu 1</a></p>
<p><a HREF="#" onMouseOver="acYardim02()">Yeni konu 2</a></p>
<p><a HREF="#" onMouseOver="acYardim03()">Yeni konu 3</a></p>
</body>
</html>
```

Böyle bir pencerede görüntülemek istediğiniz metin, Javascript fonksiyonu ile yazdırılamayacak kadar uzun ise, yeni pencere açtırma kodunuzda şöyle bir değişiklik yapabilirsiniz:

```
function acYardim01(){
yeniPencere = window.open("yardim01.htm", "yardim01", "height=300,width=400")
}
```

Bu durumda, Javascript, yeni pencerenin içeriğini kendisi document.write() metodu ile kendisi yazmayacak, pencere içeriği olarak örneğin yardım01.htm adlı dosyayı görüntüleyecektir.

Bu konuyu kapatmadan; yukarıdaki örnek kodda, Javascript'in yeni pencerenin içeriğini kendisi yazdığı acYardim01 adlı fonksiyonda kullandığımız şu koda dikkat edin:

```
yeniPencere.document.write("<p><font size=-1><a
href=\"javascript:self.close()\">Mevzuya Donmek Icin Tiklayin</a></font>")
yeniPencere.document.close()
```

Burada kullandığımız yerel Javascript kodundaki "self.close()" metodu, Javascript'e belgenin içinde bulunduğu kendi penceresini kapattırmaktadır. (self, kendisi anlamına geliyor.)

Kodumuzu biraz daha geliştirebiliriz. Sözgelimi, MouseOver olayı ile açtığımız ikinci penceresi, MouseOut olayı ile kapatabiliriz. Bunun için, tabiî önce yeni pencere açan fonksiyonlarımızdan sonra bir de pencere kapatan fonksiyon yazmamız gerekir:

```
function kapat() {
yeniPencere.close();
      }
```

Sonra, yardım pencerelerini açtığımız onMouseOver yönlendirisiyle ilgili kodun hemen arkasına, bu kez kapat() fonksiyonunu çağıran onMouseOut yönlendiricisini yazacağız. Bu örnekteki üç satırdan biri, şöyle olabilir:

```
<p><a HREF="#" onMouseOver="acYardim01()" onMouseOut="kapat()">Yeni konu 1</a></p>
```

Bu durumda, açılan yeni pencerelerin içindeki kendi kendisini kapatan kodu iliştirdiğimiz satırı tıklama imkanınız hiç olmayacağı için (neden?), bu kodun satırını, acYardim01() diğer fonksiyonlarımızdan çıkartmamız yerinde olur. Bu yeni şekliyle kodunuzu yenibelge03.htm adıyla kaydedebilirsiniz.

Form Nesneleri

Javascript'in icad edilme nedenlerinin başında, ziyaretçi ile etkileşme gelir; ziyaretçi ise Web tasarımcısı olarak size bilgileri ancak ve sadece Form ile ulaştırabilir. Bu sebeple Javascript açısından HTML'in en önemli nesneleri, Form nesneleridir.

HTML, kendi formunu kendisi oluşturur; bu bakımdan Javascript'e bir görev düşmez. Javascript formdaki bilgilerin işlenmesi sırasında devreye girer. Önce Form nesnesinin özelliklerini hatırlayalım:

<FORM

NAME=formun_adı

ACTION=CGI_programı

ENCTYPE=kodlama_türü

METHOD= GET veya POST

TARGET= pencere_adı

onSubmit="metodun_adı">

</FORM>

Bu durumda, tam teşekküllü bir form etiketi şöyle olabilir:

```
<FORM NAME="form1" ACTION="http://www.pcworld.com.tr/cgi-bin/form.pl" METHOD=
GET></FORM>
```

Bu, gerçek bir bilgi derleme formu örneği. Fakat Form nesnesi, Server'a bilgi göndermek amacı dışında da kullanılabilir; bir form nesnesi olan IPNUT'un bir türü olan düğmeleri (button) farklı amaçlı Javascript fonksiyonlarına bağlanabilir.

Form, gerçek amacıyla kullanıldığı durumlarda, Javascript, formdaki bilgilerin Server'a gönderilmeden önce sınanmasını sağlayabilir. Bunun bir örneğini, ziyaretçinin telefon numarasını doğru biçimde yazıp yazmadığını sınadığımız kodda görmüştük. Burada

kısaca, Javascript açısından form nesnesinin unsurlarını nasıl tanıyabileceğimizi ele alalım. Benzeri sınama kodları yazarken, bu unsurlara referans yapacağınız zaman, bu unsurların adlarını ve değerlerini bulma yöntemini bilmeniz gerekir.

Form etiketi içindeki bütün unsurlar **element** adlı dizi-değişkenin içine yazılırlar ve içerikleri form.element[i].value (i, unsurun sıra numarası olmak üzere) metoduyla bilinirler. Diyelim ki "bilgiForm" adlı formunuzun birinci unsuru "<INTPUT TPYE="text" NAME="musteriAdi"...> şeklinde bir etiket. Bu etiketin oluşturacağı kutuya ziyaretçinin yazacağı isim, Javascript açısından bilgiForm.elements[0].value veya bilgiForm.musteriAdi.value şeklinde bilinir.

Formun unsurları

Şimdi bir form içeren sayfa örneği yapalım; Javascript ile formun tamamen doldurulmuş olup olmadığını sınayalım. Böylece ziyaretçinin göndereceği formun eksik olduğunu anlamakta geç kalmamış oluruz. Aşağıdaki kodu, form01.htm adıyla kaydedip, Browser'da bir yerini eksik olarak duldurun ve Gönder düğmesini tıklayın.

```
<html>
<head>
<title>Form</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<script LANGUAGE="JavaScript1.2">
function denetle() {
var num = document.form1.elements.length
var dogruMu = true
       for (var i=0; i<num; i++) {
              if ((document.form1.elements[i].value == null ||
               document.form1.elements[i].value == "") &&
               (typeof document.form1.elements[i] != 'submit' ||
               typeof document.form1.elements[i] != 'reset'))
{
                     dogruMu = false
```

```
                    alert("Formdaki " + document.form1.elements[i].name +
                          " alanı doldurulmamış bulunuyor. Lütfen doldurunuz!")
                    break }
             }
       return dogruMu
      }
// -->
</script>
</head>
<body>
<form name="form1" method="POST" onSubmit="return denetle()">
<h2>Abone Formu</h2>
<p>Adınız:<br>
<input type=text size=25 maxlength=256 name="Abone_Adı"><br>
İlgi Alanı:<br>
<input type=text size=25 maxlength=256 name="İlgi"><br>
<strong>Oturduğunuz İl: <br></strong>
<input type=text size=25 maxlength=256 name="İl"><br>
<strong>Elektronik Posta Adresiniz: <br></strong>
<input type=text size=25 maxlength=256 name="ElektronikAddress"></p>
<input type=submit value="Gönder"> <input type=reset value="Sil">
</form>
</body>
</html>
```

Daha önceki örneklerden hatırlıyorsunuz; formun içerdiği bilgilerin gerçekten Server'a gitmesi için gerekli "submit" olayının önünü kesebilmek için kullandığımız "onSubmit" olayına bağlı fonksiyonun "return" ifadesiyle, kendisini görevlendiren olay-yönlendiriciye bir sonuç bildirmesini sağlamak zorundayız. Fonksiyonu biraz irdelersek, başta tanımladığımız Boolean değişkeni "dogruMu" bize doğru-yanlış türü bir sonuç vermek zorunda; ziyaretçi herşeyi doldurmuşsa (yani her alana hiç değilse bir harf yazmışsa!) bu değişkenin değeri değişmiyor. Fakat "if" sorgulamasında düğme olmayan unsurlardan birinin değeri boşsa (yani null veya "" ise) değişkenin değeri **false** olacaktır. "if" sorgulamasını kaç kere yapacağız? Bu sayıyı, belgenin form elemanlarının yazıldığı dizi-değişkenin

büyüklüğünden (document.form1.elements.length) öğreniyoruz, ve num adlı değişkene yazıyoruz. Dolayısıyla Forma yeni alanlar da eklesek, fonksiyonu yeniden yazmamıza gerek yok. Sonra, i adını verdiğimiz bir sayaç, num değişkeninin değerine eşit oluncaya kadar, elements dizi-değişkenindeki (submit ve reset dışındaki) bütün elemanların değerlerin boş olup olmadıklarına bakıyoruz. Bu değerlerden herhangi biri boş ise, if döngüsü o noktada kesiliyor ve ziyaretçiye bir uyarı kutusu ile formu doldurmasını hatırlatıyoruz. Bu arada uyarı kutumuz, boş olan alanın adını da bildiriyor. Bunu, if döngüsü kesildiği andaki i değişkeni değerindeki form elemanının adını alarak yapıyoruz.

Buradaki örnekte düğme (button) nesnesinin Click olayını onClick yönlendiricisi ile yakaladık. Form nesnesindeki diğer nesnelerden metin (text) ve metin alanı (textarea) nesneleri, onBlur, onChange, onFocus, onSelect yönlendiricileri ile; işaret kutusu (checkbox) ve radyo düğmesi (radio) onClick yönlendiricisi ile, seçme listesi (select) onBlur, onChange ve onFocus yönlendiricileri ile fonksiyon çağırabilirler.

Javascript fonksiyonlarımızda Form nesnesinin işaretleme kutularından hangisinin işaretlenmiş olduğunu belirlemek işaretli unsurun değerlerini bulabilmek, değiştirebilmek veya başka bir yerde kullanabilmek için önem taşır. Browser, ele aldığı bir HTML sayfasındaki Form nesnesinin unsurlarını ve burada CHECKBOX nesnesini bir dizi-değişkenin içinde tutar; bunlardan işaretlenmiş olanın CHECKED (işaretli) özelliğini True (doğru) yapar. Bu durumdan yararlanarak istediğimiz kutunun işaretli olup olmadığını sorgulayabiliriz. Burada nesnenin ".value" özelliği işe yaramaz; çünkü kutu işaretli olsa da olmasa da ".value" özelliği aynıdır. İşaretleme kutularında bilmek istediğimiz husus, bu nedenle işaretli olup olmadıklarıdır. Örneğin, sayfamızda, "form1" adlı bir formda "isaret01" adlı bir işaret kutusu varsa, bu kutunun işaretli olup olmadığını şu if döngüsü sınayabilir:

```
if (document.form1.isaret01.checked) {
.......
```

Bu ifade, kutu gerçekten işaretli arzu ettiğimiz bir işi yapar, değilse yapmaz.

Radyo Düğmesi (Radio)

Ziyaretçinin bir HTML sayfasında bir çok şık'tan birini seçmesini sağlayan radyo düğmesi (radio) isimli işaretleme yönteminde ise, aynı adı taşıyan bütün radyo düğmelerinden biri işaretlenince, Browser diğerlerini işaretlenmemiş hele getirir; dolayısıyla bu düğmenin değeri (.value özelliği) kullanılabilir. Aşağıdaki kodu yazar ve radyo.htm adıyla kaydederek, Browser'da açarsanız, seçtiğiniz radyo düğmesinin değerinin uyarı kutusunda kullanıldığını göreceksiniz.

```
<HTML>
<HEAD>
<TITLE>Radyo Dugmesi Örnegi</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
<SCRIPT LANGUAGE = "JavaScript1.2">
function radyoDegeri(radyoNesnesi) {
var deger = null
for (var i=0; i<radyoNesnesi.length; i++) {
if (radyoNesnesi[i].checked) {
deger = radyoNesnesi[i].value
break }
}
return deger
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="form1">
<p><input type=radio name="sanatci" value="Türkü">Neşet Ertaş</p>
<p><input type=radio name="sanatci" value="Şarkı">Zeki Müren</p>
<p><input type=radio name="sanatci" value="Pek anlamadım!">Sezen Aksu</p>
<input type=button value="Ne tür müzik seviyorum?"
onClick="alert(radyoDegeri(this.form.sanatci))">
</FORM>
</BODY>
</HTML>
```

HTML kuralları gereği, bir grup oluşturan radyo düğmeleri aynı adı taşırlar. Burada "this" anahtar kelimesi ile içinde bulunduğumuz yani "bu formun," "sanatcı" dizi-

değişkeninin tümüyle fonksiyona gönderilmesini sağlıyoruz. "radyoDegeri" adlı fonksiyonumuz zaten böyle bir öbek bilgi bekliyor ve aldığı tüm radyo düğmesi nesnesine "radyoNesnesi" adını verip, önce büyüklüğüne bakıyor. Bir dizi-değişkenin büyüklüğü eleman sayısıdır; ki bu bizim örneğimizde üçtür. "i" sayacı 0, 1 ve 2 değerleri için sınanıyor ve bunlardan birinin işaretli olması bekleniyor. İşaretli radyo düğmesi bulunduğu anda değeri "deger" değişkenine yazılıyor ve döngü kesiliyor (break). Gerisi, uyarı kutusunun "değer" değişkeni ziyaretçiye göstermesinden ibaret.

Select

Form nesnelerimizden bir diğeri ise SELECT etiketiyle oluşturduğumuz seçme kutusu. NAME, SIZE, ve MULTIPLE özelliklerine sahip olan bu nesne, onBlur, onChange, onFocus olay-yönlendiricileri ile kullanılabilir. SELECT nesnesinin değeri, kendisinin ayrılmaz parçası olan OPTION etiketinin VALUE özelliğinden gelir. Yukarıdaki radyo.htm kodunu, aşağıdaki şekilde değiştirip, sec01.htm adıyla kaydedin:

```
<HTML>
<HEAD>
<title>Seçme Kutusu Ornegi</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<SCRIPT LANGUAGE = "JavaScript1.2">
function secDegeri(secNesnesi) {
       return secNesnesi.options[secNesnesi.selectedIndex].value
       }
</SCRIPT>
</HEAD>
<BODY>
<FORM name="form1">
<p><SELECT NAME="sanatci" SIZE=1>
<OPTION value="Türkü">Neset Ertas</OPTION>
<OPTION value="Sarki">Zeki Müren</OPTION>
<OPTION value="Pek anlamadim!">Sezen Aksu</OPTION></P>
<input type=button value="Ne tür müzik seviyorum?"
onClick="alert(secDegeri(this.form.sanatci))">
</FORM>
```

```
</BODY>
</HTML>
```

Burada ziyaretçi açısından seçme işlemi, seçme kutusunun sağındaki seçme okuna basarak yapılıyor; fakat bizim açımızdan da önemli bir değişiklik var. Seçilen unsur, SELECT nesnesinin dizi-değişkenine değil, "selectedIndex" adlı endeksin değer (.value) özelliğine yazılıyor. Dolayısıyla, bir if döngüsü ile arama yapmak gerekmiyor; bu değeri uyarı kutusuna sadece ayıklama işlemiyle belirleyip geri gönderiyoruz. Burada dikkat edeceğiniz husus, fonksiyona verilen "bu form" nesnesinin "sanatcı" adlı SELECT nesnesinin bilgi kümesi içinde sadece biri, "selectedIndex" adlı endekse sahiptir. Fonksiyon bu bilgi kümesini "secNesnesi" adıyla ele alıyor ve içindeki "options" dizi-değişkenin ziyaretçinin seçtiği ve dolayısıyla sıra numarası "selectedIndex" olarak işaretlenmiş olanın değerini (.value) alıyor ve bunu doğruca Uyarı kutusuna gönderiyor.

Fakat, HTML'den biliyorsunuz ki, SELECT etiketine MULTIPLE anahtar kelimesini koyarak, ziyaretçinin birden çok seçme yapmasına izin verebilirsiniz. Bu durumda "selectedIndex" işaretini kullanabilir miyiz? Hayır; bu bir değişken olduğuna göre sadece bir değer (seçilen ilk OPTION'ın sıra numarasını) tutacaktır. Fakat SELECT nesnesinin bir de "selected" özelliği vardır ki içinde seçilen bütün OPTION'ların numarası yazılıdır. Son kod örneğimizi bu kez şöyle değiştirelim ve sec02.htm adıyla kaydedelim.

```
<html>
<head>
<title>Seçme Listesi Ornegi</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<SCRIPT LANGUAGE = "JavaScript1.2">
function secDegeri(secilenIsim) {
var liste = ""
for (var i=0; i<secilenIsim.length; i++) {
       if (secilenIsim.options[i].selected) {
       liste += secilenIsim.options[i].text + "<p>"
             }
      }
```

```
yeniPencere = window.open ("", "Seçtikleriniz", "height=200, width=200")
yeniPencere.document.write("<H2>Şu şarkıları seçtiniz:</H2><P><P>")
yeniPencere.document.write(liste)
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="form1">
<p><SELECT NAME="sanatci" SIZE=4 MULTIPLE>
<OPTION>Neset Ertas</OPTION>
<OPTION>Zeki Müren</OPTION>
<OPTION>Sezen Aksu</OPTION>
<OPTION>Kayahan</OPTION>
<OPTION>Mustafa Sandal</OPTION>
<OPTION>Müşerref Tezcan</OPTION>
</P>
<input type=button value="Hangilerini seçtim?" onClick="secDegeri(this.form.sanatci)">
</FORM>
</BODY>
</HTML>
```

Burada seçilenlerin listesini yeni bir pencerede açtığımız belgeye yazdırıyoruz. Bu size, bir formun unsurlarını elde ettiğiniz listeyi daha sonraki bir sayfada, HTML unsuru olarak kullanabileceğinizi gösteriyor. Kodu kısaca irdelersek; fonksiyonumuz, for döngüsünün içindeki if döngüsü ile, kendisine ulaşan bilgi kümesinde "selected" olarak işaretlenenleri seçmekte ve bunların metnini (.text), liste değişkenine, aralarına <P> etiketi koyarak yazıyor.

Password

Form nesnesinin Javascript yoluyla mutlaka kullanmak isteyeceğiniz bir ögesi, Password (Parola) nesnesidir. Ziyaretçinin, Password olarak nitelediğiniz INPUT kutusuna yazdığı metin, kendisine * olarak görünür; fakat Browser ve dolayısıyla Javascript bu metni bilir.

Javascript'in ilk sürümlerinde Password nesnesinin değeri Javascript tarafından bilinemezdi; fakat 1.1 sürümü ile Javascript bu nesnenin de bütün kontrolünü ele geçirdi. Aşağıdaki kodu parola.htm adıyla kaydederseniz, Password nesnesinin de değerini elde ederek, kullanabileceğinizi göreceksiniz.

```
<html>
<head>
<title>Parolayı Görüntüle</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<SCRIPT LANGUAGE = "JavaScript1.2">
function parolaNedir()
      {
alert("\nisim --->" + document.form1.gizli.name +
"\ndeger --->" + document.form1.gizli.value)
      }
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="form1">
<BR>
<B>Lütfen parola olarak herhangi bir kelime yazıp düğmeyi tıklayın:</B>
<BR><BR>
<INPUT TYPE="password" NAME="gizli" SIZE="15">
<INPUT TYPE="button" NAME="goster" VALUE="Metni görmek için tiklayin"
onClick="parolaNedir()">
</FORM>
</BODY>
</HTML>
```

Burada herhangi bir form unsuru gibi, form1'in "gizli" adlı nesnesinin değerini alarak Uyarı kutusunda gösteriyoruz. Gerçek bir programda, bu değeri bir liste ile karşılaştırarak, kullanabilirsiniz. Ancak bu karşılaştırma işleminin ziyaretçinin bilgisayarında değil, Server'da yapılması gerekir; bunun için Password nesnesinin değerinin bir CGI programında veya ASP uygulamasında değerlendirilmesi gerekir. Browser'lar bunu bildikleri için Password nesnesini

daima Server'a yollarlar. Bunu sınamak istiyorsanız, yukarıdaki kod Browser'ınızda açıkken, düğmeyi tıklamayın, klavyede Enter tuşuna basın!

Gizli Nesneler (Hidden)

HTML'in INPUT etiketiyle sayfanıza türü Hidden (Gizlenmiş) olan değişkenler koyabilirsiniz. Hidden etiketi, NAME ve VALUE özelliklerine sahiptir. Bu etiketten, genellikle ziyaretçinin Form verilerini değerlendirirken, ziyaretçi sayfa değiştirdiği halde Server'da değeri sabit kalan değişken olarak yararlanabiliriz.

Javascript Nesneleri

Bu noktaya kadar ele aldığımız bütün nesneler, Javascript'e HTML ve Browser tarafından sunuluyordu. Fakat Javascript, kendi kendisine nesneler de oluşturabilir. Örneğin, Javascript bir alfanümerik değişkeni ele aldığı anda bu onun için bir String nesnesi olur; ve dolayısıyla, bir değişkenden öte bazı özellikler ve metodlar kazanır. Dizi-değişkenler de oluşturuldukları anda Array nesnesi olurlar; yeni özellikleri ve metodları olur.

String Nesnesi

Javascript kodunuzun ya HTML'in HEAD bölümünde genel nitelikli, ya da herhangi bir fonksiyonun içinde yerel nitelikli bir değişken tanımladığınız zaman bu türüne göre bir String (alfanümerik, karakter) değişken olur. Örneğin:

var kitabınAdi = "Gazap Üzümleri"

ifadesi bize içeriği "Gazap Üzümleri" olan bir değişken kazandırır. İçeriğinin niteliği nedeniyle bu değişken, String değişkendir. Fakat şu aşağıdaki ifadeye bakarsanız, Javascript'e yeni bir String nesnesi oluşturması için de emir verebiliriz:

var kitabinAdi = new String("Gazap Üzümleri")

"new," (=yeni) hatırlıyorsanız, Javascript'e yeni bir nesne oluşturması için verdiğimiz komuttu. Burada Javascript'e yeni bir String nesnesi oluşturmasını; bunun içeriğini "Gazap

Üzümleri" yapmasını, ve bu nesneyi, "kitabinAdi" isimli değişkene atamasını söylüyoruz. değişken oluşturma açısından ikisi arasında hiç bir fark olmamakla birlikte, bu ikinci yöntem bize, alfanümerik değişkenlerin özel niteliğini, nesne olduklarını gösteriyor. Bu sayede alfanümerik değişkenlerin (yani String nesnelerinin) bazı ilave özelliklerinden yararlanabiliriz. Bu özelliklerin sık kullanılanlarını şöyle sıralayabiliriz:

length	Nesnenin uzunluğunu belirtir. Örneğin kitabinAdi.length, bize 14 değerini verir.
charAt(i)	i ile belirttiğiniz pozisyondaki karakteri verir. kitabınAdi.charAt(1) bize "a" değerini verir.
indexOf(nnn)	nnn ile belirttiğiniz karakterlerin String içinde ilk geçtiği konumun endeksini verir. kitabınAdı.indexOf("za") ifadesi, 2 değerini verir. (G, 0 olmak üzere saymaya başlayın!)
lastIndexOf(nn)	nnn ile belirttiğiniz karakterlerin String içinde geçtiği son konumun endeksini verir. kitabınAdi.lastIndexOf("a"), bize 3 değerini verir.
substring(i,j)	String nesnesinin i'de başlayan ve j'de biten bölümünü verir. kitabinAdi.subtring(0,3) ifadesi bize "Gaz" değerini verir.

Bu özellikler, sadece bir alfanümerik değişken veya öğrendiğimiz yeni adıyla String nesnesinde değil, String olabilecek herhangi bir metinde de vardır. Örneğin "Gazap Üzümleri".length, 14; "Gazap Üzümleri.charAt(1) ise "a" değerini verebilir.

String nesnelerinin kendilerine özgü metodları da vardır. Bunları da kısaca ele alalım:

.bold()	Bağladığınız String nesnesini koyu yapar. Örneğin kitabınAdi.bold(), size "<B>Gazap Üzümleri</B>" metnini verir.

.fontcolor(“renk”)	String nesnesinin görüntülenme rengini belirler. Örneğin kitabınAdi.fontcolor(“red”) size “<FONT COLOR=”red”>Gazap Üzümleri</FONT>” değerini verir.
.fontsize(“ölçü”)	String nesnesinin görüntülenmesinde harf büyüklüğünü belirler. Örneğin kitabınAdi.fontsize(“24”) size “<FONT SIZE=”24”>Gazap Üzümleri</FONT>” değerini verir.
.italics()	String nesnesinin itelik harfle görüntülenmesini sağlar. Örneğin kitabınAdi.italics() size “<I>Gazap Üzümleri</I>” değerini verir.
.toLowerCase()	String nesnesinin görüntülenmesi sırasında bütün harflerinin küçük harf olmasını sağlar. Örneğin kitabınAdi.toLowerCase () size “gazap üzümleri“ değerini verir.
.toUpperCase()	String nesnesinin görüntülenmesi sırasında bütün harflerinin büyük harf olmasını sağlar. Örneğin kitabınAdi.toLowerCase () size “GAZAP ÜZÜMLERİ “ değerini verir.

Javascript ile Dinamik HTML

Bu kadar uğraşıp, Javascript öğrenmemizin, iki amacı olabilir; ziyaretçi ile etkileşme ve HTML yapılması imkanı bulunmayan görsel etkileri oluşturma. Ziyaretçinin Form yoluyla size vereceği bilgileri, yapacağı tercihleri ve bunları tutan değişkenlerin değerlerini nasıl belirleyeceğimizi ve kullanabileceğimizi ana hatlarıyla gördük. Buraya kadar edindiğimiz bilgilerden yararlanmak ve bunları sayfalarımızın içeriğine bağlı olarak gerçek durumlara uygulamak tabir yerinde ise kolay. Javascript'in Browser ve HTML olaylarını (MouseOver, Click, Change gibi) yönlendirmekte kullandığı yönlendiricilerinden (onMouseOver, onClick, onChange gibi) yararlanarak, sayfalarımızda belki de Javascript'i icad edenlerin hiç de niyet etmedikleri bir tarzda da kullanabiliriz. Bu bölümde Javascript ile oynayabileceğimiz görsel oyunları veya Javascript-Layer ve Javascript-CSS ilişkilerini ele alacağız.

Fakat önce kısaca Dinamik HTML'in (DHTML) imkan ve yeteneklerinden söz edelim. HTML ile DHTML arasındaki fark, ikincisinin stil komutları, iki boyutlu tasarıma üst üste yığılan katmanları ekleyerek üçüncü boyutu getiren ve veritabanı ile çoklu-ortam ürünleri ile klasik Web unsurları arasında bağ kuran teknolojiler içermesidir. DHTML, HTML'e ek etiket getirmez; mevcut etiketlere ilave özellik ve nitelik kazandırır. Bunu CSS (Cascading Style Sheets) teknolojisi sağlar. Ayrıca DHTML yoluyla, HTML'in bilinen etiketlerinin bilinen özellikleri, ziyaretçinin sayfanızda yapacağı tercihlere, tutum ve davranışlarına göre dinamik olarak değişmesini programlayabilirsiniz. Başka bir deyişle, DHTML, HTML etiketlerinin program yoluyla dinamik hale getirilmesi demektir.

Browser dünyasında IE, bir HTML sayfasındaki bütün unsurların programlanmasına imkan veren ve adına Element Modeli denen bir yöntem uygulayabilir. Fakat bunu yaptığınızda, bu kez Netscape'in bu modeli tanımayan yorum tarzına takılabilirsiniz. Başka

bir ifadeyle, IE için bir sayfanın tümü, programlandığı belirtilmeden programlanabilir. Bir örnek verelim. H1 etiketini tanırsınız ve sürekli kullanırsınız; H1 olarak etiketlenmiş bir metnin MouseOver olayı ile ilgisi olamaz, diye düşünebilirsiniz. O halde şu kodu herhangi bir sayfaya yerleştirin:

<H1 onMouseOver="this.style.color = 'red';" onMouseOut="this.style.color = 'black';" >Beni Kırmızı Yap!</H1>

Sonra bu sayfayı önce IE ile sonra Netscape ile açın; her ikisinde de Mouse işaretçisini "Beni Kırmızı Yap!" kelimelerinin üzerine getirin ve geri çekin. Bu deneyden sonra hayalkırıklığına kapılmayın; aynı etkiyi Netscape ile de sağlamanız mümkün; fakat bunu Javascript programı ile yapabilirsiniz.

Şimdi bu tür Javascript programlarının örneklerini görelim.

Düğme Grafiği Animasyonu

Cascading Style Sheets (Yığılmalı Stil Sayfaları) denen ve HTML'in kendi önceden-tanımlanmış stil etiketlerinin (H1, H2, P gibi) biçimini değiştirmek dahil, kendi stil etiketlerinizi tanımlamanıza imkan veren ek imkana geçmeden önce, şu ana kadar öğrendiğimiz Javascript komutlarını ve HTML olaylarını kullanarak, kendimize bir Anime (Hareketli) İleri-Geri düğmesi içeren sayfa yapalım. Hatırlayacaksınız, HTML'in Anchor etiketi, Mouse işaretçisinin üzerine gelmesi (MouseOver) ve geri çekilmesi (MouseOut) olaylarına karşı hassastı; ve biz iki olayı onMouseOver ve onMouseOut yönlendiricileri ile istediğimiz fonksiyona bağlıyabiliyorduk. Aşağıdaki kodun mantığı, buna dayanıyor; önce fonksiyonumuzda kullanmak üzere dört değişken tanımlıyoruz ve bunlara düğmelerimizin adlarını ve boyutlarını değer olarak veriyoruz. Bu alıştırmayı yapmaya başlamadan önce aynı boyda dört grafiğiniz olması gerekir: (1) İleri düğmesinin Mouse işaretçisi üzerine geldiği sıradaki görüntüsü (ileri_on.gif), (2) İleri düğmesinin Mouse işaretçisi üzerinden çekildiği sıradaki görüntüsü (ileri_out.gif), (3) Geri düğmesinin Mouse işaretçisi üzerine geldiği

sıradaki görüntüsü (geri_on.gif), ve (4) Geri düğmesinin Mouse işaretçisi üzerinden çekildiği sıradaki görüntüsü (geri_out.gif). Konumuz grafikçilik değil, ama düğme grafiklerinizde yazı ve diğer unsurların yerlerinin aynı olması ve "on" ve "out" türleri arasında dikkat çekici bir derinlik (boyut) farkı bulunması yerinde olur.

"dugmeDegistir" isimli fonksiyonumuz, onMouseOver ve onMouseOut yönlendiricilerinden iki küme bilgi almaktadır: işlemin ileri mi geri olduğuna ilişkin "ileri" veya "geri" kelimesini ve bu bu halde kullanması gereken değişkenin adını. Fonksiyonumuz, bu bilgileri kullanarak kullanacağı değişkenin adını elde edecektir. Bunu eval() fonksiyonu ile yapıyoruz. Hatırlayacaksınız, eval() ile, Javascript'in, bir kaç değişkenin içeriğini değerlendirip, bize arzu ettiğimiz bir biçimde sunmasını sağlıyoruz. Burada eval(), "goruntuNesne" adını verdiğimiz değişken-değerini alacak ("goruntuNesne," fonksiyonun onMouseOver veya onMouseOut'tan aldığı ikinci küme bilgi, yani ya "ileri_onDugme," ya "ileri_outDugme," ya "geri_onDugme," ya da "geri_outDugme") ve buna ".scr" metodunu harekete geçirecek kelimeyi ekleyecek. Bu metod ise fonksiyona, Javascript 'e düğmenin kaynağını (source) bulması için yol gösterecek.

Aşağıdaki kodu animeDugme.htm adıyla kaydeder ve bir yerde saklarsanız, daha sonra bir çok düğme veya grafik animasyonunda size yol gösterebilir:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE= "JavaScript1.2">
<!-- Javascript kodunu eski sürüm Browserlardan saklayalim
// grafiklerimizi tanimlayalim
var geri_outDugme = new Image( 54, 44 );
geri_outDugme.src = "geri_out.gif";
var geri_onDugme = new Image( 54, 44 );
geri_onDugme.src = "geri_on.gif";
var ileri_outDugme = new Image( 54, 44 );
ileri_outDugme.src = "ileri_out.gif";
var ileri_onDugme= new Image( 54, 44 );
```

```
ileri_onDugme.src = "ileri_on.gif";
// degisiklik fonksiyonunu tanimlayalim
function dugmeDegistir(nereye, goruntuNesne) {
       document.images[nereye].src = eval( goruntuNesne + ".src" )
      }
//-->
</SCRIPT>
</HEAD>
<BODY>
<CENTER>
<A HREF="javascript:history.back()"
onMouseOver = "dugmeDegistir( 'geri', 'geri_onDugme' );window.status='Geri';return
true;"
onMouseOut = "dugmeDegistir( 'geri', 'geri_outDugme' );window.status='';return true;">
<IMG SRC="./geri_out.gif" BORDER=0 WIDTH=52 HEIGHT=42 NAME="geri">
</A>
<A HREF="javascript:history.forward()"
onMouseOver = "dugmeDegistir( 'ileri', 'ileri_onDugme' );window.status='Ileri';return
true;"
onMouseOut = "dugmeDegistir( 'ileri', 'ileri_outDugme' );window.status='';return
true;">
<IMG SRC="ileri_out.gif" BORDER = 0 WIDTH = 52 HEIGHT = 42 NAME = "ileri"></A>
</CENTER>
</BODY>
</HTML>
```

Katman Tekniği (DIV, LAYER)

Katman, HTML'e, kendi içindeki bütün unsurların bir bütün olarak ele alınmasını bildirir. Tabir yerinde ise, DIV veya LAYER etiketi ile oluşturacağınız katman ile HTML sayfasının üzerine, altını gösteren bir parşömen kat koymuş oluyorsunuz. CSS tekniklerini kullanarak Layer unsurlarını sayfada diğer herhangi bir HTML unsurundan daha büyük ölçüde ve çok daha hassas şekilde biçimlendirebilir; hatta hareket özelliği kazandırabilirsiniz. Javascript, katmanların bu özelliklerini, çeşitli olaylardan yararlanarak, olay-yönlendiricilerine ve metodlara bağlayabilir.

Ne var ki, katman oluşturmakta kullanabileceğimiz etiketlerden biri olan LAYER Netscape tarafından tanınır fakat IE tarafından tanımaz. Bu sebeple sayfalarımızda katmanları LAYER yerine DIV ile oluşturarak, iki Browser'a da hitabedebiliriz. DIV etiketi de LAYER gibi katmanlar yapar; özellikleri ve metodları da LAYER gibidir. Aralarındaki tek fark, DIV ile oluşturulacak katmanların biçim ve konum özellikleri kendi STYLE komutları ile kazandırmak zorundasınız; oysa LAYER'ın çok daha kestirme kendi biçim özellikleri vardır. Bununla birlikte STYLE metodunu kullanmakla DIV etiketine Javascript ile biçimlendirilecek daha çok "özellik" kazandırabiliriz. önce, etiketini yakından tanıyalım:

<DIV

ALIGN=CENTER, LEFT, RIGHT	Sayfada veya tablo içinde bulunduğu yere göre, ortaya, sola veya rağa hizalanmasını sağlar.
CLASS=sınıf_adı	Uygulanan stil sınıfı varsa, burada belirterek bütün DIV'in aynı stili almasını sağlarız.
DATAFLD=sütun_adı	DIV'e bir veritabanının değeri veriliyor, verilerin geleceği sütunun adı burada belirtilir.
DATAFORMATAS=HTML , TEXT	Bağlanan veritabanının HTML olarak mı, yoksa düz yazı olarak mı yorumlanacağını belirtmeye yarar.
DATASRC=#ID	Varsa, bağlanan veritabanının kimliği
ID=değer	Bu DIV'in kimliği
LANG=dil	ISO standartlarına göre bu bölümde yer alacak metnin yorumlanmasında uygulanacak dil kodu
LANGUAGE=dil	JAVASCRIPT, JSCRIPT, VBS veya VBSCRIPT. Bu DIV etiketinin içindeki Script'in dili. Hiç bir değer belirtmezseniz, JavaScript varsayılır.

STYLE=css1-özellikleri	Bu etiketin unsurlarına uygulanacak stil komutları
TITLE=başlık	Bilgi için kullanılır; Bu unsurun değeri onMouseOver halinde araç bilgi notu olarak görüntülenir.

>...</DIV>

DIV ile oluşturacağınız katman nesnesinin Görünürlük veya Pozisyon belirleme özelliklerini de stil unsurları ile belirtebilirsiniz. Örneğin bir DIV unsurunun HTML sayfası açıldığında görünmemesini istiyorsanız, bunu DIV etiketinin içine koyacağınız şöyle bir stil komutu ile sağlarsınız: "STYLE="visibility:hidden;position:absolute;top:0; left:0; z-index:-1"

Fakat, ne yazık ki, Javascript, DIV'i bir "sayfa unsuru" olarak ele alarak, Browser'a talimat vermeye kalktığında söylediklerini ya Netscape anlamayacaktır; ya da IE. Çünkü iki Browser'ın DIV ve stil belirlemelerinde şart koştukları referans biçimi farklıdır. Örneğin, Netscape, document.layers[1].visibility= "hide" şeklindeki bir Javascript ifadesi üzerine, sayfanızda ikinci katmanı görünmez hale getirirken, IE, size "document.layers1 Nesne Değil!" şeklinde hata mesajı verecektir. IE, sayfanın unsurlarına "document.all.nesne" (nesne kelimesinin yerine nesnenin adı girecek) şeklinde referans yapılmasını ister.

Her iki Browser'ın anlayacağı şekilde Javascript ile katman ve CSS komutları vererek Dinamik HTML sayfaları yapmanın bir yolu, söz gelimi, MacroMedia firmasının DreamWeaver gibi, kendi Javascript kodunu kendini oluşturan ve bunu her iki Browser'ın anlayacağı dilde yazabilen HTML editörü yazılımlar kullanmak olabilir. Ayrıca Scott Isaacs'in (scott@insideDHTML.com) **DHTML Library Version 2.0b5** adlı Javascript programını (dhtmlLib.js), http://www.insideDHTML.com adresinden indirerek, HTML sayfalarınızda kullanabilirsiniz. Bu tür "haricî Javascript belgesi" bir HTML sayfasında HEAD bölümüne şu komutla bağlanır:

<SCRIPT scr="dhtmlLib.js" language="JavaScript"> </SCRIPT>

427 satırlık bu Javascript belgesi programcı olarak size, Javascript stil komutlarnızı ister sadece Netscape'in, ister sadece IE'nin anlayacağı şekilde yazma imkanı veriyor. Bu belge, Browser IE ise Netscape'in anlayacağı komutları IE için, Browser Netscape ise IE'in anlayacağı komutları Netscape için tabir yerinde ise "tercüme" ediyor.

Böyle bir "tercüme" fonksiyonu, ana hatlarıyla şöyle olabilir:

```
function Is() {
var agent = navigator.userAgent.toLowerCase();
this.major = parseInt(navigator.appVersion);
this.minor = parseFloat(navigator.appVersion);
this.ns = ((agent.indexOf('mozilla')!=-1) && ((agent.indexOf('spoofer')==-1) &&
(agent.indexOf('compatible') == -1)));
this.ns2 = (this.ns && (this.major == 2));
this.ns3 = (this.ns && (this.major == 3));
this.ns4 = (this.ns && (this.major >= 4));
this.ie = (agent.indexOf("msie") != -1);
this.ie3 = (this.ie && (this.major == 2));
this.ie4 = (this.ie && (this.major >= 4));
this.op3 = (agent.indexOf("opera") != -1);
}
var is = new Is()
       if(is.ns4) {
       doc = "document";
       sty = "";
       htm = ".document"
      }
       else if(is.ie4) {
       doc = "document.all";
       sty = ".style";
       htm = ""
      }
```

Bu kodu, söz gelimi tercume.js adıyla kaydeder "haricî Javascript belgesi" olarak HTML sayfanızda HEAD bölümüne şöyle bir komutla bağlayabilirsiniz:

<SCRIPT scr=”tercume.js” language=”JavaScript”> </SCRIPT>

Bu fonksiyonu doğruca HTML belgenizin HEAD bölümüne de koyabilirsiniz. Her iki durumda da Javascript’e DIV nesneleriyle ilgili bütün stil komutlarınızı, bir değerlendirmeden geçirerek uygulamasını şöyle bir komutla bildirirseniz;

golgeVer = eval(doc + '["golgeKatmani"]');

Javascript, diyelim ki başlıklara gölge veren “golgeKatmani” adlı bir katmanınız varsa, ve katmanın alacağı “golgeVer” gibi bir değişkeniniz bulunuyorsa bu değişkeni Netscape için:

golgeVer = document.golgeKatmani

IE için ise:

golgeVer = document.all.golgeKatmani

diye belirtecektir. HTML sayfanıza böyle bir “haricî Javascript dosyası” iliştirmek yerine, kendi kodunuzu kendiniz yazmak isteyebilirsiniz. Şimdi böyle bir örneği birlikte yapalım. Önce sayfamızın almasını istediğimiz şu ilk görüntüyü birlikte inceleyelim:

<js006.tif>

Bu sayfada, soldaki spiral süsü, üstte de üç sekme olan bir kitap ya da defter esprisinin hâkim olduğunu görüyorsunuz. Özellikle işletim sistemlerinin ve programların grafik arayüzleri, Internet ziyaretçilerinize sekmeleri tıklama alışkanlğı kazandırmış olmalı. Bu sebeple, bu sayfa ziyaretçiler açısından nerede ne olduğu kolay anlaşılan bir sayfa sayılabilir. Ziyaretçi birinci sekmeyi tıkladığında, bu sekmeyle ilgili bilgilerin bulunduğu sayfaya gidecek, ve belli ki bu sayfa da burada, kapağın bulunduğu alanda görüntülenecektir. Daha teknik ifade edersek, sayfamız açıldığında milimetrik olarak birbirinin üzerinde bulunan dört DIV’den (Kapak, Sekme1_zemin, Sekme2_zemin, Sekme3_zemin) sadece Kapak DIV’i görünür (visible) durumda olacaktır. Ziyaretçi diyelim ki Sekme1’i tıkladığında, Javascript Kapak DIV’ini görünmez (hidden) hale, Sekme1’in sayfasını

içeren DIV'i görünür hale (visible) getirecektir. Ziyaretçinin diğer sekmelerden sonra Sekme1'i yeniden tıklayabileceğini hesaba katarak, Sekme1'e sadece Kapak DIV'ini değil, emniyette olabilmek için Sekme2 ve Sekme3'ün içerik DIV'lerini de görünmez hale getirtmemiz uygun olur. Diğer sekmeler de, kendi sayfalarını açarken, diğer sayfaları ve Kapak'ı, görünmez hale getireceklerdir. Görünmez halde olan bir DIV'i yeniden görünmez hale getirmenin Javascript veya Browser açısından bir sakıncası yoktur.

Bu sayfada sekmelerin içeriklerini tutan DIV'ler ziyaretçinin sekmeleri tıklamasının oluşturacağı Click olayının onClick ile yönlendirilmesi suretiyle değiştirilecektir. HTML'de DIV'lere ilişkin etiketi yazarken uygulayacağınız sıranın hiç önemi yoktur; Browser, DIV'lerin z-index sayısına bakarak hangisinin altta hangisinin üstte olduğunu belirler. Fakat yine de iyi programcılık gereği, DIV'leri en alttan en üste doğru yazmaktır. Şimdi, daha sonra sekmeler.htm adıyla kaydedeceğimiz kodumuzu adım adım birlikte yazalım. Tabiî, önce standart unsurlarımızla sayfanın çatısını çatalım:

```
<HTML>
<HEAD>
<TITLE>Sekmeler</TITLE>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=windows-1254">
<SCRİPT LANGUAGE="JavaScript">
<!-- Eski Browserlardan gizleyelim
.....................
//-->
</SCRİPT>
</HEAD>
<BODY bgcolor="#FFFFFF">
.....................
<p>Sekmeli sayfamıza hoş geldiniz!</p>
<p> </p>
</BODY>
</HTML>
```

Burada nokta-noktalı yerlerin birincisine Javascript fonksiyonumuz, ikincisine ise DIV etiketlerimiz girecek. Fonksiyonu inşa etmeye başlamadan önce, tıklamayla değişmeyecek DIV'lerin etiketlerini yazalım. İşe Spiral'den başlayalım; ikinci noktalı yere şunları yazalım:

```
<div id="Spiral"
       style="position:absolute;
       left:13;
       top:141px;
       width:64px;
       height:288;
       z-index:1;
       background-image: url(spiral.gif);
       layer-background-image: url(spiral.gif);
       border: 1px none #000000;
       visibility: visible">
 </div>
```

Bu örneği böyle satır satır yazmamızın sebebi, DIV etiketinin bölümlerinini açıkça görmeniz içindir: dikkat ettiyseniz, Spiral adlı bu DIV'e CSS ile tayin edebileceğimiz bütün pozisyon özelliklerini kazandırıyoruz. Zemin grafiği olarak spiral.gif diye bir resim kullanıyoruz. Kapak adını verdiğimiz diğer DIV ise şöyle oluşacak:

```
<div id="Kapak" style="position:absolute; left:73; top:141; width:290; height:288px;
z-index:2; visibility: visible; background-image: url(kapak01.gif); border: 1px none
#000000"><h1 align="center"><b><font face="Arial, Helvetica, sans-serif">KİTABIN
İÇİNDEKİLERİ GÖRMEK İÇİN SEKMELERİ TIKLAYINIZ!</font></b></h1></div>
```

Bu iki DIV'in dinamik hiç bir özelliği olmadığı için içeriklerinde onClick veya başka bir yönlendirici yok. Şimdi dinamik DIV'ler olan Sekme1, Sekme2 ve Sekme3 adını verdiğimiz üç sekmeyi oluşturalım; bunların içinde, onClick'in çağıracağı fonksiyonu ve bu fonksiyona vereceğimiz argümanlar (parametreler) bulunacaktır. işte Sekme1'in DIV'i:

```
<div id="Sekme1"
       style="position:absolute;
              left:73px;
              top:95px;
```

```
            width:96;
            height:46;
            z-index:3;
            visibility: visible">
      <a href="#" onClick="katmanGizleGoster(
            'document.layers[\'Sekme1Zemin\']','document.all[\'Sekme1Zemin\']','show',
            'document.layers[\'Sekme2Zemin\']','document.all[\'Sekme2Zemin\']','hide',
            'document.layers[\'Sekme3Zemin\']','document.all[\'Sekme3Zemin\']','hide',
            'document.layers[\'Kapak\']','document.all[\'Kapak\']','hide')">
      <img src="sekme1.gif" width="96" height="46" border="0">
      </a>
</div>
```

DIV etiketi genellikle böyle yazılmaz; fakat neyin-ne olduğunu tartışabilmek için birinci Sekme DIV'imizi açıkça yazıyoruz. DIV'in adından sonra stil komutu ve görünür olup olmadığına ilişkin visibility özelliği belirtiliyor. Sonra, bu DIV'in içine koyacağımız resme "#" bağlantısı kazandırıyor. Bu bağlantı tıpkı bir sayfa veya site bağlantısı gibi, bir metne veya grafiğe A etiketinin özelliklerini kazandırır. Hatırlarsanız, DIV etiketinin olayları arasında en azından 3 ve 4'ncü sürüm Browserlar açısından hemen hiç bir olay (Click, MouseOver gibi) olmadığını, bu nedenle DIV'in onClick, onMouseOver gibi yönlendiricilerle yönlendirilemeyeceğini söylemiştik. Bu sebeple, DIV'in içine bir resim koyarak, bu resme Anchor etiketi bağlayıp ("#" şeklindeki bu Link tıklandığında Browser hiç bir yere gitmeyecektir!) bu etiketin Click olayına cevap verme özelliğinden yararlanabiliriz. Daha sonra yazacağımız katmanGizliGoster() adlı fonksiyonumuz, onClick halinde, yapılmasını istediğimiz işlere dair parametreleri (argümanları, arguments) üçerli gruplar halinde alacak: birinci grup bilgi Netscape için kullanılacak, ikinci grup bilgi IE için kullanılacak, üçüncü grup bilgi ise hedef katmanın görünür hale mi geleceğini, yoksa gizleneceğini mi belirtmek için kullanılacak Burada görünmesini istediğimiz katman için "göster" anlamına Show, gizlenmesini istediğimiz katman için "gizle" anlamına Hide kelimelerini İngilizce kullanıyoruz; çünkü bu kelimeleri daha sonra fonksiyonda Netscape için komut olarak kullanacağız. Bir sekme tıklandığında, Javascript dört iş birden yapsın istiyoruz: Kapak ve diğer iki sekmenin

sayfaları gizlensin; tıklanan sekmenin sayfası gösterilsin. O halde, fonksiyona dört ayrı üçerli bilgi kümesini parametre olarak vereceğiz. Buraya gelmişken, diğer iki sekmenin DIV etiketlerini de yazalım:

```
<div id="Sekme2" style="position:absolute; left:169; top:95; width:95; height:46; z-
index:4; visibility: visible"><a href="#" onClick=
"katmanGizleGoster('document.layers[\'Sekme1Zemin\']',
'document.all[\'Sekme1Zemin\']','hide','document.layers[\'Sekme2Zemin\']',
'document.all[\'Sekme2Zemin\']','show','document.layers[\'Sekme3Zemin\']',
'document.all[\'Sekme3Zemin\']','hide','document.layers[\'Kapak\']',
'document.all[\'Kapak\']','hide')"><img src="sekme2.gif" width="95" height="46"
border="0"></a></div>
<div id="Sekme3" style="position:absolute; left:266; top:95; width:97; height:46; z-
index:5; visibility: visible"><a href="#" onClick=
"katmanGizleGoster('document.layers[\'Sekme1Zemin\']',
'document.all[\'Sekme1Zemin\']','hide','document.layers[\'Sekme2Zemin\']',
'document.all[\'Sekme2Zemin\']','hide','document.layers[\'Sekme3Zemin\']',
'document.all[\'Sekme3Zemin\']','show','document.layers[\'Kapak\']',
'document.all[\'Kapak\']','hide')"><img src="sekme3.gif" width="97" height="46"
border="0"></a></div>
```

Bu satırların hemen arkasından, sekmeler tıklandığında görüntülenecek olan sekme sayfalarını oluşturacak DIV etiketleri gelecektir. Onların kodları ise şöyle:

```
<div id="Sekme1Zemin" style="position:absolute; left:73px; top:141px; width:289px;
height:288px; z-index:6; visibility: hidden; background-image: url(sekme1_zemin.gif);
layer-background-image: url(sekme1_zemin.gif); border: 1px none
#000000"><p> </p><h2><b>Burada Sekme 1'in bilgileri var...</b></h2></div>
<div id="Sekme2Zemin" style="position:absolute; left:73; top:141; width:289;
height:289; z-index:7; visibility: hidden; background-image: url(sekme2_zemin.gif);
layer-background-image: url(sekme2_zemin.gif); border: 1px none
#000000"><p> </p><h2><b>Burada Sekme 2'nin bilgileri var...</b></h2></div>
<div id="Sekme3Zemin" style="position:absolute; left:73; top:141; width:289;
height:289; z-index:8; visibility: hidden; background-image: url(sekme3_zemin.gif);
layer-background-image: url(sekme3_zemin.gif); border: 1px none
#000000"><p> </p><h2><b>Burada Sekme 3'ün bilgileri var...</b></h2></div>
```

DIV'lerimizi tamamladığımıza göre HTML sayfamızda başa dönerek, fonksiyonumuzu yazabiliriz. Birinci nokta-noktalı yere şu kodu yazalım; sonra ayrıntılarına bakalım:

```
<script language="JavaScript">
```

```
<!--
function katmanGizleGoster() {
      var i, gorunurluk, parametreler, Nesne;
      parametreler = katmanGizleGoster.arguments;
      for (i=0; i<(parametreler.length-2); i+=3) {
           gorunurluk = parametreler[i+2];
           if (navigator.appName == 'Netscape' && document.layers != null) {
                Nesne = eval(parametreler[i]);
                if (Nesne) Nesne.visibility = gorunurluk;
           }
                else if (document.all != null) {
           if (gorunurluk == 'show') gorunurluk = 'visible';
           if (gorunurluk == 'hide') gorunurluk = 'hidden';
                Nesne = eval(parametreler[i+1]);
           if (Nesne) Nesne.style.visibility = gorunurluk;
           }
      }
}
//-->
</script>
```

katmanGizleGoster() fonksiyonu aldığı argümanlardan bir parametreler dizi-değişkeni yapıyor; ve oluşturacağı nesneyi bulunduğu Browser Netscape ise bunlardan "document.layers" diye başlayanlarını seçerek; bulunduğu Browser IE ise, "document.all" diye başlayan argümanları seçerek oluşturuyor; ve Browser IE ise ve hedef katman gösterilecekse "visibility" özelliği "visible," gösterilmeyecekse "hidden"; Netscape için bizim verdiğimiz Show ve Hide kelimeleri aynen kullanılıyor.

Kodları girdikten sonra sayfanızı sekmeler.htm adıyla kaydederseniz ve Browser'ınızda açarsanız, her sekme için verdiğiniz metnin görüntülendiğini, diğer sekmelerin gizlendiğini göreceksiniz:

<js007.tif> <js008.tif> <js009.tif>

Sonuç

Anlatacaklarımızın hepsi bu kadar; ama tabiî Javascript bu kadar değil. Özellikle Belge Nesne Modeli (DOM) konusunda, Javascript yoluyla sayfalarınıza veri bağlama konusunda, daha bir çok unsuru görmedik. IE5'in Browser dünyasına getirdiği yeni yeni olaylara nasıl Script yazılabileceğine değinmedik. Ama bu kitapçıkta amacımız Javascript'in bu ileri aşamalarına hazırlanmanız için gerekli temel bilgileri vermekti. Bu amaca ulaşmak için önemli bir nokta kaldı:hatasız Script yazmak.

İster başlangıç düzeyinde, ister ileri düzeyde, hangi düzeyde olursa olsun, hatasız Javascript programı yazmak için dikkat etmeniz gerekenleri belirtelim:

1. Browser'ın Javascript yorumlayıcısı (belki de şimdilik) programcının ne demek istediğini anlama imkanına sahip değildir; bu sebeple onun anlayacağı şekilde, yani nasıl yazılması gerekiyorsa, öyle yazın. Daktilo hatası yapmayın. Bir değişkeni nasıl tanımlamışsanız, baştan sona o yazılışla yazın. Bazı daktilo hataları HTML'e aittir; Javascript sizi uyarmaz bile. Hele daktilo hatası yüzünden HTML'in mantığı ve akışı bozulmuyorsa, Browser da hata mesajı vermez! Bu sebeple, programlarınızı sınadığınız Browser'a değil, kendinize güvenin ve kodlarınızı satır satır, kelime kelime okuyun.

2. Açtığınız etiketin kapanması gerekiyorsa kapatın. Özellikle </SCRIPT> etiketini unutmayın. Javascript programcının en çok hata yaptığı nokta büyüktür, küçüktür işaretleri arasında satırlar dolusu kod varsa, açtığı etiketi kapatmamaktır.

3. Parantezleri, özellikle fonksiyonların süslü parantezlerini kapatmayı unutmayın. Çift-tırnak ve tek-tırnak işaretlerinin kapatılması da çok kolaylıkla unutulur. Bir metnin içinde Javascript açısından ayraç işareti sayılan tırnakla, metne ait alıntı işareti sayılan tırnağı

birbirinden ayırmanız gerekir. Hatırlıyorsanız, Javascript'e "Bu sana ait değil; bu metnin parçasıdır!" demek istediğiniz tırnakları ters bölü işareti ile \" ve \' şeklinde yazıyoruz.

3. Tam veya kesirli sayı değişkenlerine alfanümerik (karakter, String) değişkeni muamelesi yapmayın. Örneğin, (.bold gibi) metin biçimlendirme metodları sayı tutan değişkenlere uygulanamaz; sayı olmayan değişkenler de içlerinde rakam olsa bile aritmetik işlemden geçirilemez. Sayı bekleyen fonksiyona karakter, karakter bekleyen fonksiyona sayı verirseniz; Javascript'in öfkesine sebep olabilirsiniz!

4. Javascript, işlem mantığı hatasını bulamaz. Mesela bir dörtgenin alanını bulurken, kenarlardan birinin yüzde 20'sini alıp, diğer kenarla çarpıyorsanız, Javascript'in "Dörtgen alanı öyle hesaplanmaz!" demesini boşuna beklersiniz. İşlemlerinizin mantık akışı ve matematik doğruluğunu sağlamak sizin görevinizdir.

5. Programın yazılma aşamasında değişkenlerin belirli bir noktada ne değeri aldığını Javascript'e uyarı kutularıyla size göstermesini bildirin. Sonra bu uyarı kutularını kaldırabilirsiniz. Örneğini gördüğümüz "window.onerror()" metodu da programın geliştirilme aşamasında hata yakalamaya yarar.

6. Kaynaklara başvurmaktan çekinmeyin. Javascript için en zengin kaynak, fabrikasıdır: http://home.netscape.com/computing/webbuilding/studio/scripting.html. Ayrıca adı her ne kadar Javascript değil de JScript bile olsa, Microsoft'un da zengin bir Javascript öğretim alanı vardır: http://msdn.microsoft.com/scripting/default.htm.

7. Sınayıp, yanılmaktan daha iyi öğrenme yöntemi yoktur; hata yapmaktan korkmayın. Sınayın; yanlışı düzeltin; bir daha sınayın.

Ve bol bol Script yazın.