ASP

İçin

Visual Basic Script

- Visual Basic Script
- Active Server Pages (ASP)
- MS Internet Explorer Nesne Modeli
- HTML

Selahattin Arslan
Kontrol ve Bilgisayar Mühendisi

Türkmen Kitapevi

Bu kitabı yazarken benden maddi ve manevi desteklerini esirgemeyen herkese teşekkür ediyorum.

İçindekiler

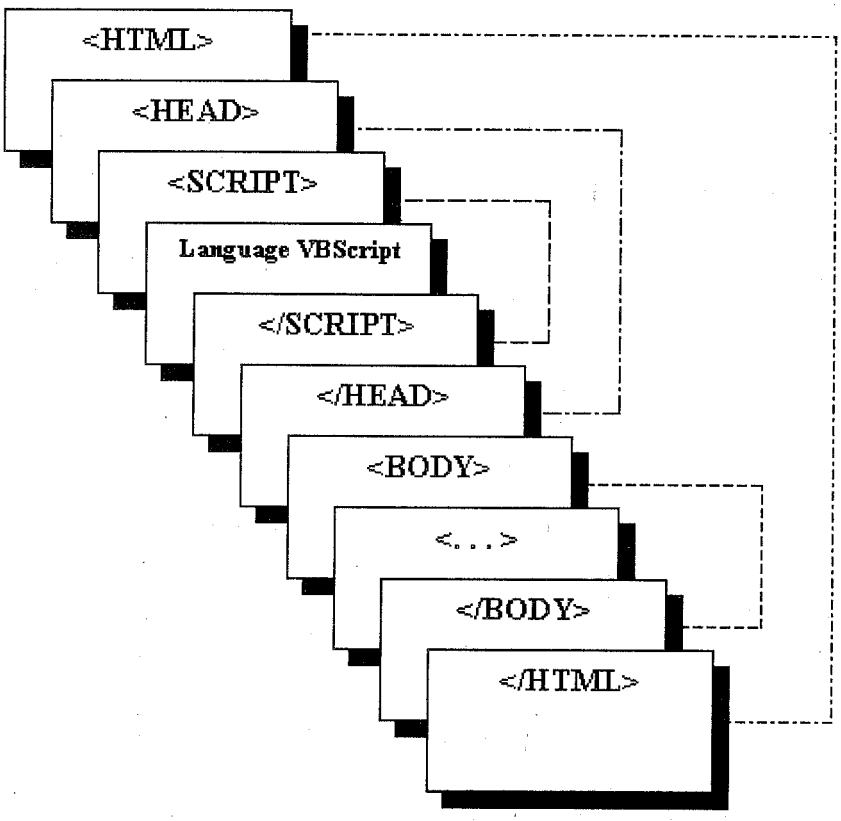
BÖLÜM 1
BİR

Giriş

- VBScript Program Yapısı
- VBScript ve Visual Basic
- ActiveX Kontrolleri ve Java Apletleri
- Program Yorumları

VBScript Program Yapısı

VBScript ile yazılmış bir program birçok yönden bir JavaScript programına benzer, en azından HTML dosyası içinde her iki script dilinin nasıl uygulanacağı konusunda benzerler. Prosedürler, bir isime atanmış olan sonlu sayıda kod bloğu olarak tanımlıdır, yani bir veya daha fazla script kodu bir isim altında toplanır ve bu kodlara verilen isime çağrı yapılarak erişilir, bunlar <HEAD> ve </HEAD> takıları arasındaki kısıma yerleştirilir. Prosedür içinde olmayan diğer kodlar ise dosya gövde kısmına yerleştirilir. Bu kodların VBScript desteği olmayan tarayıcılarda gösterilmesini engellemek istenirse o zaman VBScript kodları HTML yorum takıları arasına alınmak zorundadır. Aşağıdaki akış şeması herşeyi daha açık bir şekilde göstermektedir.



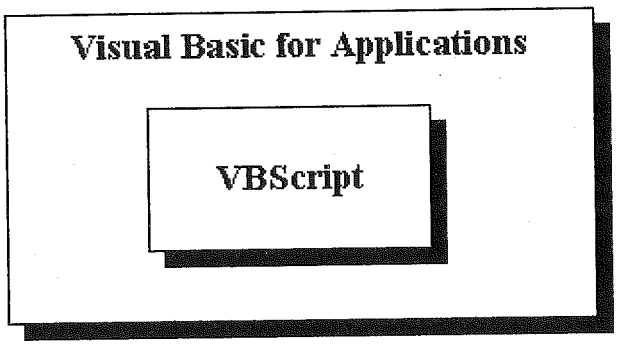
HTML içinde VBScript kodlarını eklemenin en çok tercih edilen yeri <HEAD>ve </HEAD> takıları arasıdır.

VBScript olan bir HTML dosyasının temel yapısı aşağıdaki gibidir.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript"
<!- -
' VBScript kod buraya yazılır.
- ->
</SCRIPT>
</HEAD>

<BODY>
<SCRIPT LANGUAGE="VBScript">
<!- -
' VBScript kod buraya yazılır.
- ->
</SCRIPT>
</BODY>

</HTML>
```

VBScript ve Visual Basic

Web sayfalarını geliştirmek için VBScript kullanarak başlayan programcılar daha önceden genellikle Visual Basic deneyimleri olur. Bu deneyimin ne kadarı VBScript için transfer edilebilir? Cevap epey çok olacaktır. VBScript dilinin Visual Basic dilinin bir alt kümesi olduğunu unutmayın, bundan dolayı neredeyse VBScript içindeki herşey Visual Basic dilinin bir parçasıdır. Ana fark kaybedilen şeyde yatar: Visula Basic dilinde olan yeterince çok şey VBScript dilinin bir parçası değildir.



VBScript dili VBA'nın bir alt kümesidir

Visual Basic ürün listesinde gerçekte üç adet üye vardır. En üst kısmında Visula Basic, dağıtık uygulamalar oluşturmanıza izin veren güçlü client/server yetenekleri ile tam bir

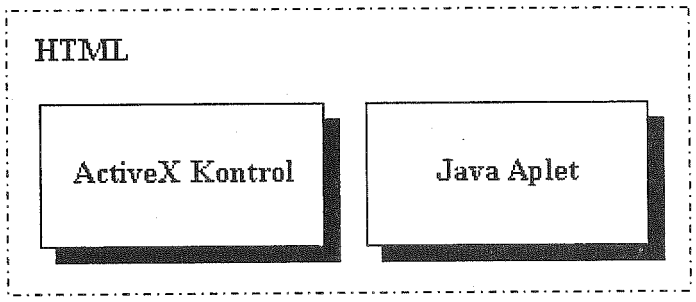
uygulama geliştirme aracına sahiptir. Sonraki, VBA (Visual Basic for Applications), Microsoft Windows uygulama script dilidir. Bu bir Visual Basic alt kümesidir, burada client/server veri erişimi, dağıtık hesaplama, ve takım kaynak kod kontrolü. En sonuncusu ise, özellikle Internet kullanımı için tasarlanmış olan Visula Basic for Applications alt kümesidir.

Bunların Visual Basic ile benzer olması, birçok farklılığı görmek açısından faydalı olabilir. Aşağıdaki tablo Basic dilinin iki farklı versiyonları arasındaki çok daha önemli birçok farklılığı tanımlar. Siz VBScript dilinden çıkarılmış olan Visula Basic dilinin bir çok yeteneğini görebilirsiniz. VBScript dilinin belirli bir görev için tasarlandığını unutmayın. Potansiyel olarak dosya erişimi gibi güvensiz yetenekleri alınmıştır, ve diğer özellikleri, Visual Basic zengin veri tipleri basitlik açısından azaltılmıştır.

Visual Basic Script ve Visual Basic for Applications arasındaki farklar

Kategori	VBScript	VBA
Diziler	Dim, Static, ReDim, Erase	Option Base, Lbound <>0
DLL çağırma	Hayır	Evet
Akış kontrol	Do...Loop For...Next For Each...Next While...Wend If...Then...Else	DoEvents GoSub...Return GoTo Satır numaraları ve etiketler On Error
Veri tipleri	Variant	Diğerleri (Boolean, Integer, gibi.)
Hata yakalama	On Error Resume Next Err Object	Erl, Error, Error$ On Error...Resume Resume, Resume Next
Dosya işlemleri	Yok	Open, Write #, gibi.
Grafik	Yok	Circle, Line, Pset, gibi.
Yapılar	Hayır	Type...End Type
Sınıf oluşturma	Hayır	Dim x as New... Set x = New... With ... End With

ActiveX Kontrolleri ve Java Apletleri



HTML kodları içine hem ActiveX kontrol hemde Java Apletler gömebilirsiniz

Siz kuşkusuz Java apletleri ve ActiveX kontolleri hakkında birşeyler işitmiş olmalısınız, bunlar özellikle bizim web kullanma yöntemimizin nasıl evreler geçirdiğini varsayarak şu anki hallerini almışlardır. Bu karmaşık nesneler (komponent) VBScripte nasıl bağlıdır? ActiveX ve Java aynı amaç için tasarlandı, amaç web kullanıcılarına ek fonksiyonellik sağlamaktır. Fakat bunlar farklı yönlerden giderler. İlk önce onların fonksiyona nasıl gereksinim duyduklarına bir göz atalım, sonra kullanıcılarına daha çok değer sağlayan web sayfaları oluşturan farklı kişileri göreceğiz. Bu aşırı derecede rekabete dayanan bir iştir , ve eğer bir başkasının sayfası sizinkinden daha çok olanak ve göz kamaştırıyorsa o zaman sizi takip edenleri peşinizde tutacağınızı beklemeyin.

İstemci bilgisayarın birçok işi , JavaScript ve VBScript geliştirmenin arkasındaki amaçlardan biri, yapmasını ister. Komutlar (script kodu) HTML dosyasının bir parçasıdır, ve istemci bilgisayar bunları işler. ActiveX ve Java apletleri ile, istemci bilgisayara indirilir ve yürütülür. Burada dikkat etmeniz gerken şey indirilen script kodları istemci tarafı için yazılan kodlardır, sunucu tarafı için yazılan kodlar istemci bilgisayara gelmez doğrudan sunucuda yürütülür.

ActiveX ve Java aynı amaç için olmalarına rağmen, onların sunumlarında farklılıklar vardır. Java yeni bir nesneye yönelik programlama dili olup Sun Microsystems tarafından geliştirilmiştir. ActiveX ise Microsoft'un nesne bağlama ve gömme (OLE) kontrolleri için yeni bir isimdir. Çok yönden farklılıklara sahip olmaları yanında, bu iki standard indirme işleminde uygun olan küçük bileşenler için onları optimize eden aynı imkanları paylaşırlar. JavaScript ve VBScript size web sayfalarınızda hem Java apletleri hemde ActiveX kontrolleri kullanmanıza izin verir.

VBScript ve JavaScript arasında büyük bir benzerlik vardır. Gerçekte, farklılıklardan çok benzerlikleri vardır denebilir. Bunlardan sonra, her ikiside aynı amaçlı olup bilgisayar programlama dili olarak geliştirilmişlerdir. Böyle olmasına rağmen tarihçeleri farklıdır. JavaScript C ve C++ dilleri baz alınarak geliştirilmiştir, VBScript ise Basic dili baz alınarak geliştirilmiştir.

Tabiki, aynı dil olmadığı göz önünde bulundurulduğunda iki dil arasında epey bir farklılık da olacaktır.

Program Yorumları

Herhangi bir standard programlama dili gibi, VBScript dilide sizin yorum eklemenize izin verir, eklemiş olduğunuz text bilgi çalışma esnasında yorumlanır'ken gözardı edilir, fakat bu bilgiler doküman ve açıklamalar için birçok bilgi verir. VBScript iki farklı şekilde yorum eklemeye izin verir: Bunlardan birincisi Rem komutu kullanılarak ve tıknak işareti (') kullanılarak olur.

Kural çok basit. Rem veya tırnak işareti ile başlayan herhangi bir satır tarayıcı tarafından bir yorum olarak yorumlanacaktır. Siz satır sonuna da bu iki yoldan birini kullanarak yorum ekleyebilirsiniz.

Rem kullandığınız zaman, Rem yorumu ve satır sonu arasında iki nokta üst üste (:) olmak zorunda.

Aşağıda birkaç tane VBScript yorumu verilmiştir:

```
' Bu bir yorumdur

Rem Bu bir yorumdur

Document .Write "Ne haber "  : Rem Buda bir yorum

Document .Write "Nasilsiniz"  ' Buda başka bir yorum
```

BÖLÜM 2

İKİ

VBScript Temelleri

- VBScript ve ASP
- VBScript Nasıl Yazılır
- İlk VBScript Kodunuz
- <SCRIPT> Etiketi
- Web Sayfasına VBScript Ekleme
- HTML Dökümanı Oluşturma
- VBScript Eklemenin Tercih Edilen Yolu
- Script Desteği Olmayan Tarayıcılarda Durum
- Scriptlere Yorum Ekleme
- VBScript Kodlama Dönüşümleri

VBScript Temelleri

VBScript ve ASP

VBScript programlama dili Visual Basic Scripting Edition olarak da adlandırılır. Bu dil herkesin bildiği ve çok kullandığı Visual Basic dilinin içinde olan bir alt versiyonudur. VBScript tanımları Visual Basic tanımlarına benzer, bundan dolayı belli kuralları ve belirtimleri oluşmuş yerleşik bir dildir. VBScript dilinin yanlızca Microsoft Internet Explorer tarayıcısı tarafından desteklenmesi onun Internet üzerindeki kullanımını kısıtlamıştır, fakat son yıllarda en çok kullanılan tarayıcı Internet Explorer olduğu gözlenmektedir. Bu da VBScript kullanımının artacağını ve yakın bir zamanda popülaritesinin artacağını göstermektedir. Diğer tarayıcıların (Netscape Navigator gibi) VBScript dilini desteklememesi İstemciler tarafında VBScript ile script yazmayı kısıtlayabilir fakat Microsoft Internet Informatin Server ve uyumlu üçüncü parti Web Sunucular için Sunucu tarafı script yazmak için çok rahat bir şekilde VBScript kullanabilirsiniz. JavaScript tarayıcıların varsayılan script dili olmasından dolayı tüm tarayıcılar tarafından desteklenir. VBScript ile JavaScript ile yapabileceğiniz şeyleri Intranet üzerinde yapabilirsiniz, zaten şu an için VBScript Intranet uygulamaları için çok kullanılmaktadır. Aktif Sunucu Sayfaları (ASP) varsayılan dili VBScripttir fakat JavaScripti dilini de ASP uygulamaları oluşturmak için kullanabilirsiniz. Bir HTML sayfasını oluşturan tarayıcı döküman nesne modelini (DOM) VBScript ile de kullanabilirsiniz. VBScript Microsoft Windows İşletim Sistemleri ortamları için geçerli olmasından dolayı diğer platformlarda kullanamazsınız.

Web üzerinde ASP hakkında değişik şekillerde birşeyler işitmiş olmalısınız, ve şimdi siz ne olduğunu merak ediyorsunuz, ve onun sizin için ne yapabileceğini düşünüyorsunuz. Microsoft firması ASP yi 1996 yılında piyasaya sürdü. ASP ilk olarak Internet Information Server 3.0 ile kullanılarak ortaya çıkmıştır. O açık teknoloji çalışma çerçevesine sahip olduğu için birçok dil kullanılarak oluşturulabilir. Karmaşık programları, veritabanlarını ASP kullanarak web sayfalarımıza kolay bir şekilde bağlayabiliriz. ASP nin özünde VBScript dilini buluruz. Bu dil ASP nin tüm gücünü web sayfalarımıza taşımamıza yardım eder, ve çok zaman insanlar arasında ASP ve ASP kodları ile konuşur, onlar gerçekten VBScript kodları demektir. VBScript ASP nin varsayılan dilidir. ASP sunucu tarafında çalışır, yani yazmış olduğumuz kodlar web sunucusu üzerinde çalışması gerektiğini belirtir, bundan dolayı web sayfası fonksiyonları tarayıcının ne olduğunu pek dikkate almaz.

Mayıs 2000 de, Microsoft dünyada 800.000 ASP geliştiricisinin Aktif Sunucu Sayfalarını hızla büyüyen bir teknoloji yapıyor olduğunu tahmin ediyordu.

ASP sayfalarımızdaki HTML kodları ile VBScript kodlarını birbirinden ayırmak için ASP.DLL 'ye HTML içinde ASP kodun nerede başlayıp nerede bittiğini söylememiz gerekir. Bunu söylemek için "<%" ve "%>" işaretleri arasına alırız. ASP motoru bu işaretleri gördüğünde bunların bir text bilgi değil çalıştırılması gereken program kodları olduğunu anlar. ASP motoruna kullanılan script dilini belirtmek için LANGUAGE

etiketini kullanırız. VBScript ile yazdığımız kodları HTML sayfamızın heryerinde kullanma imkanımız var, örneğin, bir tablo hücresi içine bir değişken değerini bile aktarabiliriz, bir matematiksel işlem sonucunu bir form elemanı ile etkileşim içine sokabiliriz, işte VBScript ile yapabileceğimiz çok şey olduğunu gördük.

ASP içinde kullandığımız dilin VBScript olduğunu belirtmek için aşağıdaki gibi bir kod kullanırız:

```
<% @LANGUAGE =VBScript %>
```

ASP yazmak için JavaScript, JScript, PerlScript, ve PHPScript dillerinide kullanabileceğinizi unutmamanız gerekir.

VBScript Nasıl Yazılır

VBScript kullanarak kod yazmak için ihtiyacınız olan nedir?

ASP nin tam gücünü ve bütün imkanlarını elde etmek için, sizin bir MS Windows NT 4.0 işletim sistemi ve IIS 4.0 üzerinde yüklü olan bir makine veya Windows 2000 Server (IIS 5.0). Siz Windows 95/98/ME işletim sistemleri üzerinde çalışan PWS (Personel Web Server) kurulu bir makinede ASP sınırlı bir versiyonunu kullanabilirsiniz.

VBScript yazmak için özel bir editöre gerek yoktur. Herhangi bir text düzenleme editöründe (Notepad, Wordpad gibi) VBScript kodlarımızı yazabilirsiniz ve derleme olmaksızın HTML sayfalarınızda kullanabilirsiniz. Kodlama yaparken dil kurallarına uymak gerekir, özel amaçlı olarak yazılmış bazı editörler kurallara uyumadığınız zaman sizi uyaracaktır.

VBScript dilini destekleyen bir Internet tarayıcı Internet Explorer versiyon 3 veya daha üzeri olabilir. IE versiyonu yükseldikçe yetenekleride artmaktadır.

Eğer VBScript dilini ASP yazmak için kullanıyorsanız o zaman bir kısıtlama ile karşı karşıyasınız demektir. Nedir bu kısıtlama? ASP dosyaları genellikle sunucu tarafında çalıştığı için MsgBox , ve InputBox fonksiyonlarını bu tür dosyalar içinde mesaj gösterme, veya kullanıcıdan bir girdi almak için kullanamazsınız, fakat istemci tarafında çalışacak bir web sayfamıza etkileşim katmak için bunları kullanmanızda hiçbir sakınca yoktur.

VBScript komutları, anahtar sözcükler, ve değişken adlarının büyük veya küçük olması fark etmez, yani siz bunları büyük veya küçük harfler ile yazabilirsiniz. Örneğin, FOR ile for aynı şeyi gösterir.

Eğer yazdığımız kod bir satıra sığmıyorsa o zaman bir alt çizgi (_) koyduktan sonra bir alt satırdan aynı komutun kalan kısmını yazmaya devam ederiz, ama tavsiye edilen mümkün olduğunca kodlarımızı tek bir satırda tamamlamak,çünkü bu program kodlarımızın okunabilirliğini artıracaktır.

İlk VBScript Kodunuz

Öncelikle bir editör ortamında gerekli olan kodları yazarak saklamak zorundayız.Yazmış olduğunuz dökümanı bir tarayıcı içinde açınız. Eğer yazmış olduğunuz editör size ilgili dökümanı hemen tarayıcı içinde açmanıza izin veriyorsa işiniz kolay demektir. Ama NotePad gibi text editörler kullanıyorsanız o zaman gerekli işlemleri manuel olarak yapmanız gerekecektir. Bir dökümanı bir kez açtıktan sonra, sonraki versiyonlar için yenilemek yeterli olacaktır. Birçok tarayıcıda yenileme işlemi için F5 tuşu kullanılır.

Dökümanı istediğiniz gibi çalışıyor olup olmadığını deneyebilirsiniz. Eğer çalışıyorsa sorun yok demektir, aksi durumda debug programları kullanarak hatayı bulmaya çalışmalısınız.

Microsoft Internet Explorer ile, siz aşağıdaki HTML kodu tarafından üretilen sayfayı gösterebilirsiniz. Eğer sayfa üzerindeki düğmeye tıklarsanız, aksiyonda VBScripti görebilirsiniz.

Aşağıdaki örnekte görüldüğü gibi VBScript kodlarımızı <SCRIPT> ve </SCRIPT> takıları arasında belirtiyoruz, ayrıca kullanmak istediğimiz Script dilini belirtmek için LANGUAGE özelliğini kullanmalıyız. Burada VBScript dilini kullanmak için <SCRIPT LANGUAGE="VBScript"> tanımlaması yapılmıştır.

HTML sayfası içinde JavaScript dilini kullanmak için yine aynı tanımlamaları yapmak zorundayız. <SCRIPT LANGUAGE="JavaScript"> şeklinde bir tanımlama yaparız.

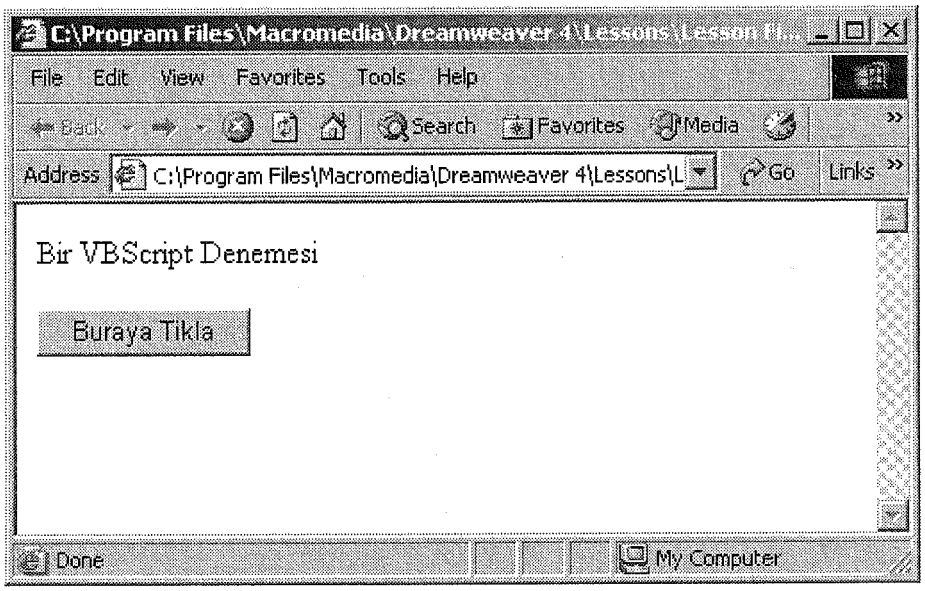
Visual Basic Script için tanımlama şekli:

```
<SCRIPT LANGUAGE="VBScript">

<!--

  Sub dugme_OnClick

        MsgBox "VBScript kodlarını buraya yazmalısınız."

  End Sub

-->

</SCRIPT>
```

VBScript bir script yazma dilidir. Bunun anlamı o tek başına bir şey yapamaz, ancak bir web sayfası ile bir anlam kazanır. HTML dökümanı içinde SCRIPT takısı içinde VBScript yazabilirsiniz. Aşağıda basit bir örnek verilmiştir.

```
<HTML>

<BODY>

<P>Bir VBScript Denemesi<BR>

<FORM>

<SCRIPT LANGUAGE="VBScript"></SCRIPT>

<INPUT TYPE=BUTTON VALUE="Buraya Tıkla" NAME=dugme1
OnClick="Alert 'Deneme bir mesaj.'">

</FORM>

</BODY>

</HTML>
```

Programın tarayıcı üzerindeki görüntüsü aşağıdaki gibidir.



Sayfa üzerindeki düğmeye tıkladığınızda izleyen mesaj kutusu ekrana gelir.

Yazmış olduğumuz basit programın çıktısını görmüş oldunuz. Bu şekilde statik web sayfasına bir aksiyonun nasıl kazandırılacağını gördünüz.
Şimdi, bu sayfayı veya herhangi bir sayfayı script ile birlikte yapmak istediğinizde, birkaç adım izlemek zorundasınız. Bu işlem birkaç denemeden sonra sizin için alışılagelmiş bir işlem halini alacaktır.

```
<HTML> 'HTML ile başlar

<BODY> 'Döküman gövdesi başlangıcı

<P>Bu bir VBScript denemesi.<BR> 'Biraz text bilgi

<FORM> 'Form başlangıcı (tüm düğmeler, text alanlar, vs. form içinde olmak
zorundadır)

<SCRIPT LANGUAGE="VBScript"></SCRIPT>

<INPUT TYPE="BUTTON" VALUE="Click Here" NAME="Button1"
'Bir düğmeye tıkla OnClick="Alert 'Deneme bir mesaj.'"> 'Mesajı göster

</FORM> 'Form sonu

</BODY> 'Döküman sonu

</HTML> 'HTML ile sona erer
```

Bu örnekte, gösterilen doküman içinde VBScript çağırma yöntemlerinden yanlızca biridir. Bu metod için, script yazma dili çıkarımla belirlenmek zorundadır. Burada önce SCRIPT takısı kullanılır.

Kodun tüm kısmı tek tırnak işaretleri ile çevrelendiğine dikkat ediniz. Eğer çift tırnak işareti ilede çevreleseydiniz ve iç kısımda yanlızca tek tırnak işaretleri kullansaydınız, o zaman yine kodunuz çalışacaktı.

Java Script için tanımlama şekli:

```
<SCRIPT LANGUAGE="JavaScript">
```

```
<!--
    alert("JavaScript kodlarını buraya yazmalısınız.")
-->
</SCRIPT>
```

Eski tip tarayıcılar scriptlerimizi tanımayacağından dolayı kodlarımızı gizlemek için

```
<--
        .....
//-->
```

işaretleri arasına yazarız.

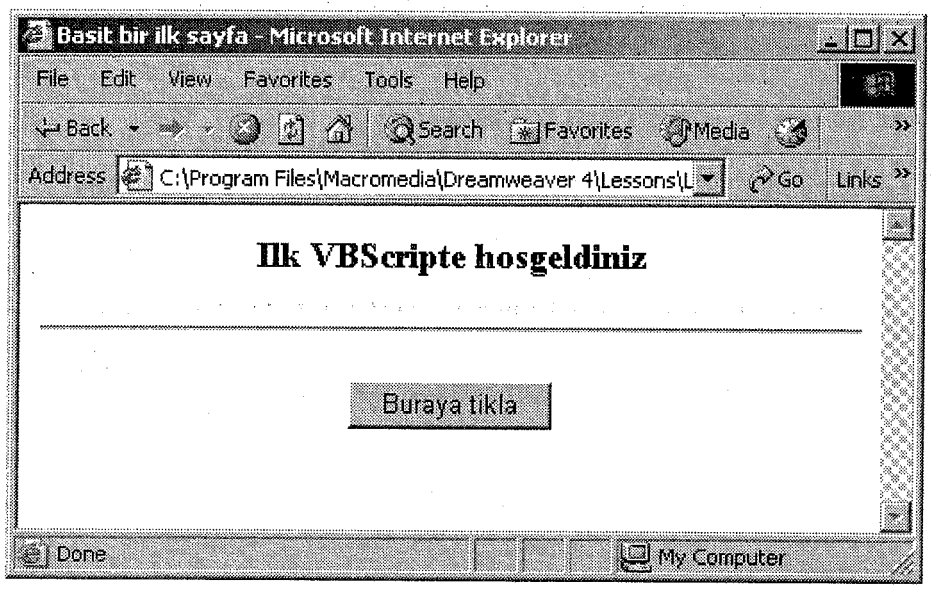
VBScript içinde bir düğmeye bir olay eklemek için aşağıdaki örnekte olduğu gibi dugme_OnClick , düğme adından sonra alt çizgi ve sonra olay kotarıcı adı yazılır.

İşte istemci tarafı için ilk VBScript örneğimiz:

```
<HTML>
<HEAD>
<TITLE>Basit bir ilk sayfa</TITLE>
<META http-equiv="content-type" content="text/html"; charset=ISO-8859-9">
<SCRIPT LANGUAGE="VBScript">
<!--
  Sub dugme_OnClick
      MsgBox "VBScripte merhaba."
  End Sub
-->
</SCRIPT>
</HEAD>
<BODY>
```

```
<CENTER>
<H3>İlk VBScripte hoşgeldiniz</H3><HR>
<FORM>
<INPUT NAME="dugme" TYPE="BUTTON" VALUE="Buraya tıkla">
</FORM>
</CENTER>
</BODY>
</HTML>
```

Örnek programın Internet Explorer üzerindeki görünümü aşağıdaki gibidir.



Sayfa üzerindeki düğmeye tıkladığınızda aşağıdaki mesaj kutusu ekrana gelir.

Internet Explorer sayfayı okuduğu zaman, o <SCRIPT> etiketini bulur, tanır VBScript kodu varmı diye, ve kodu saklar. Düğmeye tıkladığınızda, Internet Explorer düğme ve kod arasındaki bağlantıyı kurar ve prosedürü koşturur.

<SCRIPT> etiketlerindeki Sub prosedürü bir olay prosedürüdür. Prosedür adı için iki kısım vardır: düğmenin adı, dugme (<INPUT> etiketinin NAME özelliğinden), ve bir olay adı, OnClick. İki ad bir alt çizgi (_) ile birleştirilir. Her hangi bir zamanda bir düğme tıklandığında, Internet Explorer karşı gelen dugme_onClick olay prosedürünü arar bulur ve koşturur.

Internet Explorer Internet Microsoft Web sitesinde (http://www.microsoft.com) bulunabilen Internet Explorer'daki nesne modeli dökümanındaki form kontrolleri için mevcut olayları tanımlar.

Sayfalar kontrol ve prosedürlerin kombinasyonlarınıda kullanabilir. VBScript ve formlar kontroller arasındaki birkaç basit etkileşimi gösterir.

Önceki yol daha basit ve genel olmasına rağmen, siz VBScript kodu diğer iki şekildede olaylara iliştirebilirsiniz. Internet Explorer tanımlanan kontrol içinde satır içi kodun kısa kısımlarını eklemenize izin verir.

<INPUT> etiket bir düğmeye tıkladığınızda önceki kod örneği gibi aynı aksiyonu yapar:

```
<INPUT NAME="Dugme1" TYPE="BUTTON" VALUE="Buraya tıkla"
                    OnClick='MsgBox "Merhaba, Ya bu nasıl?"'>
```

Fonksiyonların kendi kendini çağırması tek tırnak işaretleri arasına alınır, ve MsgBox fonksiyonu için string çift tırnak işaretleri arasına alınır. Siz çok sayıda yapıları aralarına iki nokta üst üste (:) koyarak yapıları ayırarak kullanabilirsiniz.

Siz belirli bir kontrol için yanlızca özel bir olaya başvuru yapsın diye bir <SCRIPT> etiketi de yazabilirsiniz.

```
. . .

<SCRIPT LANGUAGE="VBScript" EVENT="OnClick" FOR="dugme">

<!--

  MsgBox "Merhaba."

-->

</SCRIPT>

. . .

veya

<INPUT TYPE="BUTTON" VALUE="Tıkla" NAME="dugme">
```

```
<SCRIPT FOR="dugme" EVENT="OnClick" LANGUAGE="VBScript">
<!--
Alert "İşte bir mesaj daha."
-->
</SCRIPT>
```

<SCRIPT> etiketi zaten olay ve kontrolü belirttiği için, siz Sub ve End Sub yapılarını kullanmazsınız.

<SCRIPT> Etiketi

JavaScript ve VBScript gibi script yazma dilleri, HTML ye bir uzantı olarak tasarlanır. Web tarayıcıları scriptleri web dökümanının bir parçası olarak alırlar. Scriptleri almak ayırmak ve işlemek sorumluluğu tarayıcılara aittir. HTML içine script eklemek için <SCRIPT> etiketi dahil edilmiştir.

<SCRIPT> ve </SCRIPT> elemanını bir HTML sayfası içine VBScript kodlarını eklemek için kullanabilirsiniz.

VBScript kod çift <SCRIPT> ve </SCRIPT> etiketleri içinde yazılır. Örneğin, verilen bir tarih ile şu anki tarih arasındaki farkı bulan ve karşılaştırma sonucunun 5 den büyük olup olmadığını test etmek için bir prosedür aşağıdaki gibi olabilir:

```
<SCRIPT LANGUAGE="VBScript">
<!--
  Function TarihFarkı( Tarih )
    TarihFarkı = (CDate( Tarih ) - Now()) > 5
  End Function
-->
</SCRIPT>
```

Now() fonksiyonu şu anki tarihi verir.

Başlangıç ve sondaki <SCRIPT> etiketleri kodu çevreler. LANGUAGE özelliği script yazım dilini gösterir. Siz dili belirtmek zorundasınız çünkü tarayıcılar diğer script

dillerinide kullanır. DagıtımTarihi fonksiyonu yorum etiketleri (<!- - ve - - >) içine gömülmüştür. Bu tarayıcıların kodu göstermeden <SCRIPT> etiketini anlamayan tarayıcılardan engeller.

Örnek genel bir fonksiyon olduğu için, o herhangi bir belirli form kontrola bağlanamaz. Siz bu kodu sayfanın HEAD kısmına dahil edebilirsiniz:

```
<HTML>

<HEAD>

<TITLE>Sipariş Verme</TITLE>

<SCRIPT LANGUAGE="VBScript">

<!--

  Function DagitimTarihi( Tarih )

    DagitimTarihi = (CDate( Tarih) - Now()) > 2

  End Function

-->

</SCRIPT>

</HEAD>

<BODY>
...
```

Siz <SCRIPT> bloklarını bir HTML sayfasının herhangi bir yerinde kullanabilirsiniz. Siz onu her iki BODY ve HEAD kısımları içine koyabilirsiniz. Bununla birlikte, siz muhtemelen tüm genel amaçlı script yazım kodlarını HEAD kısmına tüm kodları birlikte tutmak için koymak isteyebilirsiniz. Kodunuzu HEAD kısmında tutmak BODY kısmı içinden çağrılmadan önce hepsinin okunması ve çözülmesini sağlar.

Bu kurala önemli bir istisna formunuzdaki nesnelerin olaylarına cevap vermek için formlar içinde satır için script yazımını sağlamak istiyor olabilmeniz. Örneğin, siz bir form içinde bir düğmeye tıkladığınızda cevap vermesi için script kodunu gömebilirsiniz:

```
<HTML>

<HEAD>

<TITLE>Test Düğme Olayları</TITLE>
```

```
</HEAD>

<BODY>

<FORM NAME="Form1">

  <INPUT TYPE="Button" NAME="dugme" VALUE="Click">

  <SCRIPT FOR="dugme" EVENT="onClick" LANGUAGE="VBScript">

    MsgBox "Düğmeye Basıldı!"

  </SCRIPT>

</FORM>

</BODY>

</HTML>
```

Sizin kodunuzun çoğu ya Sub yada Function prosedürleri içinde görünecektir ve yanlızca sizin kodunuz tarafından belirtildiğinde çağrılacaktır. Bununla birlikte, siz VBScript kodu prosedürler dışına yazabilirsiniz, fakat hala bir SCRIPT bloğu içindedir. Bu kod yanlızca bir kez sayfa yüklendiği zaman yürütülür. Bu size veriyi başlangıç değerine ayarlama veya dinamik olarak yüklendiğinde Web sayfanızın görünümünü değiştirmenize izin verir.

Web Sayfasına VBScript Ekleme

Bir Web sayfasına herhangi bir dildeki program kodlarını eklemek için SCRIPT etiketini kullanmamız gerektiğinden bahsetmiştim. Bu eleman yazmış olduğumuz script kodlarının nerde başladığını ve nerde bittiğini tarayıcıya bildirmek için kullandığımız bir HTML elemanıdır. Bu etiket içinde kullanılan herşey script desteği olan tarayıcı tarafından bir HTML kodu değil bir script kodu olarak yorumlanır. HTML kod yazarken büyük-küçük harf ayırımı yapmayabilirsiniz, fakat scriptler genelde harflere karşı duyarlıdır, bundan dolayı dikkatli olmanız gerekir.

<SCRIPT> elemanı içinde VBScript kod yazmak için LANGUAGE=VBScript şeklinde belirtmemiz gerekir. Bu belirtim kullanılacak olan script dilinin VBScript olduğunu gösterir.

Örneğin, bir HTML dökümanı oluşturularak komut düğmesi tarafından üretilen click olayına cevap verebilecek basit bir script eklenmiştir. Sizde böyle bir HTML dökümanı içine script ekleyerek test edebilirsiniz.

HTML Dökümanı Oluşturma

Bir text editörü (Notepad gibi) ile HTML kodlarını yazabilirsiniz

```
<HTML>

<HEAD>

<TITLE>VBScript İle Çalışma</TITLE>

</HEAD>

<BODY>

<H1>İşte Bir VBScipt Örneği</H1>

<P>VBScript kullanarak web sayfanıza etkileşim katabilirsiniz.
 Ne demek istediğimi düğmeye tıkladığınızda göreceksiniz</P>

<FORM NAME="form1">

<INPUT TYPE="Button" NAME="dugmeTikla" VALUE="Buraya Tıkla">

</FORM>

</BODY>

</HTML>
```

Dosyayı saklayıp Internet Explorer içinde test edin.
Sayfa üzerindeki düğmeye tıkladığımızda bir şey olmayacağını göreceksiniz, çünkü bu düğme üzerinde meydana gelecek bir olaya cevap verebilecek herhangi bir VBScript kod eklenmemiştir.

Dosyaya aşağıdaki kodlar eklenerek yeni hali elde edilmiştir. Burada VBScript kod eklendiği için artık web sayfamızda düğmeye tıklama yapıldığında bir etkileşim olacaktır.

```
<HTML>

<HEAD>

<TITLE>VBScript İle Çalışma</TITLE>

</HEAD>

<BODY>
```

```
<H1>İşte Bir VBScipt Örneği</H1>

<P>VBScript kullanarak web sayfanıza etkileşim katabilirsiniz.
  Ne demek istediğimi düğmeye tıkladığınızda göreceksiniz</P>

<FORM NAME="form1">

<INPUT TYPE="Button" NAME="dugmeTikla" VALUE="Buraya Tıkla">

<SCRIPT FOR="dugmeTikla" EVENT="onClick" LANGUAGE="VBScript">

MsgBox "VBScript ile basit bir aksiyon"

</SCRIPT>

</FORM>

</BODY>

</HTML>
```

Dosyayı saklayıp Internet Explorer içinde test edin.

Yazdığımız VBScript ve Html kodları birlikte nasıl Çalışır ?

Eklediğimiz üç satır koda bir göz atalım. Biz VBScript kodun ne yaptığını ve HTML kodu içinde nasıl sunulduğunu bilmek isteriz. İlk satır bir scripti tanımlar. FOR argümanı bu scriptin HTML de <INPUT> takısı ile verdiğimiz dugmeTikla adlı komut düğmesi için olduğunu belirtir. EVENT argümanı bu scriptin düğmeye tıklandığında çalışması gerektiğini belirtir. LANGUAGE argümanı bunun bir VBScript modülü olduğunu ifade eder.

```
<SCRIPT FOR="cmdClickMe" EVENT="onClick" LANGUAGE="VBScript">
```

İkinci satır HTML dökümanı içindeki VBScripti ifade eder. MsgBox fonksiyonu basit bir şekilde bir mesaj diyalog kutusunu gösterir. Üçüncü satır scriptin sonunu işaretler.

Bir önceki kısımda, tanımlı komut düğmesi olan HTML takısından sonra sağ tarafa VBScript modülünü koyduk. Bu modül fonksiyonel iken, o yaklaşım tercih edilmez. HTML kendi kendine tüm takıları ve text bilgileri alır. Bunun orta kısmına VBScript eklemek ile daha karmaşık bir yapı oluşturur. Daha organize olmuş bir seçenek tüm scriptinizi HTML dökümanı ile birlikte yerleştirmektir.

VBScript Eklemenin Tercih Edilen Yolu

İkinci kısımdaki oluşturduğumuz HTML dökümanını yeniden açarız, eğer gerekli ise , ve eklediğimiz satırları geri alırız:

```
<SCRIPT FOR="dugmeTikla" EVENT="onClick" LANGUAGE="VBScript">
        MsgBox " VBScript ile basit bir aksiyon."
</SCRIPT>
```

Dökümanı aşağıdaki satırları ekleyerek yeniden düzenleyin:

```
<HTML>
<HEAD>
<TITLE>VBScript Ekleme</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub dugmeTikla_OnClick
MsgBox " VBScript ile basit bir aksiyon."
End Sub
-->
</SCRIPT>
</HEAD>
<BODY>
<H1>İşte Bir VBScipt Örneği</H1>
<P>VBScript kullanarak web sayfanıza etkileşim katabilirsiniz.
  Ne demek istediğimi düğmeye tıkladığınızda göreceksiniz</P>
<FORM NAME="form1">
<INPUT TYPE="Button" NAME="dugmeTikla" VALUE="Buraya Tıkla">
</FORM>
</BODY>
</HTML>
```

Dosyayı saklayıp Internet Explorer içinde test edin. Sonuç sayfası aşağıdaki gibi olmalı.

Sayfa üzerindeki düğmeye tıkladığınızda aşağıdaki mesaj kutusu ekrana gelir.

Nasıl Çalışır?

Bu ikinci metod önceki örnekteki gibi aynı <SCRIPT> takısı ile başlar. Bu scriptin orta kısmında sayfamız için fonksiyonellik sağlayan üç satır vardır. İlk satır dugmeTikla_OnClick adlı sub prosedürü tanımlar. Her dugmeTikla kontroluna tıklandığında bu yürütülecektir. Prosedürün bu tipi bir olay prosedürüne işaret ettirilir. Bu olay ile ilişkilendirdiğimiz prosedür düğmeye her tıklandığında yürütülür.

Sub dugmeTikla_OnClick

İkinci satırda ise MsgBox fonksiyonunu yine buluruz, üçüncü satır ise sub prosedürün bitimini gösterir.

Script Desteği Olmayan Tarayıcılarda Durum

Tüm tarayıcılar script yazma dillerini desteklemez. Birkaçı yanlızca JavaScripti destekler. VBScript ise yanlızca Microsoft's Internet Explorer tarafından desteklenir. Siz bu durumda yazdığınız scritpte ne olacağı hakkında kaygı duyabilirsiniz. Genellikle tarayıcılar text ile sık sık yaptıkları şeyi yapacaklardır, onlar scriptinizi web sayfasının bir parçasıymış gibi göstereceklerdir. Açık bir şekilde, bu beklediğiniz sonuç değil. Bu problemler için basit bir yol scriptlerinizi yorum takıları (<!- - ve - ->) arasına koymak olabilir. Aşağıda örnek script içinde yorum takılarının eklenmiş hali görülmektedir.

```
<HTML>

<HEAD>

<TITLE>VBScript ile çalışma</TITLE>

<SCRIPT LANGUAGE="VBScript">

<!--

MsgBox "Internet Web Sayfama Hoşgeldiniz!"

-->

</SCRIPT>

</HEAD>

</HTML>
```

Şimdi, VBScripti desteklemeyen bir tarayıcı bu sayfayı işlediği zaman, scriptinizi bir yorum olarak görecek ve basit şekilde onu görmezden gelecektir.

Scriptlere Yorum Ekleme

VBScript içinde kodlama yaparken daha sonraki güncellemelerde size yaptığınız kodlamayı daha iyi hatırlatacak birtakım önemli şeyleri not düşmek ve bu notların bir kod değilde açıklama olarak kalmasını isterseniz.

O zaman böyle bir şeyi yapmak için, ya Rem ifadesini yada (') tırnak işaretini kullanarak yorumlarınızı program kodunuz içine ekleyebilirsiniz. Eklediğiniz bu açıklamalar (veya yorumlar) tarayıcılar tarafından gözardı edilecektir.

Herhangi bir standard programlama dili gibi, VBScript kod içinde Script yorumlayıcılar tarafından göz ardı edilen text yorumlara izin verir. Yorum oluşturmak için Rem ifadesi kullanılır ve tek tırnak kullanılarakta aynı şey yapılabilir.

```
Rem Bu bir deneme web sayfasıdır.
```

veya

```
' Bu bir deneme web sayfasıdır.
```

Rem ifadesini kullandığınızda, kodun sonu ve Rem yorumu arasında iki nokta üst üste (:) olmalı.

Aşağıda VBScript yorumları örnekleri verilmiştir.

```
'Bu bir yorum

Rem Bu bir yorum

Document write("Merhaba ") :Rem başka bir yorum

Document write("Hepiniz")   ' ve başka biri daha
```

VBScript Kodlama Dönüşümleri

Kodlama Dönüşümü Nedir?

Kodlama dönüşümleri Microsoft Visual Basic Scripting Edition kullanılarak kod yazmanıza yardım eder. Kodlama dönüşümlerin çoğu nesnelerin, değişkenlerin, ve prosedürlerin isimlerini dönüştürmeyi kapsar.

Yorumlama dönüşümleri

Text formatlama ve yol gösterici satırlar ekleme

Uygun bir kodlama dönüşüm kümesini kullanmanın ana amacı bir script kodlama biçimini ve yapısını standard hale getirmektir veya script kümesini siz veya başka birileri kolay bir şekilde okuyabilsin veya anlayabilsin diye.

İyi kodlama dönüşümleri kullanmak ile mümkün olduğunca sezgiye dayalı diğer diller ile dönüşümlerde doğru, okunabilir ve belirli kaynak kod olarak sonuç verir.

Sabit isimlendirme dönüşümleri

Sabit isimleri büyük harfler ile yazılmış olmalı ve sözcükler arasında alt çizgi (_) olmalı.

Örneğin,

KULLANICI_SAYISI

YENI_SAYFA

Değişken isimlendirme dönüşümleri

Tutarlılık ve okunabilirlik amacı için, aşağıdaki tabloda verilen ön ekleri VBScript içindeki değişkenler için tanıtıcı isimler ile kullanınız.

Alt tip	Önek	Örnek
Boolean	bln	blnBulundu
Byte	byt	bytKod
Date (Time)	dtm	dtmBasla
Double	dbl	dblTolerans
Error	err	errSiraNum
Integer	int	intMiktar
Long	lng	lngMesafe
Object	obj	objGuncel
Single	sng	sngOrtalama
String	str	strIklAdi

Değişken kapsamı

Değişkenler daima mümkün olduğunca en küçük kapsamda tanımlanmalı. VBScript değişkenleri aşağıdaki kapsama sahip olabilirler.

Kapsam	Değişkenin tanımlandığı yer	Görünürlük
Prosedür-seviyesi	Olay, fonksiyon, veya Sub prosedür	Tanımlandığı prosedür içinde görünür.
Script-seviyesi	HTML sayfasının HEAD kısmında, herhangi bir prosedür dışında	Script içinde herhangi bir prosedür içinde görünür.

Değişken kapsam önekleri

Script büyüklüğü büyürken, değişkenlerin kapsam farkını hızlı bir şekilde anlamak için kullanılan değer. Önde olan bir bir-harf kapsam öneki yazımı buna bir önek sağlar, burada değişken isimlerinin büyüklüğünü artırmaksızın yapılır.

Kapsam	Önek	Örnek
Prosedür-seviye	None	dblHız
Script-seviye	s	sblnHesapla

Tanıtıcı değişken ve prosedür isimleri

Bir değişken gövdesi veya prosedür adı büyük küçük harf karışık olarak kullanmalı ve mümkün olduğunca amacını tam olarak tanımlamalı. Buna ek olarak prosedür adları bir fiille başlamalı, örneğin, KapatPencere gibi.

Çok sık olarak kullanılan veya uzun terimler, standard kısaltmalar isimleri uygun bir yapıda tutmanıza yardım etmesi için önerilir. Genelde, değişken isimler 32 karakterden daha fazla olabilir ve okunması zor olabilir.

Nesne isimlendirme dönüşümleri

Verilen tabloda VBScript programlama yapılırken değerlendirebileceğiniz değişik nesneler için önerilen dönüşümleri liste halinde mevcuttur.

Nesne tipi	Önek	Örnek
3D Panel	pnl	pnlGrup
Animated button	ani	aniMailKutusu
Check box	chk	chkYanlizcaOkunabilir
Combo box, drop-down list box	cbo	cboIngilizce
Command button	cmd	cmdCik
Common dialog	dlg	dlgDosyaAc
Frame	fra	fraDil
Horizontal scroll bar	hsb	hsbAlan
Image	img	imgIkon

Label	lbl	lblYardimMesaji
Line	lin	linDikey
List Box	lst	lstPolitikaKodu
Spin	spn	spnSayfalar
Text box	txt	txtBilgi
Vertical scroll bar	vsb	vsbOrani
Slider	sld	sldOlcek

Kod yorumlama dönüşümleri

Tüm prosedürler ne yaptıklarını tanımlayan kısa bir yorum ile başlamalı. Bu tanım detaylı bir şekilde ne yaptığını anlatmamalı, çünkü bunlar sık sık zaman içinde değişebilir, böylece gereksiz açıklamalardan kaçınmış oluruz. Kodun kendisi ve gerekli satır içi yorumlar sunumun nasıl yapıldığını gösterir.

Prosedüre geçilen argümanlar onların amacı açık olmadığı zaman ve prosedür belirli bir bölge içinde olan argümanlar beklediği zaman tanımlanmalı. Fonksiyon değerler döndürür ve prosedür tarafından değiştirilen diğer değişkenler de, özellikle referans argümanlar, her bir prosedürün başlangıcında tanımlanmalıdır.

Prosedür başlık yorumları aşağıda verilen başlık yazma kısımlarını kapsamalı. Örneğin, "kodunuzu formatlama " kısmına bakın.

Kısım Başlığı	Yorum İçeriği
Amaç	Prosedürün ne iş yaptığı.
Etkilenmeler	Bu prosedürü etkileyen herhangi bir değişken, kontrol, veya diğer elemanın listesi
Etkiler	Herbir dış değişken, kontrol, veya eleman üzerindeki prosedürün etki listesi.
Girişler	Her bir argümanın nicin olduğu açıklanır. Herbir argüman ayrı bir satırda açıklanır.
Dönüş değerleri	Döndürülen değerin açıklaması.

Aşağıdaki noktaları unutmayın:

Herbir önemli değişken bildirimi bildirilen değişkenin kullanımını tanımlayan bir satır içi yoruma sahip olmalı.

Değişkenler, kontroller, ve prosedürler karmaşık sunum detayları için yanlızca gerekli olan satır içi yorumları yeterince açık olarak isimlendirmeli.

Scirpt başlangıcında, scripti, nesneleri belirtme, prosedürler, algoritmalar, diyalog kutuları, ve diğer sistem bağımlılıklarını tanımlayan bir ön görüş dahil etmelisiniz.

Kodunuzu formatlama

Mantıksal yapıyı ve iç içe olmayı yansıtmak için kod biçimlendirmesine izin verirken mümkün olduğunca ekran alanı korunmalı.

```
<SCRIPT LANGUAGE="VBScript">

<! - -

'*****************************************************************************
' Amaç      : KullaniciListesi dizisinde belirli bir kullanıcının ilk
'             göründüğü yer.
' Girişler  : strKullaniciListesi():  araştırılan kullanıcıların listesi.
'             strHedefKullanici:  aranacak kullanıcı listesi.
' Dönüşler: strKullaniciListesi içindeki strHedefKullanici nın ilk olduğu yerin
'             indis numarası
'             Eğer hedef kullanıcı bulunamaz ise –1 döner.
'*****************************************************************************

Function IlkKullaniciBul( strKullaniciListesi(), strHedefKullanici )

   Dim i

   ' Çevrim sayıcı.

   Dim blnBulundu

   ' Hedef bulundu bayrağı.

   IlkKullaniciBul = -1

   i = 0

   Do While i <= Ubound(strKullaniciListesi) and Not blnBulundu

      If strKullaniciListesi(i) = strHedefKullanici Then

         blnBulundu = True

         IlkKullaniciBul = i

      End If
```

```
    Loop
End Function
-->
</SCRIPT>
```

Sayfaları formatlama

Web sitelerindeki örnekler aşağıdaki şablonu kullanırlar:

```
<!DOCTYPE HTML PUBLIC "-//IETF/DTD HTML//EN">
<!-- Microsoft Visual Basic Scripting için önerilen şablon -->
<HTML>
<HEAD>
<TITLE> </TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Function OdemeTarihi (odemeT)
    OdemeTarihi = (Cdate(odemeT) - Now()) > 2
End Function
-->
</SCRIPT>
</HEAD>
<BODY>
<FONT FACE=ARIAL SIZE=2>
<!-- global ilk değerler -->
<!- -  statik sayfa içeriği buraya yazılır. -->
</FONT>
```

</BODY>

</HTML>

Şablon yapı aşağıdaki özelliklere sahiptir:

VBScript prosedürleri, kontrol için olay prosedürleride kapsar, sayfanın <HEAD> kısmında olur. Bu bir yerdeki tüm kodu bir yerde tutar.

<SCRIPT> takısının kotaramadığı tarayıcılardan kodu gizlemek için HTML (<!- - ve - - >) takıları ile başlar ve sona erer.

<HTML>, <HEAD>, <TITLE>, ve <BODY> takıları standard bir HTML sayfası oluşturmak için kullanılır.

Buna ek olarak, bir site üzerindeki örnekler <FORM> takılarını tüm tarayıcılar içinde kontrol etmek için <INPUT> elemanlarından oluşmuş grubu çevreler.

BÖLÜM 3

Üç

VBScript Değişkenleri

- Değişken Tanımlama
- Değişkenleri İsimlendirme Kuralları
- Değişkenlerin Kapsamı ve Yaşamsüresi
- Değişkenlere Değerler Atama
- Değişkenler ve Dizi Değişkenler

VBScript Değişkenleri

Bir değişken sizin scriptinizin çalışması esnasında değişebilen program bilgilerinin saklandığı bilgisayar bellek lokasyonuna işaret eden uygun bir yer tutucudur. Örneğin, siz nesnesayisi adlı bir değişkeni belirli bir web sayfasında bir nesneye bir kullanıcının kaç kez tıkladığının sayısını saklamak için oluşturabilirsiniz. Değişkenin bilgisayar belleğinde nerede saklandığının bir önemi yoktur yani sizin bunu bilmenize gerek yoktur. Önemli olan şey değişken ismine göre değeri görme ve değiştirmedir. VBScript içinde, değişkenler daima Variant temel bir veri tipidir.

Variant olarak tanımlanan değişken üç tür değer tutabilir: skalar yani tek boyutlu değerler, diziler, ve nesne işaretçileri. VBScript her değişken için kullanılacak bilgisayar bellek lokasyonuna bir sembolik isim atar , atanan bu isim program içinde o bellek alanını temsil edecek şekilde kullanılır. Tanımlanan değişken adı bir VBScript anahtar sözcüğü olamaz. Eğer yanlışlıkla bir değişkene bir anahtar sözcük adını verirseniz, uygulamanızda çok ciddi problemlere neden olabilir. Yanlış yazılmış değişken isimlerini VBScript hata vermeden yeni bir değişken adı gibi yorumlar. Örneğin, eğer siz E-posta ile aynı anlama gelecek olan eposta yazarsanız, VBScript birbirinden tamamen farklı olan iki değişken oluşturacaktır. Bu tip durumlardan kurtulmak için Option Explicit bildirimini programımızın en baş kısmında yapmak zorundayız.

Değişken Tanımlama

Siz scriptiniz içinde Dim ifadesi, Public ifadesi ve Private ifadesi kullanarak değişken tanımlamaları yapabilirsiniz.

Dim: VBScript içinde değişken tanımı yapmak için kullanılan anahtar sözcük olup bir program içinde aynı isime sahip iki değişken olamaz.

Dim ile yapılan değişkenler bilgisayarın belleğinde yer alır, dolaysıyla bunlar geçici tanımlamalardır.

Dim komutu ile hem dinamik hemde statik diziler tanımlayabilirsiniz. Bir statik dizi Dim yapısı ile belirtilen sayı kadar elemana sahiptir. Bunun yanında siz elemanları 0 dan başlamak üzere numaralandırmak zorundasınız. Dim bildirimini göz önünde bulundurursak, 0,1,2,3,4,5,6,7,8 ve 9 gibi 10 elemanlı bir dizi oluşturur.

```
<%

Dim statikdizi( 9 )

%>
```

Bir dinamik dizi tanımı iki parantez arası boş bırakılarak yapılır. Bu tanımdan sonra, siz programınızın en sonunda ReDim yapısını kullanarak yeni bir boyut ve eleman sayısını bildirebilirsiniz. Gerçekte, istediğiniz kadar bir dinamik diziyi bildirebilirsiniz.

```
<%
Dim dinamikdizi()
...
ReDim dinamikdizi(34)
...
ReDim dinamikdizi(334)
%>
```

Diziler 60 tane boyuta kadar VBScript tarafından desteklenir. Örneğin, aşağıdaki kod dört boyutlu bir dizi oluşturur. İlk boyut 10 elemana sahip, ikincisi 20 elemana sahip, üçüncüsü 30 elemana sahip, ve dördüncüsü ise 100 elemana sahiptir. Toplam eleman sayısı ise her boyuttaki elemanları çarpımı kadardır.

Sizin oluşturabileceğiniz dizi boyut sayısı mevcut bellek ile sınırlanır. Eğer mevcut belleğiniz aşan bir dizi tanımı yaparsanız, o zaman bir hata mesajı alacaksınız.

```
<%
Dim dortboyutdizi(10, 20, 30, 100)
%>
```

Toplam eleman sayısı: 11x21x31x101 = 723261 olacaktır. Burada indisin sıfırdan başladığını unutmayın.

Örneğin:

```
<%
Dim Sonuc
Sonuc = 200
Response.Write Sonuc
%>
```

Siz aralarına virgül koymak şartı ile tek bir Dim yapısı ile aynı satırda birden fazla değişken tanımı yapabilirsiniz.

Örneğin:

```
Dim Ust, Alt, Sol, Sag

Dim Renk

Dim X, Y, Z, W

X=100
```

Siz scriptiniz içinde basit bir şekilde değişken adını kullanarak tam olarak bir değişken tanımıda yapabilirsiniz. Bu genellikle iyi bir uygulama şekli değildir çünkü scriptiniz çalıştığı zaman beklenmeyen sonuçlara neden olmasından dolayı siz bir değişken adını bir veya daha çok yerde telaffuz edemeyebilirdiniz. Bu nedenden dolayı , tüm değişkenlerin açık olarak tanımlanmasını zorunlu yapmak için Option Explicit ifadesi mevcuttur. Option Explicit ifadesi scriptiniz içinde yazılan ilk ifade olmalıdır.

Option Explicit: Program içinde kullanılacak tüm değişkenlerin tanımlanmasını zorunlu hale getirmek için kullanılır. Option Explicit etkin olduğu zaman, VBScript bilinmeyen bir sembol gördüğünde bir derleme hatası verir. Varsayılan olarak Option Explicit etkin olmadığı zaman, VBScript tanınmayan bir sembol gördüğünde o sembol adı olan yeni bir değişken oluşturur.

Değişkenleri İsimlendirme Kuralları

Değişken adları VBScript içinde herhangi bir isimlendirme için standard kurallara uyar.

Bir değişken adı

- Alfabetik bir karekterle başlamak zorunda.

"A" dan "Z" ye, "a" dan "z" ye, ve "_" ile başlar

- Değişken içinde nokta olmamalı.

Abc.xy34 şeklinde bir değişken tanımı olamaz, fakat Abc_xy34 şeklinde bir değişken tanımı yapabilirsiniz.

- Uzunluğu 255 karakteri geçmemeli.

Maksimum değişken adı uzunluğu 255 karakterden büyük olamaz, en az bir karakter olabilir o da bir harf olmak zorunda; örneğin, A, B, gibi, 3, 5 gibi bir tek karakterden oluşan bir değişken olmaz.

- Tanımlandığı kapsamda o isimde başka bir değişken olmamalı.

Global bağlamda bir program içinde aynı ada sahip iki veya daha fazla değişken tanımı yapılamaz. Değişken adları tek olmak zorunda.

Değişkenlerin Kapsamı ve Yaşamsüresi

Bir değişkenin kapsamı (geçerli olduğu alan) onu tanımladığını yer tarafından belirlenir. Bir prosedür içinde bir değişken tanımı yaptığınızda, yanlızca prosedür içindeki kod o değişkene erişebilir ve değerini değiştirebilir. Bu şekilde tanımlanan değişken lokal bir kapsama sahiptir ve prosedür seviyesi değişken olarak adlandırılır. Eğer siz bir değişkeni prosedür dışında tanımlarsanız, siz o değişkeni scriptiniz içinde tüm prosedürlere tanıtmış olursunuz. Bu bir script seviyesi değişkendir , ve o script seviyesi bir kapsama sahiptir.

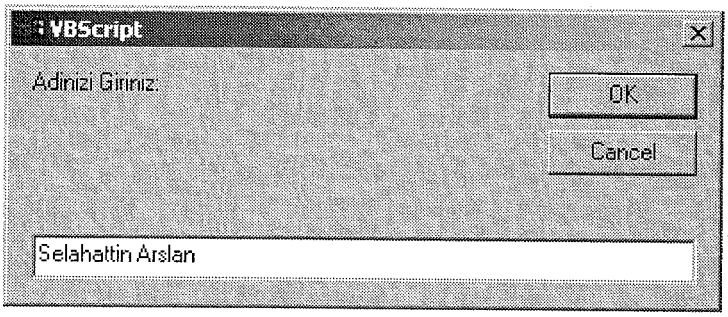
Bir değişkenin yaşam süresi ne kadar var olacağı süreye bağlıdır. Bir script seviyesi değişkenin yaşam süresi tanımlandığı yerdeki zamandan scriptin sonlanacağı zamana kadar devam edecektir. Prosedür seviyesinde ise, bir değişken siz o prosedür içinde kaldığınız sürece olacaktır. Lokal değişkenler prosedür yürütülürken geçici olarak saklama yerleri olarak kullanmak iyi bir yöntemdir. Siz farklı prosedürlerde aynı isime sahip çok sayıda değişkenlere sahip olabilirsiniz çünkü bu değişkenler yanlızca tanımlandığı prosedür tarafından tanınır ve kullanılırlar.

Değişkenlere Değerler Atama

Değişkenlere izleyen ifadedeki gibi bir şekilde değer ataması yapılır: değişken ifadenin sol tarafındadır ve değişkene atamak istediğiniz değer sağ taraftadır. VBScript içinde atama yapmak için (=) operatörü kullanılır.

Örneğin:

```
Dim Adi

Adi = InputBox("Adınızı Giriniz: ")
```



MsgBox "İsminiz " & Adi & " Olarak Girildi"

Yukarıdaki şekildede göreceğiniz gibi Adi değişkeni programımız içinde girilen adı içermektedir. Bu değeri program sonlanana kadar istediğiniz yerde kullanabilirsiniz.

```
' Dizi değişken tanımı

Dim B

Dim Dizi

Dizi=Array(5)     ' Bu dizi indisi 0  dan başlar 5 e kadardır.
                  ' 6 elemanlı bir dizi.

B = 200


' Dizilere ulaşmak için indisleri kullanırız.

Dizi(0) ="Dizinin 1. elemanı"

Dizi(3) ="Dizinin 4. elemanı"

Response.Write "B değişkeninin değeri=" & B & "<BR>"

Response.Write "Dizinin 1. elemanı =" & Dizi(0) & "<BR>"
```

Değişkenler ve Dizi Değişkenler

Çok zaman, siz tanımladığınız değişkene tek bir değer atamak istersiniz. Tek bir değer içeren değişkenler olan sayılar ve stringler birer skalar değişkendir. Diğer zamanlar, tek bir değişkene bağlı bir değer atamadan daha çok uygundur. O zaman siz bir dizi değeri içeren bir değişken tanımı yapabilirsiniz. Bu bir dizi değişkeni olarak adlandırılır. Dizi değişkenleri ve skalar değişkenler aynı şekilde tanımları yapılır (bir dizi tanımında değişken ismini takip eden ( ) parantezler hariç). Aşağıdaki örnekte, 11 eleman içeren tek boyutlu bir dizi tanımlanmıştır:

```
Dim A(10)
```

Bir skalar listesini veya nesne işaretçileri değerlerini tutan variantlar dizilerdir. Bir dizi değişken gerçekte değerler listesini tutmaz, o bellekteki topluluğun ilk pozisyonuna bir işaretçiyi tutar. Siz bir dizi oluşturduğunuzda, bilgisayar tüm diziyi tutabilecek kadar bellekten yer ayırır. VBScript dinamik dizileri destekler, dinamik diziler oluşturulduktan sonra tekrar boyutlandırılabilir. Böyle olmasına rağmen böyle bir şey yaptığınız zaman, bilgisayar yeni diziyi tutacak kadar büyüklükte yeni bellek alanı ayırır ve sonra orijinal dizi değerlerini yeni yerlerine kopye eder. Bu yüzden, maksimum sabit boyutlu bir dizi oluşturmamız , dinamik bir diziyi yeniden boyutlandırmaya göre çok hızlıdır.

Parantezler içinde 13 sayısı görülmesine rağmen, VBScript içinde tüm diziler sıfır tabanlıdır yani ilk indis sıfırdır, bundan dolayı bu dizi gerçekte 14 eleman içerir. Sıfır tabanlı dizilerde, bir dizinin eleman sayısı daima parantez içinde gösterilen sayının bir fazlası olacaktır. Bu tür diziler sabit uzunluklu bir dizi olarak adlandırılır.

Siz dizi indislerini kullanarak dizinin her bir elemanına bir değer atayabilirsiniz.

Başlangıç indis değeri 0 ve son indis değeri 13 dür, veri dizi elemanlarına aşağıdaki şekilde atamaları yapılır:

```
Daire(0) = 256

Daire(1) = 324

Daire(2) = 100
...

Daire(13) = 55
```

Benzer şekilde, siz istediğiniz belirli bir dizi elemanını indis kullanarak herhangi bir elemanından veri alabilirsiniz.

Örneğin:

```
. . .
degisken = Daire(8)
. . .
```

Diziler tek bir boyutla sınırlı değildir. Siz 60 boyutuna kadar sahip diziler tanımlayabilirsiniz, ama çoğu kişi bu boyutları en fazla 3 veya 4 e kadar kullanır. Siz parantez içinde bir dizinin büyüklüğünü gösteren sayırları virgül ile ayırarak çok boyutlu diziler tanımlayabilirsiniz. Aşağıdaki örnekte, Tablo değişkeni 6 satır ve 11 sütun içeren iki boyutlu bir dizidir:

```
Dim Tablo(5, 10)
```

İki boyutlu bir dizide, ilk sayı daima satır sayısını gösterir; ikinci sayı ise sütun sayısını gösterir.

Siz scriptiniz çalışırken büyüklüğü değişen bir dizi tanımı yapabilirsiniz. Bu bir dinamik dizi olarak adlandırılır. Dizi bir prosedür içinde ilk olarak tanımlanırken ya Dim yapısı yada ReDim yapısı kullanımıyla yapılır. Bununla birlikte, dinamik bir dizi için, parantezler içinde büyüklük veya boyut tanımı yapılmaz.

Örneğin:

```
Dim dizim()

ReDim digerDizi ()
```

ReDim: Bu yapı kurallarına uygun, ve daha sonra ihtiyacımız olacak kadar tekrardan boyutlandırmak üzere tanım yapmamıza izin verir. Burada dinamik dizi orijinal olarak ya Dim, Private, veya Public kullanılarak bildirilmiş olabilir. Bir dizi tanımını yapmak için ilk olarak ReDim kullandığınızda, siz ya tek yada çok boyutlu bir dizi tanımlayabilirsiniz. Bunun yanında, boyutların sayısını belirttikten sonra, artık bu boyut sayısından geriye dönüş olmaz. Çok boyutlu bir dizi tanımı yaptıktan sonra, siz yanlızca en son elemanın büyüklüğünde yeni bir bildirim yapabilirsiniz.

Dinamik bir diziyi kullanmak için, boyutların sayısını ve her bir boyutun büyüklüğünü belirlemek için alt sıra bir tanımlama olarak ReDim yapısını kullanmak zorundasınız. Aşağıdaki örnekte, ReDim dinamik bir dizinin büyüklüğünü başlangıç değerini 15 olarak ayarlar. Alt bir tanım sırasında ReDim yapısı dizi büyüklüğünü yeniden 20 olarak belirler, fakat diziyi yeniden boyutlandırmada dizi içindeki mevcut değerleri korumak için **Preserve** anahtar sözcüğünü kullanmak gerekiyor.

```
ReDim dizim(15)
. . .
```

```
ReDim Preserve dizim(20)
```

Örnekte, dizi() tanımımız üç boyuta sahip ve en son elemanı yeniden boyutlandırmak istiyoruz. En son elemanı 35 den 5 şeklinde değiştirdiğimiz zaman, geriye kalan 30 elemanı kaybederiz. Aşağıdaki satırlar hemen arka arkaya olmak zorunda değil program içinde herhangi bir yerde olabilir.

```
<% Dim dizi() %>
. . .

<% ReDim dizi(10, 20, 35) %>
. . .

<% ReDim Preserve dizi(10, 20, 355) %>
. . .

<% ReDim Preserve dizi(10, 20, 5) %>
. . .
```

Dinamik bir boyutlandırmada kaç kez yaptığının bir önemi yok çünkü bu sınırsızdır, böyle olmasına rağmen, eğer siz bir diziyi daha küçük yapmak isterseniz, atılan elemanlardaki verileri kaybedebilirsiniz.

Siz aşağıdaki şekilde bir dizi oluşturabilirsiniz.

```
Dim renkler(1 to 16 )
```

Burada 16 elemanlı bir dizi var ve ilk indisi 1 den (0 değil) başlamakta olup son indiside 16 dır.

Geçerli dizi indis değerleri bölgesi dizinin sınırlarıdır. Siz alt sınır ve üst sınır değerlerini LBound(DiziDegiskeni) ve UBound(DiziDegiskeni) fonksiyonlarını kullanarak bulabilirsiniz.

Örneğin:

```
Dim Renk( 1 to 16 )

Response.Write "Alt sınır değeri=" & LBound(Renk) &"<BR>"

Response.Write "Üst sınır değeri=" & UBound(Renk) &"<BR>"
```

Eğer siz daima sıfır-tabanlı yani dizinin en düşük indis değeri sıfırdan başlayan diziler kullandığınızda, büyük bir ihtimalle sorunlar yaşayacaksınız. Sıfır-tabanlı bir dizi kullanıyorsanız o zaman sizin dizinin toplam eleman sayısı:

ToplamElemanSayisi=(Ubound(Dizi)– Lbound(Dizi)+1)

Formülü ile hesaplanabilir.

Sizin Variant bir değişkende tutabileceğiniz üçüncü bir değer ise nesne işaretçileridir. Server.CreateObject fonksiyonu ile bir nesne oluşturduğunuzda, VBScript nesnenin verisini tutmak için ayrı bir yerde bir bellek alanı ayırır. Server.CreateObject fonksiyonu bellekteki o pozisyona ilişkin bir işaretçi döndürür. Sizin nesne değişkeniniz o işaretçiyi tutar ve Variant değişkeninizdeki işaretçi bir dizinin ilk elemanının bellekteki adresine işaret eder ve diğer elemanlara indis değerleri değiştirilerek ulaşılır. VBScripte bir nesne işaretçisi ve değer arasındaki farkı belirtelim diye bir nesne değişkeni oluştururken Set anahtar sözcüğünü kullanmak zorundayız.

Örneğin:

```
Dim nesneDic

Set nesneDic=Server.CreateObject("Scripting.Dictionary")
```

Siz bir değişkenin bir nesne tutup tutmadığını isObjetc metodunu kullanarak belirleyebilirsiniz. Bir nesne değişkeni oluşturduktan sonra, o nesneye ait metodları ve özellikleri nokta yazım şeklini kullanarak çağırabilirsiniz.

Örneğin:

```
nesneDic.Add  Anahtar.Deger
```

ASP bir sayfa sonlandığında tüm lokal tanımlı değişkenleri yok eder yani değişkenler tarafından tutulan bellek alanları serbest bırakılır. Siz yinede eğer bir nesne değişkeniniz var ise, bu nesne değişkenine Nothing değerini atayarak kendiniz için iyi bir deneyim yapmış olacaksınız.

```
Set nesneDic=Nothing
```

Değişkene Nothing değerini atayarak nesne tarafından tutulan bellek alanını başka bir işlemde kullanılması için serbest bırakmış oluruz.

BÖLÜM
DÖRT
4

VBScript Veri Tipleri

- Variant Alt Tipleri
- Empty
- Null
- Nothing
- Boolean
- Byte
- Integer
- Currency
- Long
- Single
- Double
- Date
- String
- Object
- Error

VBScript Veri Tipleri

Bir script yazma dili olan VBScript Variant olarak adlandırılan tek bir veri tipine sahiptir. Bir Variant, veri tipinin nasıl kullanılacağına bağlı olarak farklı türde bilgi içeren özel bir veri tipidir. Variant yanlızca VBScript içinde veri tipi olduğu için, o aynı zamanda VBScript içinde tüm fonksiyonlar tarafından döndürülen veri tipidir.

En basit şeklinde, bir Variant ya bir sayısal yada bir string bilgi içerir. Bir Variant sayısal bir içerik taşıdığında sayısal bir değişkenmiş gibi ve string bilgi içerdiğinde ise bir string değişkenmiş gibi davranır. O, eğer siz sayılar gibi verilerle çalışıyorsanız, VBScript ona sayılar gibi yaklaşır ve uygun olan sayı tipini seçer. Benzer olarak , eğer siz yanlızca stringler ile çalışıyorsanız, VBScript onlara string veriler gibi davranır. Siz daima sayıları (" ") gibi tırnak işaretleri arasına alarak bir string gibi davranmasını sağlayabilirsiniz.

Variant Alt Tipleri

Basit sayısal ve string sınıflandırmasının ötesinde, bir Variant sayısal bilginin bilinen yapısı hakkında daha çok üstünlüklere sahiptir. Örneğin, bir tarih veya zamanı gösteren sayısal bilgiye sahip olabilirsiniz. Diğer tarih ve zaman verisi ile kullanıldığında, sonuç daima bir tarih veya zaman olarak ifade edilir. Siz Boolean değerlerden kayan noktalı sayılara kadar geniş bir alana sahip olabilirsiniz. Variant içinde taşınan bu farklı kategoriler alt tipler olarak adlandırılır. Çok zaman, siz bir Variant içinde bilgilerinizi koymak isteyebilirsiniz, ve Variant onun içindeki veriye en uygun şekilde davranır.

Aşağıda Variant 'ın taşıyabileceği alt veri tipleri verilmiştir.

Empty

Variant için ilk değer atanmamıştır. Sayısal değişkenler için değer ilk değer 0 veya string değişkenler için ilk değer sıfır uzunluklu bir string ("").

Empty anahtar sözcüğü Dim yapısı kullanılarak bildirildiği zaman bir değişkene otomatik olarak atanan değerdir. Tanımlanan değişkenin Empty gibi bir değere sahip olmasının ne anlama geldiğini anlamak için önemlidir, sanki kodunuz içinde atama yapmıyormuşsunuz gibi.

IsEmpty() fonksiyonunu kullanarak bir değişkenin empty olup olmadığını test edebilirsiniz. Empty , Null veya Nothing ile aynı şey değildir.

```
<% =IsEmpty(degisken) %>
```

Sonuç True olacaktır.

```
<% =IsEmpty("Bu bir string.") %>
```

Sonuç False olacaktır.

Null

Variant geçerli bir veri içermemektedir. Null anahtar sözcüğü bir değişkenin geçersiz bir değer içerdiğini gösterir. Siz bir değişkene Null değerini atayabilirsiniz ve IsNull() fonksiyonunu kullanarak bir değişkenin Null olup olmadığını anlamak için kullanabilirsiniz. Eğer değişken değerini yazdırırsak, çıkış Null olmalı. Null, Empty veya Nothing ile aynı şey değildir.

```
<% degisken = null %>
```

```
<% =IsNull(degisken) %>
```

Sonuç True olacaktır.

```
<% =IsNull("Bu bir string.") %>
```

Sonuç False olacaktır.

Nothing

Nothing anahtar sözcüğü bir nesneden bir nesne değişkenini ayırmak için kullanılır. Eğer birkaç nesne değişkeni aynı nesneye işaret ediyorsa, hepsi sistem kaynakları bırakılmadan önce Nothing olarak ayarlanmalıdır. Bu şekilde mevcut kaynakları korumuş oluruz.

Nothing değerini atamak için Set anahtar sözcüğünü kullanmalıyız.

```
<% Set nesne = Nothing %>
```

Boolean

Ya True yada False içerir. False anahtar sözcüğü 0 değerine sahiptir. False anahtar sözcüğünü koşullu ifadelerde test etmek için temel bir değer olarak kullanabilirsiniz. True anahtar sözcüğü –1 değerine sahiptir. True anahtar sözcüğünü koşullu ifadeleri test etmek için temel bir değer olarak kullanabilirsiniz

Byte

Byte veri tipi 0 ile 255 arasında bir tamsayı değer içerir.

```
<%
Dim bytSayi
bytSayi=129
Response.Write bytSayi
' Çıktısı 129 şeklinde olacaktır.
%>
```

Integer

-32,768 ile 32,767 arasında bir tamsayı içerir.

```
<%
Dim intSayi
intSayi=12945
Response.Write intSayi
' Çıktısı 12945 şeklinde olacaktır.
%>
```

Currency

Currency değer aralığı -922,337,203,685,477.5808 ile 922,337,203,685,477.5807 arasında bir değer içerir.

```
<%
Dim curMiktari
curMiktari =1294576869.567
Response.Write curMiktari
' Çıktısı 1294576869.567 şeklinde olacaktır.
%>
```

Long

Long değer aralığı -2,147,483,648 ile 2,147,483,647 arasında bir tamsayı içerir.

```
<%

Dim lngBorc

lngBorc =1294576869

Response.Write lngBorc

' Çıktısı 1294576869 şeklinde olacaktır.

%>
```

Single

Tek duyarlı , negatif değerler için kayan noktalı sayı -3.402823E38 ile -1.401298E-45 arasındadır; pozitif değerler 1.401298E-45 ile 3.402823E38 arasındadır.

```
<%

Dim snglMiktar

snglMiktar =-2.402823E38

Response.Write snglMiktar

' Çıktısı -2,402823E+38 şeklinde olacaktır.

%>
```

Double

Çift-duyarlıklı, kayan-noktalı sayı negatif değerler için -1.79769313486232E308 ile -4.94065645841247E-324 arasındadır; pozitif değerler için 4.94065645841247E-324 ile 1.79769313486232E308 arasındadır.

```
<%

Dim dblMiktar

dblMiktar =-1.79769313486232E305

Response.Write dblMiktar
```

' Çıktısı -1,79769313486232E+305 şeklinde olacaktır.

%>

Date (Time)

Tarih bilgisini January 1, 100 ve December 31, 9999 arasında içeren bir sayı. Date fonksiyonu bilgisayar sisteminiz tarafından belirlenen o anki tarih bilgisini döndürür.

Normalde iki hanelik yıl bilgisi (AA/GG/YY) döndürür, fakat yeni bir yüzyıl başlangıcı olması nedeniyle, artık dört hanelik yıl bilgisi (AA/GG/YYYY) döndürülmektedir.

Burada, A = ay, G = gün, ve Y = yıl dır.

<% =Date %>

Çıktısı 3/31/2002 şeklinde olacaktır.

Tarih çıktı formatı bilgisayarınızdaki bölgesel ayarlara göre değişecektir.

String

Uzunluk olarak 2 milyara yakın karakter alabilen bir değişken uzunluklu string.

<%

Dim strAdi

' *strAdi* değişkeni bir string değer içerdiği için artık bir string değişken olarak
' davranacaktır.

strAdi = "Selahattin Arslan"

Response.Write strAdi

' Çıktısı "Selahattin Arslan" olacaktır.

%>

Object

Bir nesne içerir. Bir nesne oluşturmak için New anahtar sözcüğünü Set ile birlikte kullanılarak bir Class veya bir RegExp için bir tutamaç oluşturulur. İlk önce, Dim kullanarak, atama yapılacak olan tutamaç değişkenini bildiririz. Sonra, bir tutamaç oluşturabilirsiniz.

```
<%
Dim aramaDeseni
' aramaDeseni değişkeni bir düzenli ifade nesne değişkeni olur.
Set aramaDeseni = New  RegExp
%>
```

Error

Bir hata numarası içerir.

Siz veriyi bir alt tipten diğerine dönüştürmek için dönüşüm fonksiyonları kullanabilirsiniz. Ek olarak, VarType fonksiyonu sizin verinizin Variant içinde nasıl saklandığı hakkında bilgi döndürür.

Err nesnesi en son oluşan çalışma zamanı hatası hakkında bilgi içerir. Default özellik olarak Number özelliği kabul edilir. Bu özellik hata numarasını içerir.

```
<%
Dim hatano, hatabilgi
On Error Resume Next
Err.Raise 6
hatano = Err.number
hatabilgi = Err.description
If hatano <> 0 Then
  Response.Write "Bir hata oldu, Hata no: " & hatano & " Bilgi '" & hatabilgi & "'."
End If
%>
```

BÖLÜM
BEŞ
5

VBScript Sabitleri

- VBScript İçinde Sabit Tanımlama
- Renk Sabitleri
- Tarih ve Zaman Sabitleri
- Tarih Format Sabitleri
- Değişik Sabitler
- MsgBox Sabitleri
- String Sabitleri
- TriState Sabitleri
- VarType Sabitleri
- Karşılaştırma Sabitleri
- Dosya Özellikleri Sabitleri
- Dosya Giriş/Çıkış Sabitleri

VBScript Sabitleri

Bir sabit asla değişmeyen bir sayı veya stringin yerini alan anlamlı isimdir. VBScript bir çok önemli sabit tanımlar. Siz VBScript dil referansından bu önemli sabitler hakkında bilgi alabilirsiniz.

Kodunuz içinde kullanabileceğiniz faydalı olabilecek sabitlerin çoğu VBScript içine eklenmiştir. Sabitler o an değerin ne olduğunu hatırlamanıza gerek kalmaksızın uygun değerleri kullanmak için bize uygun bir yol sağlar. Sabitleri kullanarak program kodlarımızı daha okunabilir ve yapılmak istenen değişiklikler için daha kolay bir şekil alır. VBScript içinde tanımlanan bu iç sabitlerin değeri değişmez. Bu yüzden siz ayrıca bu sabitleri kodunuz içinde tanımlamanıza gerek yoktur. Kısacası bu sabitlerin gösterdiği değerleri programınızın heryerinde kullanabilirsiniz.

VBScript İçinde Sabit Tanımlama

Siz Const yapısını VBScript içinde kullanarak kullanıcı tanımlı sabitler oluşturabilirsiniz. Const yapısının kullanımı ile, siz anlamlı isimle ve onlara atanan ivedi değerler ile string veya sayısal sabitler oluşturabilirsiniz.

Bir sabit tanımı yaptıktan sonra, artık bu sabitin değerini programın hiçbir yerinde değiştiremezsiniz. Eğer bu değeri değiştirmeye çalışırsanız, o zaman uygunsuz atama diye bir hata mesajı alacaksınız.

Örneğin:

```
Const isim = "Bu benim adım"

Const miktar = 2001

Const tarih =  #10-05-99#
```

String ivedi değerler iki adet çift tırnak (" ") arasına alınır. Tırnak işaretleri çok açık bir şekilde string değerleri sayısal değerleri ayırır. Siz tarih ivedi değerlerini ve zaman ivedi değerleri sayısal işaretlerde (#) arasına alınır.

Örneğin:

```
Const dogumtarihi = #16-10-99#

Const PORT = 53

Const MARKA="Mercedes"
```

Siz değişkenlerden farklı sabitlere bir isimlendirme şemasını uydurmak isteyebilirsiniz. Bu sizin script çalışırken yeni bir değer atamaya çalışmanızı engeller. Örneğin, siz bir "vb" veya "con" sizin sabit isimlerinizde kullanmak isteyebilirsiniz, veya siz tümü büyük harflerden oluşan sabit isimleri verebilirsiniz. Değişkenlerden sabitleri ayırma sizin daha karmaşık scriptler yazdığınızda karışıklığı eler.

```
<% Const sabit = 14.123 %>

Günün değişim oranı <% =sabit %> %.
```

Çıktısı:

Günün değişim oranı 14.123 %.

Aynı anda birden fazla sabit tanımı yapabilirsiniz.

```
<% Const selam = "Merhaba!", sayi = 345, kitap = "VBScript" %>
```

Aşağıda VBScript tarafından sağlanan sabitler kategoriler halinde verilmiştir.

- Renk Sabitleri
- Tarih ve Zaman Sabitleri
- Tarih Format Sabitleri
- Değişik Sabitleri
- MsgBox Sabitleri
- String Sabitleri
- Tristate Sabitleri
- VarType Sabitleri
- Karşılaştırma Sabitleri
- Dosya Özellikleri Sabitleri
- Dosya Giriş/Çıkış Sabitleri

Renk Sabitleri

Script yazarken kullanabileceğimiz sekiz temel renkleri tanımlar.

Bu sabitler VBScript içinde tanımlı olduğu için, siz onları kullanmadan önce tanımlamak zorunda değilsiniz. Bu sabitlerin gösterdiği değerleri programınızın heryerinde kullanabilirsiniz.

Sabit	Değer	Tanımı
vbBlack	&h00	Siyah
vbRed	&hFF	Kırmızı

vbGreen	&hFF00	Yeşil
vbYellow	&hFFFF	Sarı
vbBlue	&hFF0000	Mavi
vbMagenta	&hFF00FF	Magenta
vbCyan	&hFFFF00	Çiyan
vbWhite	&hFFFFFF	Beyaz

Örnekte renk için sabit isimleri doğrudan yazılmıştır.

```
Sub mesajgoster( deger)

If deger = 0 Then

mesaj.ForeColor = vbWhite

mesaj.Font.Bold = True

mesaj.Font.Italic = True

End If

End Sub
```

Tarih ve Zaman Sabitleri

Değişik tarih ve zaman fonksiyonları tarafından kullanılan tarih ve zaman sabitlerini tanımlar.

Bu sabitler VBScript içinde tanımlı olduğu için, siz onları kullanmadan önce tanımlamak zorunda değilsiniz. Bu sabitlerin gösterdiği değerleri programınızın heryerinde kullanabilirsiniz.

Sabit	Değer	Tanımı
vbSunday	1	Pazar
vbMonday	2	Pazartesi
vbTuesday	3	Salı
vbWednesday	4	Çarşamba
vbThursday	5	Perşembe
vbFriday	6	Cuma
vbSaturday	7	Cumartesi
vbUseSystemDayOfWeek	0	Haftanın ilk günü için sistem ayarlarınızda belirtilen haftanın gününü belirtir.
vbFirstJan1	1	Ocak 1 deki hafta için kullanın.
vbFirstFourDays	2	Yeni yıldaki en az dört güne sahip ilk haftası için kullanın.
vbFirstFullWeek	3	Yılın tam ilk haftası için kullanın.

Örnekte, Tuesday (Salı) haftanın ilk günü olarak tanımlıdır. Başlangıç olarak Salı günü alındığına göre çıkış kaç hafta olur. 1/1/2001 ve şu anki tarih arasındaki fark alınmaktadır.

```
<% =DateDiff("w", Date, "1/1/2001", 3) %>
```

Çıktısı:

93

veya

```
<% =DateDiff("w", Date, "1/1/2001", vbTuesday) %>
```

Çıktısı:

93

Tarih Format Sabitleri

Tarihleri ve zamanları formatlamak için kullanılan sabitler.

Bu sabitler VBScript içinde tanımlı olduğu için, siz onları kullanmadan önce tanımlamak zorunda değilsiniz. Bu sabitlerin gösterdiği değerleri programınızın heryerinde kullanabilirsiniz.

Sabit	Değer	Tanımı
vbGeneralDate	0	Bir tarih veya zamanı göster. Sayılar için, bir tarih ve zaman göster. Eğer kesirli kısım yok ise, yanlızca bir tarih göster. Eğer tamsayı bir kısım yoksa, yanlızca zamanı göster. Tarih ve zaman görünümü sizin bölgesel ayarlarınız tarafından tanımlanır.
vbLongDate	1	Tarihi uzun formatta göster.
vbShortDate	2	Tarihi kısa formatta göster.
vbLongTime	3	Zamanı uzun formatta göster.
vbShortTime	4	Zamanı kısa formatta göster.

Yukarıdaki tabloda tarih ve zaman görünümleri sizin sisteminizdeki bölgesel ayarlarınız tarafından tanımlanır.

Değişik Sabitler

Herhangi bir kategoriye uymayan sabitleri tanımlar.

Bu sabitler VBScript içinde tanımlı olduğu için, siz onları kullanmadan önce tanımlamak zorunda değilsiniz. Bu sabitlerin gösterdiği değerleri programınızın heryerinde kullanabilirsiniz.

Sabit	Değer	Tanımı
vbObjectError	-2147221504	Kullanıcı-tanımlı hata numaraları bu değerden daha büyük olmalı, örneğin, Err.Raise Number = vbObjectError + 1000 gibi.

MsgBox Sabitleri

Düğme görünürlüğü, etiketleme, davranış, ve dönüş değerlerini tanımlamak için MsgBox fonksiyonu içinde kullanılan sabitler.

Aşağıdaki sabitler bir mesaj kutusu üzerinde hangi düğmelerin ve ikonların ve hangi düğmenin varsayılan olduğunu belirtmek için kullanılır. Bu sabitler VBScript içinde tanımlı olduğu için, siz onları kullanmadan önce tanımlamak zorunda değilsiniz. Bu sabitlerin gösterdiği değerleri programınızın heryerinde kullanabilirsiniz.

Sabit	Değer	Tanımı
vbOKOnly	0	Yanlızca OK düğmesi görünür.
vbOKCancel	1	OK ve Cancel düğmeleri görünür.
vbAbortRetryIgnore	2	Abort, Retry, ve Ignore düğmeleri görünür.
vbYesNoCancel	3	Yes, No, ve Cancel düğmeleri görünür.
vbYesNo	4	Yes ve No düğmeleri görünür.
vbRetryCancel	5	Retry ve Cancel düğmeleri görünür.
vbCritical	16	Kritik mesaj ikonu görünür.
vbQuestion	32	Uyarı Sorgu ikonu görünür.
vbExclamation	48	Uyarı mesaj ikonu.
vbInformation	64	Bilgi Mesaj ikonu görünür.
vbDefaultButton1	0	İlk düğme varsayılan.
vbDefaultButton2	256	İkinci düğme varsayılan.
vbDefaultButton3	512	Üçüncü düğme varsayılan.
vbDefaultButton4	768	Dördüncü düğme varsayılan.

İzleyen sabitler bir kullanıcının seçtiği düğmenin hangisi olduğunu belirlemek için MsgBox fonksiyonu ile kullanılır. Bu sabitler yanlızca projeniz bu sabitleri içeren

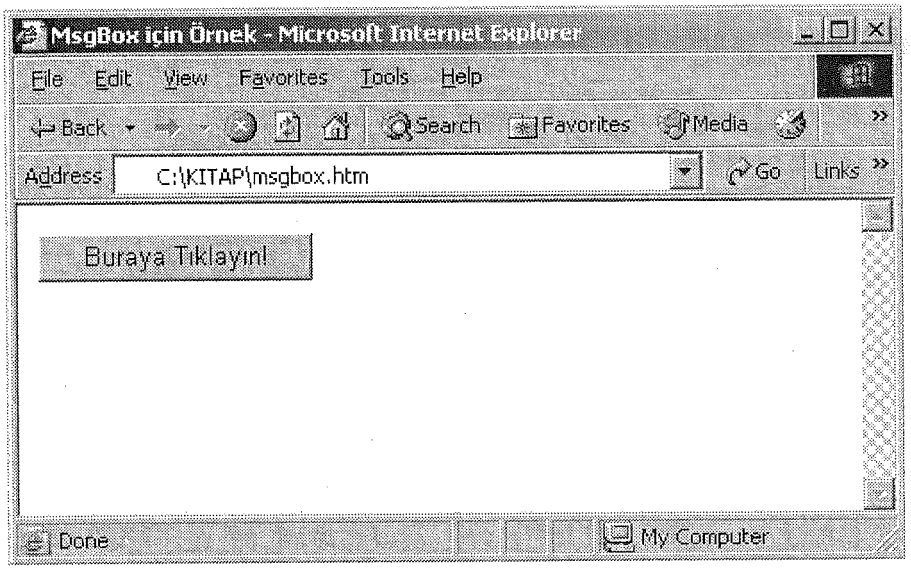
uygun tip kütüphanesi ayrı bir referansa sahip olduğunda kullanılabilir. VBScript için siz ayrı bir şekilde kodunuz içinde bu sabitleri bildirmek zorundasınız.

Sabit	Değer	Tanımı
VbOK	1	OK düğmesine tıklandı.
VbCancel	2	Cancel düğmesine tıklandı.
VbAbort	3	Abort düğmesine tıklandı.
VbRetry	4	Retry düğmesine tıklandı.
VbIgnore	5	Ignore düğmesine tıklandı.
VbYes	6	Yes düğmesine tıklandı.
VbNo	7	No düğmesine tıklandı.

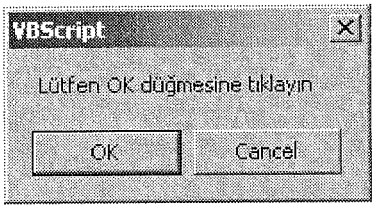
Aşağıdaki örnekte MsgBox ile ekrana Ok ve Cancel düğmeleri olan bir mesaj kutusu gelir.

```
<HTML>
<HEAD>
<TITLE>MsgBox için örnek</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=">
</HEAD>

<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<FORM>
<INPUT TYPE="button" NAME="dugme" VALUE="Buraya tıklayın!">
</FORM>
<SCRIPT LANGUAGE="VBScript" >
Sub dugme_onclick
  MsgBox "Lütfen OK düğmesine tıklayın", VBOKCancel
End Sub
</SCRIPT>
</BODY>
</HTML>
```

Örnek programın tarayıcı üzerindeki çıktısı şekildeki gibidir.



Düğmeye tıkladıktan sonra ekranda izleyen mesaj kutusunu görürüz.



String Sabitleri

İçeriği değiştirilemeyen karakterlerden oluşmuş string sabitlerini tanımlar.

Bu sabitler VBScript içinde tanımlı olduğu için, siz onları kullanmadan önce tanımlamak zorunda değilsiniz. Bu sabitlerin gösterdiği değerleri programınızın heryerinde kullanabilirsiniz.

Sabit	Değer	Tanımı
VbCr	Chr(13)	CR .
VbCrLf	Chr(13) & Chr(10)	CR/LF.
vbFormFeed	Chr(12)	Form besleme

VbLf	Chr(10)	Satır besleme.
vbNewLine	Chr(13) & Chr(10) veya chr(10)	Platform-belirli yeni satır karakteri; her ne ise platform için uygundur.
vbNullChar	Chr(0)	Karekterin sahip olduğu değer 0.
vbNullString	Sıfır değerine sahip string.	Sıfır-uzunluklu stringler gibi değil ("");dış prosedürleri çağırmak için kullanılır.
VbTab	Chr(9)	Yatay tab.
vbVerticalTab	Chr(11)	Dikey tab.

TriState Sabitleri

Sayıları formatlayan fonksiyonlar ile kullanılan sabitleri tanımlar. Bu sabitler yanlızca bu sabitlerin tanımlarını içeren uygun bir tip kütüphanesine (ortaya çıkan nesnelerin, özelliklerin, ve metodların standard tanımlarını içeren başka bir dosya içindeki bileşen veya bir dosyadır.) net olarak işaret ediyorsa o zaman yapmış olduğunuz proje içinde kullanabilirsiniz. VBScript için, kodunuz içinde bu sabitleri bildirmek zorundasınız.

Sabit	Değer	Tanımı
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Default ayarları kullan.

Aşağıdaki tablodaki sabitler VBScript içinde yer aldığı için, siz bunları kullanmadan önce tanımlamak zorunda değilsiniz. Her biri için gösterilen değerleri sunmak için kodunuzun içinde herhangi bir yerinde bunları kullanabilirsiniz.

Sabit	Değer	Tanımı
vbUseDefault	-2	Default ayarları kullan.
vbTrue	-1	True
vbFalse	0	False

VarType Sabitleri

Değişik Variant alt tiplerini tanımlar.

Bu sabitler yanlızca projeniz bu sabit tanımlarını içeren uygun tip kütüphanesine ayrı bir referans ettiği zamam mevcut olur. VBScript için, siz bu sabitleri kodunuz içinde ayrı olarak tanımlamak zorundasınız.

Yazmış olduğunuz kodlar içinde aynı sonuçları almak için ya sabit adlarını yada değerlerini kullanırsınız.

Sabit	Değer	Tanımı
vbEmpty	0	Varsayılan
vbNull	1	Geçerli veri içermez
vbInteger	2	Tamsayı alt tipi
vbLong	3	Long alt tipi
vbSingle	4	Single alt tipi
vbSingle	5	Double alt tipi
vbCurrency	6	Currency alt tipi
vbDate	7	Date alt tipi
vbString	8	String alt tipi
vbObject	9	Nesne
vbError	10	Error alt tipi
vbBoolean	11	Boolean alt tipi
vbVariant	12	Variant (variant dizileri için kullanılır)
vbDataObject	13	Veri erişim nesnesi
vbDecimal	14	Onlu alt tipi
vbByte	17	Byte alt tipi
vbArray	8192	Dizi

VarType() fonksiyonu verilmiş olan bir değişken adının alt tipini belirlemek için kullanılır.

```
<% alttip="Bu bir string alt tipidir" %>

<% =VarType(alttip) %>
```

Çıktısı: 8 (String alt tipi , burada 8 , **vbString** sabitine karşılık gelir)

```
<% alttip=43456 %>

<% =VarType(alttip) %>
```

Çıktısı: 2 (Tamsayı alt tipi, burada 2 , **vbInteger** sabitine karşılık gelir)

Karşılaştırma Sabitleri

Karşılaştırma yaparken karşılaştırmanın hangi bazda yapılacağını belirtmek için izleyen tablodan ya sabit adı yada değeri kullanılır.

Sabit	Değer	Tanımı
VBBinaryCompare	0	İkili karşılaştırma
VBTextCompare	1	Text karşılaştırma
VBDataBaseCompare	2	Veritabanı iç kısımdaki bilgiyi karşılaştır

Aşağıdaki örnekte Add metodu en son satırda başarısız duruma düşüyor çünkü anahtar olan "c" zaten mevcuttur. Eğer CompareMode VBBinaryCompare olarak yeni bir anahtar "B" olarak ayarlanıyorsa o zaman sözlüğe "Cakarta" eklenecektir.

```
<%
Dim sozluk
Set sozluk = CreateObject("Scripting.Dictionary")
sozluk.CompareMode = VBTextCompare
sozluk.Add "a", "Amerika"
sozluk.Add "b", "Bolivya"
sozluk.Add "c", "Canada"
sozluk.Add "d", "Danimarka"
sozluk.Add "C", "Cakarta"  ' burada başarısız olur çünkü text karşılaştırma var.
%>
```

Çıktısı:

"America"

"Bolivya"

"Canada"

"Danimarka"

Dosya Özellikleri Sabitleri

nesne.Attributes [= yeniÖzellik]

Bu özellik bir dosyanın değişik özelliklerini almak veya değiştirmek için kullanılır. mevcut özellik değerleri aşağıda tabloda verilmiştir.

Adı	Değer	Tanımı	Read/Write özelliği
Normal	0	Normal dosya	Read/write
ReadOnly	1	Yanlızca okunabilir dosya	Read only
Hidden	2	Gizli dosya	Read/write
System	4	Sistem dosyası	Read/write
Volume	8	Disk sürücü etiketi	Read only
Directory	16	Folder veya dizin	Read-only
Archive	32	En son yedeklemeden sonra dosya değişti.	Read/write
Alias	64	Bağ veya kısayol	Read-only
Compressed	2048	Sıkıştırılmış dosya	Read-only

Read/Write : Hem yazılabilir hemde okunabilir özellikte olduğunu gösterir.
Read-only : Sadece okunabilir, yazılamaz özellikte olduğunu gösterir.

Aşağıdaki kod bir dosyanın read/write veya read-only olup olmadığını sınamanın nasıl yapılacağını gösterir. Gördüğünüz gibi, mantıksal operatörleri değişik özellikleri almak veya değiştirmek için kullanabiliriz.

```
<%
Dim dosyasis
Dim yazi
Dim okunurdosya
Set dosyasis = CreateObject("Scripting.FileSystemObject")
Set yazi = dosyasis.CreateTextFile("c:\dosya.txt")
yazi.Write "Dosyanın yazılabilir olduğunu bulmak için basit bir test."
yazi.close
Set okunurdosya = dosyasis.GetFile("c:\dosya.txt")
' ReadOnly yerine 1 de yazabilirsiniz
If Not okunurdosya.Attributes And ReadOnly Then
  Response.Write "Dosya Read/Write özelliğine sahiptir."
```

```
Else
  Response.Write "Dosya Read-only özelliğine sahiptir."
End If
%>
```

Çıktısı:

"Dosya Read/Write özelliğine sahiptir."

Dosya Giriş/Çıkış Sabitleri

Dosya giriş/çıkış işlemleri yapmak istediğinizde dosyayı nasıl veya hangi modda açacağınızı belirlemek için tablodaki ya sabit isimlerini yada değerlerini kullanabilirsiniz.

Sabit	Değer	Tanımı
ForReading	1	Dosyayı yanlızca okunabilir olarak açar.
ForWriting	2	Dosyayı yazmak için açar. Eğer dosya mevcut ise, dosya içeriği üzerine yazar.
ForAppending	8	Bir dosyayı açar ve dosya son satırına konumlanır. Dosya içeriği üzerine yazılmaz.

Aşağıdaki program örneği dosya.txt adlı dosyanın yazma işlemi için açılmasını sağlar.

```
<%
Dim dosyasistemi
Dim ornekdosya
Dim textakimi
' bu tip dosyaya yazma işlemleri streaming olarak adlandırılır.
Set dosyasistemi = CreateObject ("Scripting.FileSystemObject")
Set ornekdosya = dosyasistemi.CreateTextFile ("c:\dosya.txt", true)
Set ornekdosya = dosyasistemi.GetFile("c:\dosya.txt")
' dosya açma modu olarak ForWriting yerine 2 de yazabilirsiniz.
```

```
Set textakimi = ornekdosya.OpenAsTextStream ( ForWriting, -2 )
textakimi.Write "Bu dosyadaki halen mevcut olan text bilgi üzerine yazar."
textakimi.Close
%>
```

BÖLÜM
ALTI
6

VBScript Operatörleri

- Aritmetik Operatörler
- Karşılaştırma Operatörleri
- Mantıksal Operatörler
- Boolean Operatörler
- Operatör Öncelikleri

VBScript Operatörleri

VBScript aritmetik operatörler, karşılaştırma operatörleri, birleştirme operatörleri, ve mantıksal operatörler olmak üzere geniş bir operatörler kümesine sahiptir. Bu operatörleri kullanarak değişik kombinasyonlar yapabilirsiniz.

Aritmetik Operatörler

Sayısal ifadeler üzerinde aşağıdaki tabloda verilen operatörleri kullanarak aritmetik işlemler yapabilirsiniz.

Tanımı	Sembol
Üs	^
Tekli eksileme	-
Çarpma	*
Bölme	/
Tamsayı Bölme	\
Modül Aritmetiği	Mod
Toplama	+
Çıkarma	-
String ekleme	&

^ Üs Alma

Bir sayının üstel kuvvetini almak için kullanılır. *sonuc* argümanı herhangi bir sayısal değişken, *sayı* herhangi bir sayısal ifade, ve üs de sayısal bir ifadedir.

```
sonuc = sayı^üs
```

Eğer üs bir tamsayı ise, sayı değişkeni negatif olabilir. Tek bir ifade içinde birden fazla üs alma işlemi yapıldığında, üs soldan sağa doğru olarak yapılacakmış gibi ^ operatörü değerlendirilir. Eğer ya sayı yada üs bir Null ise, sonuçta Null olur.

```
Dim a, b , sonuc

a = 10

b = 4
```

```
sonuc = a^3     ' sonuç 1000 olur

sonuc = b^2     ' sonuç 16 olur
```

Eksileme ve Çıkarma (-)

-1 ile çarpılmış bir sayıyı, veya bağlama bağlı olarak iki sayı arasındaki farkı döndürür. sonuc herhangi bir değişken, sayı herhangi bir sayısal ifade, sayı1 herhangi bir sayısal ifade, ve sayı2 de herhangi bir sayısal ifade.

```
sonuc = sayı1 – sayı2
```

ve

```
-sayı
```

Birinci yazım şeklinde, - operatörü iki sayı arasındaki farkı bulmak için aritmetik çıkarma işlemi yapar. İkinci yazım şeklinde, - operatörü bir ifadenin negatif değerini gösteren tekli eksileme operatörü olarak kullanılır.

Eğer bir veya heriki ifade Null ifade ise, sonuç Null olur. Eğer bir ifade Empty ise, ona 0 mış gibi davranılır.

Çarpma Operatörü (*)

Verilen iki sayının çarpma işlemini yapar. sonuc herhangi bir değişken, sayı1 herhangi bir sayısal ifade, sayı2 de herhangi bir sayısal ifade.

```
sonuc = sayı1 * sayı2
```

Eğer bir veya heriki ifade Null ifadeler ise, sonuç Null olur. Eğer bir ifade Empty ise, o zaman 0 mış gibi davranır.

```
<%

Dim X, Y

X = 10

Y = 20

Response.write X*Y

%>
```

Bölme Operatörü (/)

Verilen iki sayı arasında bir bölme işlemi yapar ve sonuç olarak kayan noktalı bir sayı döner. sonuc herhangi bir değişken, sayı1 herhangi bir sayısal ifade, sayı2 de herhangi bir sayısal ifade.

```
sonuc = sayı1 / sayı2
```

Eğer herhangi bir ifade Null ise, sonuç Null olur. Empty olan herhangi bir ifadeye 0 olarak davranılır.

```
<%

Const BOLEN = 125

Dim Bolunen

Dim Sonuc

Bolunen = 346

Sonuc = Bolunen/BOLEN

Response.write Sonuc

' Sonuç 2,768 olacaktır.

%>
```

Tamsayı Bölme (\ )

İki sayıyı böler ve sonucu bir tamsayı (ondalık kısmı yok) olarak verir. sonuc herhangi bir değişken, sayı1 herhangi bir sayısal ifade, sayı2 de herhangi bir sayısal ifade.

```
sonuc = sayı1 \ sayı2
```

Bölme işlemi yapılmadan önce, sayısal ifade Byte, Integer, veya Long alt tip ifadelere yuvarlanır.

Eğer herhangi bir ifade Null ise, sonuç Null olur. Empty olan herhangi bir ifadeye 0 olarak davranılır.

```
<%

Dim x
```

```
Dim y

x = 201

y = 25

Response.write "Bölme sonucu " & x\y & "dir."

' Bölme sonucu 8 dir.

%>
```

Mod Operatörü

Modül aritmetiğini yapar. İki sayıyı böler ve kalanı döndürür. sonuc herhangi bir değişken, sayı1 herhangi bir sayısal ifade, sayı2 de herhangi bir sayısal ifade.

```
sonuc = sayı1 Mod sayı2
```

Mod operatörü sayı1'i, sayı2 ye (kayan-noktalı sayıları tamsayıya yuvarlar) böler ve kalanı sonuç olarak döndürür. Örneğin, aşağıdaki ifadede, T 5 e eşit olur.

```
T = 19 Mod 6.7
```

Eğer herhangi bir ifade Null ise, sonuçta Null dır. Empty olan herhangi bir ifadeye 0 olarak davranır.

Toplama Operatörü (+)

İki sayının toplam işlemini yapar. sonuc herhangi bir sayısal değişken, ifade1 herhangi bir ifade, ve ifade2 de herhangi bir ifadedir.

```
sonuc = ifade1 + ifade2
```

+ operatörünü iki karakter stringini birleştirmek için kullanabilmenize rağmen, sizin & operatörünü kullanmanız iyi olur.

+ operatörünü kullandığınız zaman, siz bir ekleme veya string birleştirme olup olmayacağını belirleyemezsiniz.

İfadelerin alt tipi aşağıdaki yöntemde + operatörünü davranışını belirler.

Eğer ... ise	İşlem
Herikiside sayısal ifade	Ekleme
Herikiside string ifade	Birleştirme
Bir ifade sayısal ve diğeri string ifade	Ekleme

Eğer bir veya heriki ifade Null ifadeler ise, sonuç Null olur. Eğer heriki ifadede Empty ise, sonuç tamsayı olur. Bunun yanında, eğer yanlızca bir ifade Empty ise, diğer ifade değişmeden sonuç olarak döner.

```
<%

Dim a, b, c, d

a = 100

b = 200

c = "10"

d = "ABC"

Response.write a + b    ' sonuç 300 olur.

Response.write a + c    ' sonuç 210 olur.

Response.write c + d    ' sonuç "10ABC" olur.

%>
```

Birleştirme Operatörü (&)

Verilen iki veya daha çok stringi birleştirir. Siz stringleri birleştirmek için ekleme (+) operatörünü de kullanabilirsiniz, fakat böyle yapmak iyi bir fikir değildir.

```
<%

Const ILK ="Kasabanın"

Const IKINCI ="Sırrı"

Dim kitap

Kitap = ILK & IKINCI & "2. Baskı"
```

```
Response.write Kitap

' Çıktısı "Kasabanın Sırrı 2. Baskı"

%>
```

Karşılaştırma Operatörleri

Tanımı	Sembol
Eşitlik	=
Eşitsizlik	<>
Daha küçük	<
Daha büyük	>
Daha küçük veya eşit	<=
Daha büyük veya eşit	>=
Nesne Eşitliği	Is

Eşitlik ve Atama Operatörü (=)

Bir değişken veya bir özelliğe bir değer atar. VBScript içinde, siz iki ifade arasında bir eşitlik olup olmadığınıda sınamak içinde "=" operatörünü kullanabilirsiniz. degisken herhangi bir değişken adı veya yazılabilir bir özellik, *deger* ise herhangi bir sayısal veya string ivedi değer, sabit veya ifade olabilir.

```
degisken = deger
```

Eşitlik işaretinin sol tarafındaki isim basit skalar bir değişken veya bir dizinin bir elamanı olabilir. Eşitlik işaretinin sol tarafındaki özellikler yanlızca çalışma zamanında yazılabilir özellikler olabilir.

```
Dim x, y

x= 10

y = 20

x = y

Response.write x

' x=20 olur
```

= operatörü bir değişkene bir değer atamak için kullanılır, veya stringleri veya sayıları karşılaştırmak, veya bir değişkenin değerini yazdırmak için kullanılır.Atama için

kullanıldığında, = operatörünün sol tarafındaki tek bir değişken sağ taraftaki bir veya daha çok değişken ve/veya ifadeler ile tanımlanan değer verilir.

```
<% numara = 5.0 + 7.9*(444.999 / 456.9) %>

<% string = "Bugün hava çok güzel." %>

<% degisken = xxx + yyy + zzz %>

<% dizi(40) = 12345 %>
```

= operatörü iki sayı veya stringi karşılaştırmak ve aynı olup olmadığını görmek içinde kullanılır. Stringler için, karşılaştırmada büyük küçük harf ayırımı yapılır.

```
<% ="Elma"="Elma" %> True

<% ="eLMa"="ElmA" %> False

 <% =123=123 %> True

<% =5.67=98.7 %> False
```

= operatörü bir değişkenin değerini yazdırmak için kullanılır, ifade <% %> ile çevrelenmeli. = operatörü ve değişken arasında boşluklar olabilir.

```
<% arabam = "Opel Astra Elegance" %>

<% =arabam %>  Opel Astra Elegance

<% = arabam %> Opel Astra Elegance
```

Eşitliksizlik Operatörü (<>)

<> operatörü eşit değil veya eşitsizlik operatörü olarak bilinir.

```
<%

If 128 <> 777 Then

  Response.Write("Eşit değil") 'Eşit değil

End If

%>
```

Daha Küçük Operatörü (<)

< operatörü birinci ifadenin ikinci ifadeden daha küçük olup olmadığını belirler.

```
<%
123 < 456   'True olur
"a" < "A"   'False olur
%>
```

Daha Büyük Operatörü (>)

> operatörü birinci ifadenin ikinci ifadeden daha büyük olup olmadığını belirler.

```
<% 123 > 456 %> False
<% "a" > "A" %> True
```

Daha Küçük veya Eşit Operatörü (<=)

<= operatörü birinci ifadenin ikinci ifadeden daha küçük veya eşit olup olmadığını belirler.

```
<%
123 <= 456  'True olur
"a" <= "A"  'False olur
%>
```

Daha Büyük veya Eşit Operatörü (>=)

>= operatörü birinci ifadenin ikinci ifadeden daha büyük veya eşit olup olmadığını belirler.

```
<% 123 >= 456 %> False
<% "a" >= "A" %> True
```

Is Operatörü

Nesne eşitliğini test eder. Eğer her iki nesnenin işaretçisi aynı nesneye işaret ediyorsa True değerini döndürür.

```
sonuc = nesne1 Is nesne2
```

sonuc herhangi bir sayısal değişken, nesne1 herhangi bir nesne, ve nesne2 de herhangi bir nesnedir. Eğer nesne1 ve nesne2 aynı nesneye işaret ediyorsa, sonuç Ture olur; eğer aynı nesneye işaret etmiyorlarsa sonuç False olur. İki değişken birkaç şekilde aynı nesneye işaret ettirilebilir.

Aşağıdaki örnekte, Nesne1 ,aynı nesne olarak Nesne2 ye işaret ettiriliyor.

```
Set Nesne1 = Nesne2
```

Aşağıdaki örnek Nesne1 ve Nesne2 aynı nesne olarak Nesne3 ünde işaret ettirilmesi sağlanıyor.

```
Set Nesen1 = Nesne3

Set Nesne2 = Nesne3
```

Mantıksal Operatörler

Tanımı	Sembol
Mantıksal değil	Not
Mantıksal ve	And
Mantıksal veya	Or
Mantıksal darveya	Xor
Mantıksal eşitlik	Eqv
Mantıksal karıştırma	Imp

Not Operatörü

Bir ifadeyi mantıksal olarak değilini yapmak için kullanılır. sonuc herhangi bir değişken , ve ifade ise herhangi bir ifadedir.

```
sonuc = Not ifade
```

Aşağıdaki tablo sonucun nasıl belirleneceğini gösterir.

İfade	Sonuç
True	False
False	True
Null	Null

Ek olarak, Not operatörü herhangi bir değişkenin bitlerini tersine çevirir ve aşağıdaki tabloya benzer bir sonuç ortaya çıkar.

İfade bitleri	Sonuçtaki bitler
0	1
1	0

And Operatörü

İki ifade üzerinde bir mantıksal ve işlemi yapmak için kullanılır. sonuc herhangi bir sayısal değişken, ifade1 herhangi bir ifade, ve ifade2 de herhangi bir ifadedir.

```
sonuc = ifade1 And ifade2
```

Eğer heriki ifade True ise, o zaman sonuç True olur. Diğer durumlarda ise sonuç False olacaktır.

Aşağıdaki tablo sonucun nasıl belirleneceğini gösterir.

Eğer ifade1	ve ifade2 ise	Sonuç
True	True	True
True	False	False
True	Null	Null
False	True	False
False	False	False
False	Null	False
Null	True	Null
Null	False	False
Null	Null	Null

And operatörü iki sayısal ifadede belirli pozisyonlardaki bitlerin bit seviyesinde karşılaştırmasını da yapar.

Aşağıdaki tablo And operatörüne göre sonuçların nasıl olduğunu gösterir.

Eğer ifade1	ve ifade2 ise	Sonuç
0	0	0
0	1	0
1	0	0
1	1	1

Not: Yukarıdaki tabloda ifadeler ve sonuç bit seviyesindedir.

Eqv Operatörü

İki ifade üzerinde mantıksal bir eşitlik olup olmadığını kontrol etmek ve bir sonuç üretmek için kullanırız. sonuc herhangi bir sayısal değişken, ifade1 herhangi bir ifade, ve ifade2 de herhangi bir ifadedir.

```
sonuc = ifade1 Eqv ifade2
```

Eğer herbir ifade Null ise, sonuç da Null olur. Hiçbir ifade Null değilse, o zaman sonuç aşağıdaki tabloya göre belirlenir.

Eğer ifade1	ve ifade2 ... ise	Sonuç
True	True	True
True	False	False
False	True	False
False	False	True

Eqv operatörü iki ifade içindeki belirli pozisyonlardaki bitlerin bit bazında karşılaştırmasını yapar ve aşağıdaki tabloya bağlı olarak sonuçta bitleri ayarlar.

Eğer ifade1 içindeki bit	ve ifade2 içindeki bit ... ise	Sonuç
0	0	1
0	1	0
1	0	0
1	1	1

Imp Operatörü

İki sayı üzerinde mantıksal bir kesinliği yapmak için kullanılır. sonuc herhangi bir sayısal değişken, ifade1 herhangi bir ifade, ve ifade2 de herhangi bir ifadedir.

```
sonuc = ifade1 Imp ifade2
```

Aşağıdaki tablo sonucun nasıl belirleneceğini gösterir.

Eğer ifade1	ve ifade2 ise	Sonuç
True	True	True
True	False	False
True	Null	Null
False	True	True
False	False	True
False	Null	True
Null	True	True
Null	False	Null
Null	Null	Null

Eqv operatörü iki ifade içindeki belirli pozisyonlardaki bitlerin bit bazında karşılaştırmasını yapar ve aşağıdaki tabloya bağlı olarak sonuçta bitleri ayarlar.

Eğer ifade1	ve ifade2 ise	Sonuç
0	0	1
0	1	1
1	0	0
1	1	1

Not: Yukarıdaki tabloda ifadeler ve sonuç bit seviyesindedir.

Or Operatörü

Boolean veya Mantıksal kullanımı ile iki ifadeyi karşılaştırmak için kullanılır. sonuc herhangi sayısal bir değer, ifade1 ve ifade2 herhangi bir ifadedir. Eğer ifadelerden her ikisi veya biri True ise, sonuç True olacaktır.

```
sonuc = ifade1 Or ifade2
```

Aşağıdaki tablo sonucun nasıl belirleneceğini gösterir.

İfade1 ...	İfade2	Sonuç
True	True	True
True	False	True
True	Null	True
False	True	True
False	False	False
False	Null	Null
Null	True	True
Null	False	Null
Null	Null	Null

Or operatörü iki sayısal ifade belirli pozisyonlardaki bitlerin bit seviyesinde karşılaştırmasını da yapar.

Aşağıdaki tablo Or operatörüne göre sonuçların nasıl olduğunu gösterir.

İfade1	İfade2	Sonuç
0	0	0
0	1	1
1	0	1
1	1	1

Not: Yukarıdaki tabloda ifadeler ve sonuç bit seviyesindedir.

Xor Operatörü

Bir boolean dar veya (XOR) işlemini yapar. sonuc herhangi bir sayısal değişken, ifade1 ve ifade2 herhangi bir ifadedir.

```
sonuc = ifade1 Xor ifade2
```

Eğer yanlızca biri True ise, sonuç True olacaktır. Bunun yanında, eğer biri Null ise, sonuç da Null olur. İfadelerden hiçbiri Null değilse, sonuç aşağıdaki tabloya göre belirlenir.

İfade1	İfade2	Sonuç
True	True	False
True	False	True
False	True	True
False	False	False

Xor operatörü iki sayısal ifade belirli pozisyonlardaki bitlerin bit seviyesinde karşılaştırmasını da yapar.

Aşağıdaki tablo Xor operatörüne göre sonuçların nasıl olduğunu gösterir.

İfade1	İfade2	Sonuç
0	0	0
0	1	1
1	0	1
1	1	0

Not: Yukarıdaki tabloda ifadeler ve sonuç bit seviyesindedir.

Boolean İşlemleri

Çoğu programcılar er geç boolean lojiğini kullanmak zorunda kalırlar. Boolean işlemleri ve bilgisayarlar birbiri için yapılmıştır çünkü 1 ve 0 bitleri arasında bir boolean işleminin sonucu için yanlızca mümkün olan iki değer vardır.

Bilgisayar değerleri ikili sayılar olarak tutar. Bir ikili sayı bir bit dizisidir. Sola doğru hareket etmek ile, herbir bit yanlızca 1 ve 0 dan oluşan iki değerden birini alır. Ayarlandığında (sıfır olmayan) , n nin 0 ile başlayan pozisyonun olduğu yerdeki her bir bit 2 üzeri n değerine sahiptir. 8 bitlik bir sayıda en sağdaki bit 2 nin 0 ncı kuvveti veya 1 dir. Sonraki bit 2 nin 1 nci kuvveti , veya 2 dir ve böylece sola doğru gider. Burada herbir bit değeri sağındakinin iki katıdır. Bu yüzden 12 nin ikilik tabandaki değeri 00001100 dır.

VBScript dört boolean işlemini destekler: AND, OR, NOT ,ve XOR. Siz iki değerdeki bit dizisinin aynı olup olmadığını görmek için AND işlemini yapmalısınız. Herbir pozisyondaki bit aynı ise sonuç da 1 olacaktır aksi taktirde 0 olacaktır.

Operatör Öncelikleri

Bir ifadede birkaç işlem meydana geldiğinde, operatör önceliği olarak adlandırılan önceden tanımlı sırada her bir parça değerlendirilir ve çözülür. Siz parantezleri kullanarak diğer ifadelerden daha önce olacak şekilde bir önceliğe sahip olarak işlem görmesini istediğiniz ifadelere öncelik kazandırabilirsiniz. Parantez içindeki operatörler daima parantez dışındaki ifadelerden önce işleme alınırlar. Parantezler ile, bunun yanında, standard operatör önceliğide değiştirilebilir.

İfadeler bir kategoriden daha çoğunda operatörler içeriyorsa, aritmetik operatörler, önce değerlendirilir, sonra karşılaştırma operatörleri değerlendirilip, ve lojik operatörler en sonda değerlendirilir. Tüm karşılaştırma operatörleri eşit önceliğe sahiptir; bu operatörler göründükleri sırada soldan sağa doğru değerlendirilir. Aritmetik ve lojik operatörler aşağıdaki öncelik sırasına göre değerlendirilir.

Aritmetik Operatörler

Tanımı	Sembol
Üs	^
Tekli eksileme	-
Çarpma	*
Bölme	/
Tamsayı Bölme	\
Modül Aritmatiği	Mod
Toplama	+
Çıkarma	-
String ekleme	&

BÖLÜM
YEDİ
7

VBScript Koşullu İfadeleri

- Program Akışını Kontrol Etme
- If...Then İfadesi
- If...Then...Else İfadesi
- If...Then...ElseIf İfadesi
- Select Case İfadesi

VBScript Koşullu İfadeleri

Her modern dil birçok mantıksal ifadeye sahiptir. Mantıksal ifadeler size koşullu olarak kodları yürütmenize izin verir. Tamamlanmış bir ifade bir kod bloğu olarak şekillenir. Koşullu ifadeleri kullanarak yapmış olduğumuz programın akışını, belirli noktalarda belirli durumlara göre değiştirebiliriz.

Program Akışını Kontrol Etme

Siz koşullu yapılar ve çevrim yapıları ile sizin scriptinizin akışını kontrol edebilirsiniz. Koşullu yapıların kullanımıyla, siz karar verebilen ve tekrarlar yapabilen aksiyonları VBScript kod ile yazabilirsiniz. Aşağıdaki koşullu yapılar VBScript içinde mevcuttur.

- If...Then İfadesi
- If...Then...Else İfadesi
- If...Then...ElseIf İfadesi
- Select Case İfadesi

If...Then İfadesi

If koşullu ifadesi sadece tek bir koşul doğru ise o zaman bir grup ifadeyi yürütür. Diğer koşullu ifadelerin iç kısmınada If ifadesini yerleştirebilirsiniz. If ve End If ifadesini birlikte kullanmak zorundasınız yoksa bir hata mesajı alırsınız. Then ,If ile aynı satırda olmak zorundadır.

Örnekler,

```
<%
If tarih = "4/04/2002" Then
  Response.Write tarih
End If
%>
```

Bir başka örnek

```
<%
```

```
If birsayi < 25 Then
  Response.Write "Oda sıcaklığı 25 dereceden az"
End If
%>
```

If...Then...Else İfadesi

If...Then...Else yapısı bir veya daha fazla yapının yürütülmesi için bir koşulun sonuca bağlı olarak True veya False olup olmadığını değerlendirmek için kullanılır. Genellikle koşul bir değeri veya değişkeni diğeri ile karşılaştırmak için bir karşılaştırma operatörü kullanan bir ifadedir. Karşılaştırma operatörleri hakkında daha fazla bilgi için, karşılaştırma operatörlerine bakınız. If...Then...Else yapıları ihtiyaç duyduğunuz çok sayıda seviyede iç içe kullanabilirsiniz.

Örnekler,

```
Dim i
i = 1
If i =1 Then
' koşul doğru ise yürütülecek kod bloğu buraya yazılır
End If
```

Aşağıda bir veya daha çok ElseIf yapısı eklenerek daha karmaşık bir koşullu yapı elde edildi.

```
Dim i
i = 1
If i =1 Then
' eğer i eşit 1 ise yürütülecek kod bloğu
ElseIf i= 2 Then
' eğer i eşit 2 ise yürütülecek kod bloğu
End If
```

Birçok durumda If ve ElseIf koşullarının başarısız olmasında siz birkaç kodu yürütmek istersiniz. Bu işi yapmak için bir Else koşulunu kullanın.

Örneğin:

```
Dim i

i = 1

If i =1 Then

' eğer i eşit 1 ise yürütülecek kod bloğu

ElseIf i= 2 Then

' eğer i eşit 2 ise yürütülecek kod bloğu

Else

' eğer i  1 ve 2 den farklı ise yürütülecek kod bloğu

End If
```

Eğer bir koşul True ise yapıları koşturma

Bir koşul True olduğunda yanlızca bir yapıyı yürütmek için, If...Then...Else yapısı için tek-satır yazım şeklini kullanın. Aşağıdaki örnek tek-satır yazım şeklini gösterir. Bu örnekte Else anahtar sözcüğünün kullanılmadığına dikkat ediniz.

```
Sub tarihAt()
   Dim tarih
   tarih = #2/13/95#
   If tarih < Now Then tarih = Now
End Sub
```

Bir satırdan daha çok kodu koşturmak için, siz çok-satırlı (veya blok) yazım şeklini kullanmak zorundasınız. Bu yazım şekli aşağıdaki örnekte görüldüğü gibi End If yapısını içine dahil eder:

```
Sub mesajgoster( deger)

If deger = 0 Then
```

```
  mesaj.ForeColor = vbRed

  mesaj.Font.Bold = True

  mesaj.Font.Italic = True

End If

End Sub
```

Eğer bir koşul True ise tüm ifadeleri yürütme ve eğer koşul False ise diğerlerini yürütme

Siz If...Then...Else yapısını yürütülebilir yapıların iki bloğunu tanımlamak için kullanabilirsiniz: bir blok eğer koşul True ise diğeri ise koşul False olduğunda yürütülecek bloktur.

```
Sub mesajgoster( deger)

  If deger = 0 Then

    mesaj.ForeColor = vbRed

    mesaj.Font.Bold = True

    mesaj.Font.Italic = True

  Else

    mesaj.Forecolor = vbBlack

    mesaj.Font.Bold = False

    mesaj.Font.Italic = False

  End If

End Sub
```

If ...Then...ElseIf ... İfadesi

If...Then...Else yapısının bir değişik şekli birkaç alternatifden seçmenize izin verir. ElseIf cümleciği eklenerek If...Then...Else yapısının fonksiyonelliği genişletilir bundan dolayı siz farklı olabilirliklerden program akışını kontrol edebilirsiniz.

Örneğin:

```
Sub Sonuclar( deger)
```

```
If deger= 0 Then
  MsgBox deger
ElseIf deger = 1 Then
  MsgBox deger
ElseIf deger = 2 then
  Msgbox deger
Else
  Msgbox "Değer geçersiz!"
End If
End Sub
```

Siz alternatif seçenekler sağlamak için ihtiyacınız kadar çok ElseIf cümleciği ekleyebilirsiniz. ElseIf cümleciğinin geniş kullanımı sık sık yavaşlatmaya neden olur. Birçok alternatif arasından seçmenin en iyi yolu Select Case yapısının kullanılmasıdır.

Select Case İfadesi

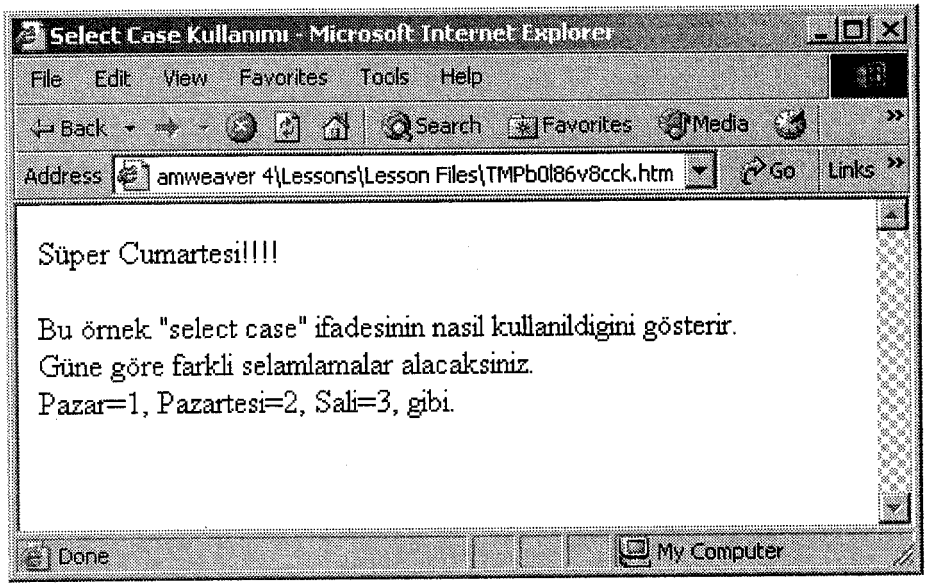
Select Case yapısı çok sayıda blok yapıları arasından bir blok yapının seçilerek yürütülmesi için If...Then...Else yapısına bir alternatif sağlar. Bir Select Case yapısı If...Then...Else yapısına benzer bir yetenek sağlar, fakat o kodu daha etkin ve okunabilir yapar.

Bir Select Case yapısı yapının en üstünde yer alan ve birkez değerlendirilebilecek bir tek test ifadesi ile çalışır. İfadenin sonucu o zaman yapı içindeki herbir Case için değerler ile karşılaştırılır. Eğer bir uyum varsa, o Case ile ile iliştirilmiş yapıların bloğu yürütülür, aşağıdaki örnekteki gibi:

```
<HTML>
<BODY>
<SCRIPT TYPE="text/vbscript">
```

```
gun=WeekDay(date)
Select Case gun
 Case 1
  Document.Write("İyi uykular pazar")
 Case 2
  Document.Write("Günaydın!")
 Case 3
  Document.Write("Bugün salı!")
 Case 4
  Document.Write("Çarşamba!")
 Case 5
  Document.Write("Perşembe...")
 Case 6
  Document.Write("Nihayet cuma!")
 Case else
  Document.Write("Süper Cumartesi!!!!")
End Select
</SCRIPT>
<P>Bu örnek  "select case" ifadesinin nasıl kullanıldığını gösterir.<BR>
Güne göre farklı selamlamalar alacaksınız.<BR>
Pazar=1, Pazartesi=2, Salı=3, gibi.</P>

</BODY>
</HTML>
```

Verilen program kodunun tarayıcı üzerindeki görünümü şekildeki gibidir.



Select Case yapısı yapının en üstündeki bir ifadeyi değerlendirir. If...Then...Else yapısı her bir ElseIf yapısı için farklı bir ifadeyi değerlendirebilir. Yanlızca herbir ElseIf yapısı aynı ifadeyi değerlendiriyorsa o zaman If...Then...ElseIf yapısı yerine Select Case yapısı kullanılabilir.

BÖLÜM SEKİZ 8

VBScript ve HTML Formları

- HTML Nedir
- Form Nedir
- Form Özellikleri
- Form Olay Kotarıcıları
- Form Elemanları
- TextBox Elemanı
- Command Button Elemanı
- CheckBox Elemanı
- Radio Button Elemanı
- TextArea Elemanı
- Formların Onaylanması
- Sayısal Değerlerin Kullanımı
- Form Bilgilerini Sunucuya Gönderme

VBScript ve HTML Formları

HTML Nedir

HTML (HyperText Markup Language) kısaltması olup web sayfalarının tasarımında metin işaretleme dili olarak kullanılır. İşaretleme asıl dökümandan ayrıdır. HTML Web tarayıcılarına sayfa tasarımında sayfaların nasıl görüntüleneceklerini bildirmek için kullanılır. Tarayıcıyı kullanarak HTML kodları ile biçimlendirilmiş bir web sayfasını ziyaret ettiğinizde ,tarayıcı tarafından uygun bir şekilde yorumlanarak bir görünüm elde edilir bu da size web sayfasının görünümü olarak yansıyacaktır.

Burada anlatılmak istenen HTML deki dökümanların nasıl üretiminin yapıldığına dair bir giriş niteliği taşır. HTML ,web sayfası tasarımında kullanılıyor olması nedeniyle HTML kullanımı ve web sayfası tasarımı için dosyaların nasıl oluşturulacağı kısaca anlatılacaktır.

HTML yapısal dökümanlar oluşturmanıza izin verir. Başlık komutları dökümanınızı ayırır ve kategorize eder, aynı zamanda text ve resimlerin biçimlendirilmesi ve gösterilmesi, kullanıcıdan bilgi girdisi, ve bilgiyi sunucuya göndermek gibi işlemleri yapar. Buna ek olarak text veya resimler üzerinde bir fare tıklaması ile gidebileceğiniz özel alanlar oluşturmanıza ve alanlardan kendi içine veya başka bir HTML dosyaya bir hiperbağ oluşturmanıza izin verir. Böylece çok sayıda HTML dosyası arasında bir bağ oluştururuz.

HTML kodları ile karar verme imkanı olmamasına rağmen, web sayfası iki önemli amaca hizmet eder:

1. Programcı olmayanların bile faydalı bilgiler ile dolu etkileşimli siteler yapmaları için bir yoldur.
2. Internet ile bağlantısı ile, bilgilerin global olarak herkes için mevcut olmasını sağlar.

HTML neden önemlidir?

HTML olmadan önce, bilgi ile dolu herkesin okuyabileceği bir ekran oluşturmak çok zordu. Birincisi, bir program yazmadan veya PowerPoint kullanmadan bir sunum yapmak veya herkesin okuyacağı bir bilgiyi göstermek çok zordu. İkincisi, yazılan bu programlar işletim sistemi ortamına bağımlıdır, oysa HTML işletim sistemi ortamından bağımsızdır. Bu gibi önemli kolaylıklar sağlayan HTML milyonlarca insanın bilgilere kolay bir şekilde erişimini sağlar.

Form Nedir

Web formları okuyucuya bilginin birkaç aksiyon için sunucuya gönderilmesini sağlar. Örneğin, birçok isim ve e-mail adres alıyorsunuz web sayfası üzerinden bunları kayıt etmek için sunucuya gönderilmesi gerekir bunun için bu sayfa üzerindeki formu submit ederek sunucuya göndermemiz gerekir. Gönderilen bilgiler bir veri tabanında saklanarak daha sonra bu bilgilere erişim imkanı sağlanır.

Sunucuya gelen bilgiler üzerinde işlemler genellikle script veya programlar tarafından yapılır, VBScript, Perl, JavaScript gibi, bu diller text, dosyalar, ve bilgiler üzerinde işlemler yapmak için kullanılır.
Form elemanı bir kullanıcı tarafından doldurulabilecek bir giriş alanı , etiketler ve menüler tanımlar.

<FORM> ve </FORM> etiketleri arasında tanımlamalar yapılır. <FORM> etiketi çok sayıda özelliğe sahiptir , aşağıda bunlardan en çok kullanılan birkaç tanesi verilmiştir:

Form Özellikleri

ACTION = "URL"

Bu nitelik form içeriğini işleyecek olan sunucu üzerinde olan script veya programın URL 'sini içerir.

METHOD =" GET ! POST "

Bu yöntem form üzerindeki bilgilerin sunucuya nasıl gönderileceğini belirtir. Bunun için iki seçenek var "GET" ve "POST", bu metodlar bilgileri URL ye ekler ve sunucuya gönderir GET ile bu uzunluk sınırlı ve kriptolama yapılmadığı için güvensiz, fakat POST ile gönderilecek bilgi daha uzun olabilir ve gönderilen bilgi kriptolandığı için güvenlidir.

TARGET="_blank | çerçeveadı | _parent | _self | _top"

Sunucuya gönderilen bilginin işlendikten sonraki cevap niteliğindeki bilgiyi nerede gösterileceğini belirler, _self formu gönderen pencerede gösterilmesini, _blank yeni ve boş bir sayfada gösterilmesini, _parent çerçeve içeren pencerede gösterilmesi, ve _top tam tarayıcı penceresinde gösterilmesini sağlar. Siz bu seçenekleri Tarayıcınızda deneyerek bilgilerinizi pekiştirebilirsiniz.

Form Olay Kotarıcıları

OnClick ,onMouseDown, onMouseUp,onMouseOver,onMouseOut,onSubmit ="script"
İzleyen örnekte, form elemanı üzerinde 3 adet text elemanı ve bir adet submit düğmesi tanımlanmıştır. Form doldurulduktan sonra submit ettiğimizde

"http://www.pcs.com/script/adres.asp" URL içinde belirtilen sunucu adresine gidilerek adres.asp script dosyasına ulaşılır bu script dosya gönderilen form bilgileri üzerinde gerekli işlemleri yapar ve eğer gerekli ise istemciye geri dönüş bilgisi gönderir.

<FORM> ve </FORM> etiketleri arasında form elemanları tanımlanmıştır buna dikkat edin.

Ayrıca burada forma "form1" adı verilmiştir, bu ad VBScript ile HTML kodlarının birlikte kullanımında işinize çok yarayacaktır.

ACTION="adres.asp"

"adres.asp" dosyası html dosyası ile aynı dizindedir.

METHOD="POST" olarak tanımlanmıştır, burada POST yerine GET 'de kullanabiliriz.

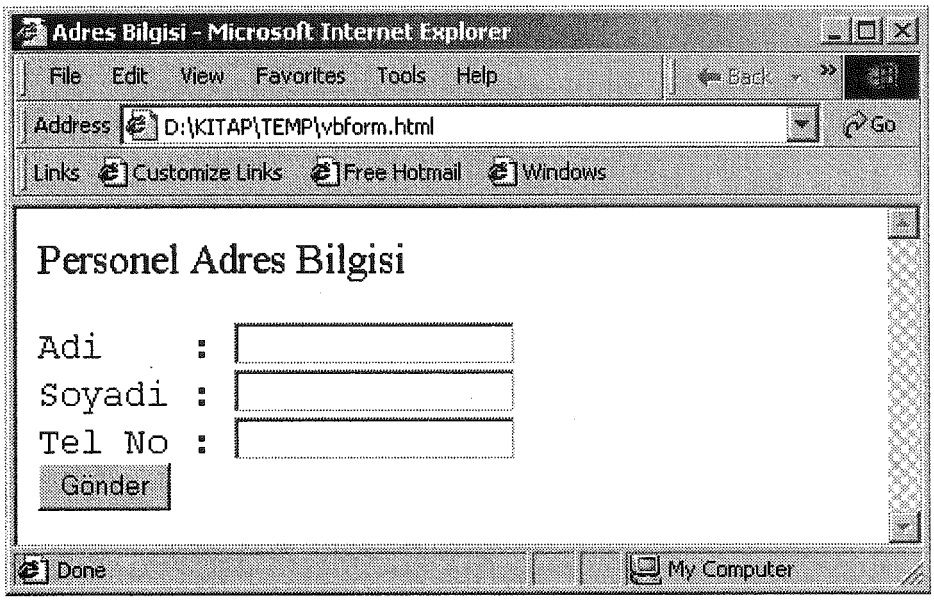
HTML dosyası içinde ayrıca <PRE> ve </PRE> etiketleri kullanılmıştır, bunun kullanımının sebebi, kapsadığı metinin önceden biçimlendirildiğini, yani boşluk, ve diğer biçimlendirme karakterlerinin aynen sayfa üzerindede olmasını sağlar, bu şekilde güzel bir görünüm elde ederiz.

İşte örneğimizin tam HTML kodlaması:

```
<HTML>

<HEAD>

<TITLE>Adres Bilgisi</TITLE>

<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

</HEAD>

<BODY bgColor="#FFFFFF" TEXT="#000000">

<FORM NAME="form1" METHOD="POST" ACTION="adres.asp">

PERSONEL ADRES BİLGİSİ

<PRE>

Adı    : <INPUT TYPE="text" NAME="adi" MAXLENGTH="20">

Soyadı : <INPUT TYPE="text" NAME="soyadi" MAXLENGTH="20">

Tel No : <INPUT TYPE="text" NAME="telno" MAXLENGTH="15">

<INPUT TYPE="submit" NAME="Submit" VALUE="Gönder">
```

```
</PRE>
</FORM>
</BODY>
</HTML>
```

Önceki örnekteki HTML dosyasının web tarayıcısı üzerindeki görünümü:



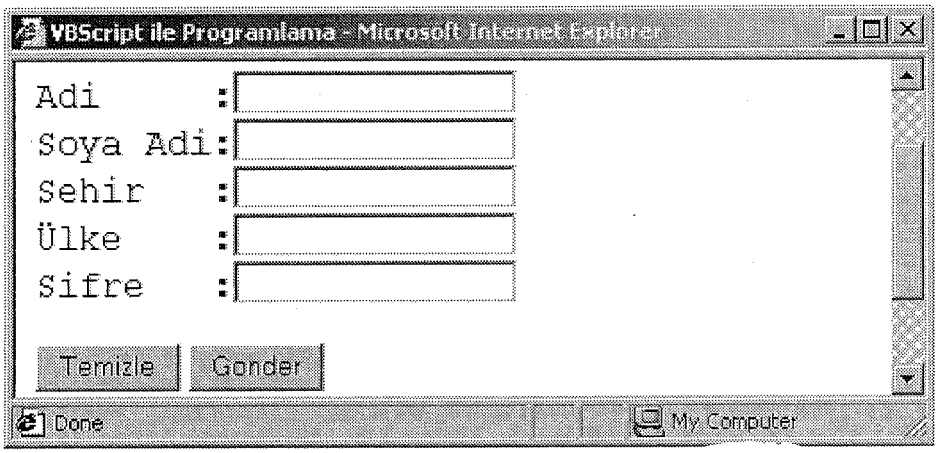
Örnek 2:

İkinci örnekte "GET" metodu kullanılmıştır.

```
<HTML>
<HEAD>
<TITLE>VBScript ile Programlama</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</HEAD>
<BODY>
```

```
<FORM NAME="ad" ACTION="bilgi.asp" METHOD="GET">

<H2>Bilgileri Doldurunuz</H2>

<A HREF="default.htm">Geri git</A>

<PRE>
Adi     :<INPUT TYPE="TEXT" NAME="Ad">
Soya Adi:<INPUT TYPE="TEXT" NAME="Soyad">
Sehir   :<INPUT TYPE="TEXT" NAME="Sehir">
Ülke    :<INPUT TYPE="TEXT" NAME="Ulke">
Sifre   :<INPUT TYPE="password" NAME="Sifre">
</PRE>

<INPUT TYPE="RESET" VALUE="Temizle">

<INPUT TYPE="SUBMIT" VALUE="Gonder">

</FORM>

<IMG SRC="help.gif">

<A HREF="deneme.html">Bilgial</A>

</BODY>

</HTML>
```

Önceki örnekteki HTML dosyasının web tarayıcısı üzerindeki görünümü:



Form Elemanları

Bir eleman bir form için bir giriş kontrolünü belirtir. Giriş tipi TYPE özelliği ile belirtilir ve bu tip belirtimi ile text (tek satır veya çok satırlı) alanı, düğme, şifre, onay kutusu, seçenek düğmesi gibi form elemanları tanımlamamıza izin verir.

Text Box Elemanı

TYPE="text" ile belirtilen TEXT giriş elemanı form üzerinde bir veri giriş alanıdır. Bu elamanın tanımı aşağıda verilmiştir. Bir TEXT elemanına bir isim (NAME) verebilirsiniz, ne kadar uzunlukta (MAXLENGTH) veri girişinin olacağını belirleyebilirsiniz, ayrıca ilk değer ataması yapabilirsiniz (VALUE).

```
<INPUT TYPE="text" NAME="textfield" MAXLENGTH="20" VALUE="TEXT">
```

Command Button Elemanı

Genel amaçlı bir düğme tanımı yapabilirsiniz, fakat bu düğmenin etkisini görmek için onClick olay kotarıcısı ile ilişkilendirmek gerekir. Düğmeye tıklandığında onClick olay kotarıcısının belirlediği script yürütülerek istediğimiz işlemi yaptırabiliriz.

Bir düğme elemanına isim (NAME) verebilirsiniz, ki bu isim DOM modeline göre VBScript kod içinde çok kullanılacaktır, düğme üzerine etiket bilgisi (VALUE) koyabilirsiniz, ayrıca bir olay kotarıcı ile ilişkilendirebilirsiniz.

Ayrıca TYPE="submit" belirtimi ile formu sunucuya gönderen bir düğme tanımı yapabilirsiniz.

```
<INPUT TYPE="button" NAME="adres" VALUE="Gönder" onClick="script">

<INPUT TYPE="submit" NAME="adres" VALUE="Gönder">
```

CheckBox Elemanı

Bu eleman bir onay kutusunu kontrolünü gösterir, onay kutusu form denetimlerinin işaretlenmiş veya işaretlenmemiş durumları vardır, ancak bu denetimler gruplandığında kullanıcının bir defada birkaç kutuyu işaretlemesine izin verir.

Bir onay kutusuna bir isim (NAME) verebilirsiniz, etiket bilgisi (VALUE) koyabilirsiniz, ve ilk değer olarak işaretli olup olmamasını (CHECKED) belirleyebilirsiniz.

<INPUT TYPE="checkbox" NAME="checkbox" VALUE="checkbox" CHECKED>

Radio Button Elemanı

Radyo düğmesi bir seçenek düğmesini gösterir. seçenek düğmeleri gruplandığında bir anda sadece bir tanesinin işaretlenmesine izin verir.

Bir radyo düğmesine bir isim (NAME) verebilirsiniz, etiket bilgisi (VALUE) koyabilirsiniz, ve ilk değer olarak işaretli olup olmamasını (CHECKED) belirleyebilirsiniz.

<INPUT TYPE="radio" NAME="radiobutton" VALUE="radiobutton" CHECKED>

TextArea Elemanı

Bu eleman form üzerinde çok sayıda satır girişine imkan sağlaması için kullanılır. Bu elemanı kullanarak bir şey hakkında yorum alabiliriz, veya ona bu alanda çok satırdan oluşan bilgiler verebiliriz.

Örneğin,

<TEXTAREA NAME="coksatir" COLS="50" ROWS="10">

Bu eleman 50 kolon ve 10 satırdan oluşan bir giriş alanını tanımlar.

</TEXTAREA>

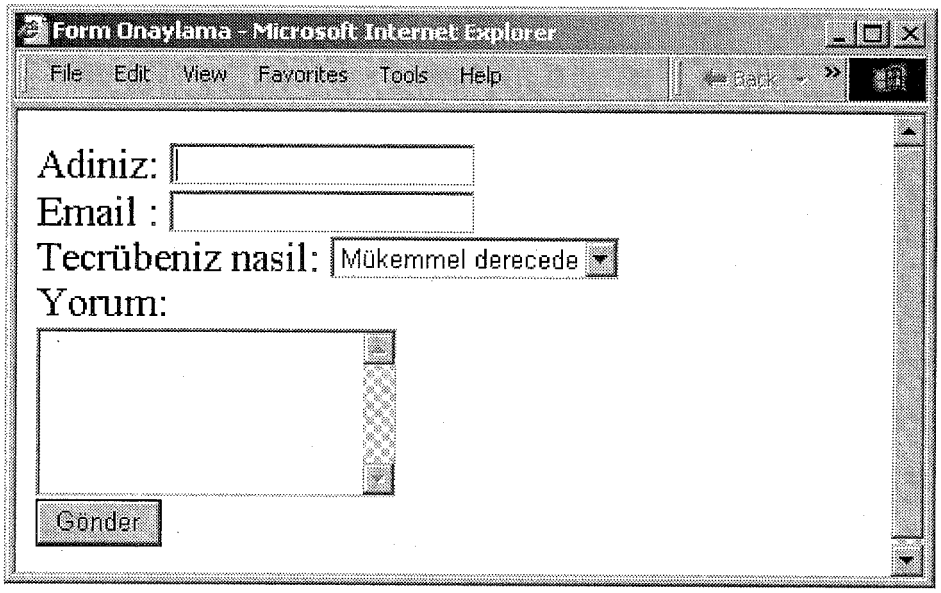
Formların Onaylanması

Form onaylama web sayfa tasarımcının bilmesi gereken bir şeydir. Size bilgilerin bir formdan nasıl alındığını ve form üzerindeki bilgilerin geçerli olup olmadığının nasıl onaylanacağını göstereceğim.

Öncelikle kullanıcıdan bilgi almak için form tasarımı yapıp gerekli ayarlamaları yapmalıyız. Kullanabileceğiniz birkçok değişik form elemanına sahipsiniz, fakat ben daha çok text box alanı, dropdown box elemanı, ve text area elamanları üzerinde duracağım. İşte ASP geliştiricilerden bilgi toplamak için küçük bir form.

```
<HTML>
<HEAD>
<TITLE>Form Onaylama</TITLE>
<BODY>
<FORM ACTION="onayla.asp" METHOD="POST">
Adınız: <INPUT TYPE="Text" NAME="adi"><BR>
Email : <INPUT TYPE="Text" NAME="email"><BR>
Tecrübeniz nasıl:
<SELECT NAME="tecrube" SIZE="1">
<OPTION VALUE="kotu">Kötü derecede</OPTION>
<OPTION VALUE="zayif">Zayıf derecede</OPTION>
<OPTION VALUE="orta">Orta derecede</OPTION>
<OPTION VALUE="iyi">İyi derecede</OPTION>
<OPTION VALUE="cokiyi">Çok iyi derecede</OPTION>
<OPTION VALUE="Mükemmel">Mükemmel derecede</OPTION>
</SELECT>
<BR>
Yorum:<BR>
<TEXTAREA NAME="yorum" COLS="20" ROWS="5"
WRAP="VIRTUAL"></TEXTAREA><BR>
<INPUT TYPE="Submit" VALUE="Gönder">
</FORM>
</BODY>
</HTML>
```

Hazırladığımız formun Internet Explorer tarayıcıdaki görünümü



Bu ileri yönde sunucuya gönderilmeli. Anlayacağınız gibi, o form üzerindeki bilgileri onayla.asp ye gönderir. Şimdi bizim bilgileri almamız ve sonra kullanıcının form üzerinde eksik bilgi girip girmediğini veya yanlış bir bilgi girişi varmı diye kontrol etmemiz gerekir.

Form üzerinden gönderilen ad ile ilgili bilgiyi almak için Request.Form("adi") komutunu kullanırız, ve diğer alanlar da aynı şekilde alınır. Form üzerinden bilgileri almak için onayla.asp içinde aşağıdaki kodların olması gerekir.

```
<%

formAdi = Request.Form("adi")

formEmail = Request.Form("email")

formTecrube= Request.Form("tecrube")

formYorum = Request.Form("yorum")

%>
```

Tüm bilgiler birkaç değişken içine alınır. Siz istediğiniz değişkenleri kullanabilirsiniz, fakat sizin onları isimlendirmeniz kendiniz için kullanımı daha kolay olacaktır. Aşağıda herbir değişkenin isimlendirmesini sizin için yaptım, ve sonra alanın adını program içinde kullandım.

Ve şimdi bilgilerimizi kontrol edeceğiz ve onu yapmak için birkaç fonksiyon elimizde var. Fakat önce formdan fazlalık olan boşlukları çekip almanın öneminden bahsedeceğim. Kullanıcı fazladan boşluklar eklemese bile, bazen değişik nedenlerden dolayı tarayıcı tarafından eklenir. Bu boşlukları çekip almak için Trim() fonksiyonunu kullanırız. Bilgi toplama scriptimizi tekrar yazalım.

```
<%

formAdi = Trim(Request.Form("adi"))

formEmail = Trim(Request.Form("email"))

formTecrube = Trim(Request.Form("tecrube"))

formYorum = Trim(Request.Form("yorum"))

%>
```

Şimdi herhangi bir alanın boş olup olmadığını kontrol edeceğiz. Len() fonksiyonunu bu işlemi yapmak için kullanabiliriz. Ben sık sık herbir alan için minumum bir değer ataması yaparım. Örneğin, bir email adresi çok nadir 8 den daha az karakter alır. Burada e-mail alan kısmında girilen değerin uzunluğunun 8 den büyük olup olmadığını kontrol ediliyor. Yorum alanı seçimlilik olduğundan dolayı uzunluğunu kontrol etmeyeceğiz. Uzunluğu kontrol etmek için bir IF THEN ifadesini kullanırız.

```
<%

IF len(formEmail)<8 THEN

 onayliForm = false

ELSE

 onayliForm = true

END IF

%>
```

Siz alanın özel bir karakter içerip içermediğini de bulabilirsiniz. Örneğin, bir email adres @ karakterini içermek zorunda. Bunun için InStr() fonksiyonunu kullanırız. Aranan string bulunamaz ise, 0 (sıfır) değeri döner. Bundan dolayı buna benzer bir şekilde bir email sınaması yapabiliriz.

```
<%

IF len(formEmail)<6 OR InStr(formEmail,"@")=0 THEN

 onayliForm = false
```

```
ELSE

 onayliForm = true

END IF

%>
```

Onun için kontrol edilecek son şey döndürülen değerin " veya ' içerip içermediğidir." sembolü bir VBScript stringin başlangıç ve sonunu belirler, bundan dolayı eğer birisi ad alanına Selahattin "Bilgisayar Mühendisi" Arslan yazarsa, VBScript string bilginin Selahattin den sonra sona erdiğini düşünecektir. Ve bir sonraki komut arandığında, o Bilgisayar Mühendisi string bilgisini bulacak ve sonra yeni stringin başlangıcı " Arslan olacaktır. Tahmin edebileceğiniz gibi, bu bir hata ile sonuçlanacaktır. ' işareti bir veritabanı stringin başlangıç ve sonunu belirler. Bundan dolayı bir bilgiyi bir veritabanına saklamak istediğinizde, siz tüm ' işaretleri bilgi içinden almalısınız. Bunu yapmanın yolu tüm " ve ' işaretlerini Replace() fonksiyonu ile çekip almaktır. Fakat onları ne ile alacaksınız? Çözüm çift sayıda " veya ' kullanmaktır. Siz bu şekilde kullanılan işaretlerin bir VBScript kod başlangıcı ve veritabanı string başlangıcı ve sonu olmadığını söyleyebiliriz, fakat doğal olarak stringin bir parçası olabilir. Bunu yapmanın yolu, söylediğim gibi, çift halde kullanmaktır. Aşağıdaki kodu kullanarak bir string içindeki " işaretlerini yerdeğiştirebilirsiniz.

```
<%

formEmail = Replace(formEmail,"""","""""")

%>
```

O bütün " işareti ile biraz garip görünebilir, fakat olaya daha iyi çalışırsanız, " işaretlerinin miktarının doğru olduğunu göreceksiniz. Siz bir hata almamak için bu işlemi mümkün olduğunca hemen yapmalısınız ve bunun önemini unutmamalısınız.

Şimdi onayla.asp için kodun son hali:

```
<%

formAdi = Trim(Replace(Request.Form("adi"),"""",""""""))

formEmail = Trim(Replace(Request.Form("email"),"""",""""""))

formTecrube = Trim(Replace(Request.Form("tecrube"),"""",""""""))

formYorum = Trim(Replace(Request.Form("yorum"),"""",""""""))

onayli_Form = true

IF len(formAdi)<2 THEN
```

```
 onayliForm = false
END IF
IF len(formEmail)<6 OR InStr(formEmail,"@")=0 THEN
 onayliForm = false
END IF
IF len(formTecrube)<3 THEN
 onayliForm = false
END IF
IF NOT onayliForm THEN
 %>
<HTML>
<BODY>
Hata. Tarayıcıya dönmek için tıkla, ve uygun bir şekilde doldur!
</HTML>
</BODY>
<%
ELSE
'Buraya veritabanına göndermek için bilgi yerleştirebilirsiniz. Eğer bunu
'yapacaksınız, mümkün olan ' işaretlerin alındığından emin olun.
%>
<HTML>
<BODY>
<B>Formu doldurduğunuz için teşekkürler <%=formAdi%>  !</B>
<BR><BR>
Aşağıdaki bilgileri gönderdiniz:<BR>
```

```
Adı: <I><%=formAdi%></I><BR>

Email: <I><%=formEmail%></I><BR>

Tecrübe: <I><%=formTecrube%></I><BR>

Yorum: <I><%=formYorum%></I>

</HTML>

</BODY>

<%

END IF

%>
```

Başka bir örnek,

İşte istemci-tarafı form onaylama için basit bir örnek. HTML kod bir text kutusu ve bir düğme içindir. Eğer aşağıdaki HTML kodlarını Internet Explorer kullanarak görmek isterseniz, küçük bir text kutusu ve onu izleyen bir düğme göreceksiniz.

```
<HTML>

<HEAD>

<TITLE>Form Onaylama</TITLE>

<SCRIPT LANGUAGE="VBScript">

<!--

Sub Onayla

Dim form1

Set form1 = Document.forms("gecerliForm")

If IsNumeric(form1.Text1.Value) Then

If form1.Text1.Value < 1 Or form1.Text1.Value > 20 Then

    MsgBox "Lütfen 1 ile 20 arasında bir değer girin."
Else

MsgBox "Teşekkürler."
```

```
End If

Else

   MsgBox "Lütfen sayısal bir değer girin."

End If

End Sub

-->

</SCRIPT>

</HEAD>

<BODY>

<H3>Basit bir form onaylama</H3><HR>

<FORM ID="gecerliForm" ACTION="sonuc.asp" onsubmit="Onayla(); return
false;" language="jscript">

1 ile 20 arasında geçerli bir değer girin:

<INPUT NAME="Text1" TYPE="TEXT" SIZE="2">

<INPUT NAME="Submit" TYPE="Submit" VALUE="Gönder">

</FORM>

</BODY>

</HTML>
```

Bu text kutusu ve basit bir VBScript sayfasındaki örnekler arasındaki fark text kutusunun value özelliği girilen değeri sınamak için kullanılır. Value özelliğini almak için, kod text kutusunun adına başvuruyu sınırlamaya sahip. Siz daima tam başvuruyu Document.gecerliForm.Text1 şeklinde yazabilirsiniz. Bunun yanında, çok sayıda form kontrol bileşenlerine başvuruya sahip olduğunuz yerde, siz burada yapılmak isteneni yapacaksınız. İlk önce bir değişken tanımla. Daha sonra Set yapısını form1 değişkenine formu atamak için kullanın. Bir düzenli atama yapısı, Dim gibi, burada çalışmaz; siz bir nesneye başvuruyu korumak için Set kullanmak zorundasınız.

Sayısal Değerlerin Kullanımı

Örneğin bir sayıya karşı gelen bir değeri doğrudan test ettiğini farketmelisiniz: o IsNumeric fonksiyonunu text içindeki bir stringin bir sayı olup olmadığından emin

olmak için kullanır. VBScript otomatik olarak string ve sayıları dönüştürmesine rağmen, o daima bir kullanıcının girdiği değeri onun alt tipi için test etmek ve gerektiği gibi fonksiyon dönüşümlerini kullanmak için iyi bir deneyimdir. Text kutu değerleri ile toplama yapıldığı zaman, değerleri sayılara dönüştürmek gerekir çünkü toplama işareti (+) operatörü hem toplama hemde string birleştirme işlemlerini yapar. Örneğin, eğer Text1 "1" ve Text2 "2" değerlerini içeriyorsa, siz aşağıdaki sonuçları görürsünüz:

```
S = Text1.Value + Text2.Value

' S "12" string değerini alır

S = CDbl(Text1.Value) + Text2.Value

' S 3 sayısal değerini alır
```

Form Bilgilerini Sunucuya Gönderme

Basit onaylama örneği bir düz düğme kontrol bileşenini kullanır. Eğer bir Submit kontrol bileşeni kullanılsaydı, örnek veriyi sınamak için asla göremezdi, ve herşey ani olarak sunucuya giderdi. Submit kontrol bileşeninden kaçınma size veriyi sınamanıza izin verir, fakat o veriyi sunucuya göndermez. O ek bir kod satırı daha gerektirir:

```
<SCRIPT LANGUAGE="VBScript">

<!--

Sub dugme1_OnClick

Dim form1

Set form1 = Document.gecerliForm

If IsNumeric(form1.Text1.Value) Then

  If form1.Text1.Value < 1 Or form1.Text1.Value > 10 Then

    MsgBox "Lütfen 1 ile 20 arasında bir değer girin."

  Else

    MsgBox "Teşekkürler."

    Form1.Submit   ' veri doğru; sunucuya gönder.

  End If
```

```
Else
  MsgBox "Lütfen sayısal bir değer girin."
End If
End Sub
-->
</SCRIPT>
```

Veriyi sunucuya göndermek için, kod bir veri doğru ise form nesnesindeki Submit metoduna başvurur.

Siz IsNumeric() fonksiyonunu (döndürülen değer true veya false olabilir) girilen bir değerin sayısal olup olmadığını kontrol etmek için kullanabilirsiniz. Eğer siz bilginizin özel bir durumda olmasını istiyorsanız, o zaman siz Ucase() ve Lcase() fonksiyonlarını kullanabilirsiniz. Bir alanın boş olduğunu kontrol etmek için IsNull() veya IsEmpty() fonksiyonlarını kullanabilirsiniz.

Eğer sunucuya gönderilen formların okumakta bir problem var ise, FORM etiketinin ENCTYPE özelliğini kaldırmalısınız.

Almaya çalıştığınız alan adı ile form üzerindeki alan adının aynı olduğundan yüzde yüz emin olmalısınız.

FORM etiketinin ACTION özelliğinin doğru sayfaya işaret ettiğinden emin olun. (ve METHOD özelliği için POST kullanın)

BÖLÜM
DOKUZ
9

VBScript Çevrim İfadeleri

- Çevrim Oluşturan İfadeler
- Do...Loop Çevrimi
- While...Wend Çevrimi
- For...Next Çevrimi
- For Each...Next Çevrimi

VBScript Çevrim İfadeleri

Çevrim Oluşturan İfadeler

Program kodlarını yazarken çok sık olarak kullandığınız bir işlem veya bir işlem dizisini tekrarlı olarak yapmak gereken durumlar ortaya çıktığı zaman çevrimler çok işinize yarayacak ve bu işi kolayca yapmanıza imkan verecektir. Birkaç çevrim, koşul False olana kadar ifadeleri tekrar eder; başka ifadeler de ise bir koşul True olana kadar çevrim devam eder. Belirli sayıda ifadeleri tekrarlı şekilde yürütülmesini sağlayan çevrimlerde vardır.

VBScript içindeki mevcut çevrim ifadeleri aşağıdadır:

- **Do...Loop**

Bir koşul True iken veya True olana kadar çevrim yapar.

- **While...Wend**

Bir koşul True ise çevrim yapar. Burada koşul False olduğu zaman çevrim sona erer.

- **For...Next**

Belirtilen sayıda ifadeleri yürütmek için bir sayaç kullanır. Bu sayaç değeri son değere ulaşana kadar çevrim devam eder.

- **For Each...Next**

Bir koleksiyondaki her bir madde veya bir dizinin her bir elemanı için bir grup ifadeleri tekrarlar. Bu ifadeler nesneler üzerinde ardışık işlemler yapmak için kullanılır.

Do...Loop Çevrimi

İlk çevrim olarak ele alacağımız Do...Loop çevrimi bir blok ifadenin çok defa çalıştırılması için kullanabilirsiniz. İfadeler bir koşul ya True iken yada koşul True olana kadar tekrarlanır.

Koşul True iken yapıların tekrarlanması

While anahtar sözcüğünü Do..Loop yapısında koşulun sağlanıp sağlanmadığını sınamak için kullanırız. Çevrim içine girmeden önce koşulun doğruluğunu sınayabiliriz, veya çevrim en az bir kez yürütüldükten sonra koşulun doğruluğunu sınayabiliriz. Örnekte

eğer sayi değeri 0 yerine 10 olarak ayarlanırsa, iç kısımdaki ifadeler çevrimi asla çalışmayacaktır. Sayi değeri 9 olsa iç kısımdaki ifadeler çevrimi sadece bir kez çalışacaktır çünkü bir sonraki koşul sınaması sonucu False değerine sahip olacaktır.

```
<HTML>

<BODY>

<SCRIPT TYPE="text/vbscript">

sayi =0

Do While sayi < 10

Document.Write(sayi & "<br>")

sayi=sayi+1

Loop

MsgBox "Çevrim " & sayi & " kez yapıldı."

</SCRIPT>

</BODY>

</HTML>
```

Başka bir örnek, bu örnekte koşul sağlanmasa bile en az bir kez çevrim içindeki ifadeler yürütülür. Burada 5 kez çevrim olacaktır, sayi en son değer olarak 5 değerini alacak ve çevrim sona erecektir.

```
<%

sayi = 1

Do

sayi = sayi + 1

Loop While sayi < 5

%>
```

Başka bir örnek, bu örnekte koşul sağlanmaz ise çevrim içindeki ifadeler hiç yürütülmeden çevrim sona erer. Burada sayi değeri ilk olarak 1 alınış olup 4 kez çevrim içindeki ifadeleri yürütecektir.

```
<%

sayi = 1

Do While sayi < 5

sayi = sayi + 1

Loop

%>
```

Koşul True olana kadar bir yapıyı tekrarlama

Bir Do...Loop yapısı içinde bir koşulun doğruluğunu sınamak için Until anahtar sözcüğünün kullanılması için iki yöntem vardır. Siz çevrime girmeden önce (aşağıda kontrol1 örneğinde gösterildiği gibi) koşulun doğruluğunu sınayabilirsiniz, veya çevrim en az bir kez yürütüldükten (aşağıda kontrol2 örneğinde gösterildiği gibi) sonra koşulun doğruluğunu sınayabiliriz. Koşul False olduğu sürece , çevrim meydana gelecektir.

```
Sub kontrol1()

Dim sayici, numara

sayici = 0

numara = 20

Do Until numara = 10

numara = numara - 1

sayici = sayici + 1

Loop

MsgBox "Çevrim " & sayici & " kez yapıldı."

End Sub


Sub kontrol2()

Dim sayici, numara

sayici = 0
```

```
numara = 1

Do

numara = numara + 1

sayici = sayici + 1

Loop Until numara = 10

MsgBox "Çevrim " & sayici & " kez yapıldı."

End Sub
```

Başka bir örnek, burada sayi değeri 5 olana kadar çevrim yapılacak ve 5 olduğunda çevrim sona erecektir.

```
<%

sayi = 1

Do

sayi = sayi + 1

Loop Until sayi = 5

%>
```

Do...Loop yapısı içinde iken çevrimden çıkma

Siz bir Do...Loop çevrimi içinden Exit Do yapısını kullanarak çıkabilirsiniz. Çünkü siz genellikle olması gereken bir durum, sonsuz çevrimi istememeniz gibi, meydana geldiğinde çıkmak istersiniz. Siz bir If...Then...Else yapısının True ifade bloğunda Exit Do ifadesini kullanmalısınız. Eğer koşul False ise , çevrim alışıldığı şekilde çalışmaya devam eder.

Aşağıdaki örnekte, numara değişkenine sonsuz bir çevrim oluşturacak bir değer ataması yapıldı. If...Then...Else yapısı sonsuz tekrarlamayı engellemek için bu koşulun doğruluğunu sınar.

```
Sub cevrimcikis()

Dim sayici, numara

sayici = 0

numara = 9
```

```
Do Until numara = 10
    numara = numara - 1
    sayici = sayici + 1
    If numara < 10 Then Exit Do
Loop
MsgBox "Çevrim " & sayici & " kez yapıldı."
End Sub
```

While...Wend Çevrimi

While...Wend çevrimi VBScript içinde desteklenir fakat While...Wend yapısındaki esneklik fazla olmadığından dolayı genellikle bu çevrim yerine Do...Loop çevrimi kullanılır.

```
While < koşul >
    ' VBScript ifadeleri
Wend
```

Örneğin:

```
<SCRIPT LANGUAGE="VBScript">
Dim x
For x = 1 to 10
    Document.Write x & '<BR>'
Next
x=0
While x <= 10
    xi = x + 1
    Document.Write x & '<BR>'
Wend
```

```
</SCRIPT>
```

Yukarıdaki her iki çevrimde x değişkeni 10 olana kadar yürütülür. Tüm For...Next çevrimleri While çevrimi ile aynıdır fakat bunun tersi doğru değildir.

Örneğin, siz aşağıdaki çevrimi bir For...Next çevrimi olarak yazamazsınız.

```
<SCRIPT LANGUAGE="VBScript">
Dim s
s="a"
While s<>"abcd"
        s = s & Chr ( Asc (Right( s, 1 )) + 1 )
        Document.Write  s & "<BR>"
Wend
</SCRIPT>
```

Başka örnekler

```
<%
While deneme = 250
        Rem çevrim içine istediğiniz kodu yazabilirsiniz.
WEnd
%>
```

Bu örnekte zamanın saat kısmı 8 den küçük olduğu sürece çevrim devam eder, 8 e eşit veya büyük olduğunda çevrim sona erer.

```
<%
While Hour(Time) < 8
        Rem çevrim içine istediğiniz kodu yazabilirsiniz.
WEnd
%>
```

For...Next Çevrimi

For...Next ifadelerini belirli sayıda bir blok ifadenin yürütülmesi için kullanabilirsiniz. Çevrimler için, her bir çevrimde bir artan veya azalan bir sayaç değişkeni kullanmanız gerekir.

Aşağıdaki örnekte For...Next çevrimi beş kez yürütülür.

```
<SCRIPT LANGUAGE="VBScript">

Dim i

Dim sayac

sayac = 0

For i = 1  to  5

    sayac = sayac + 1

Next

Document.Write sayac

' sayac değeri 5 olacaktır

</SCRIPT>
```

Aşağıdaki örnek programda yaz olarak adlandırılan prosedürün 10 kez yürütülmesine neden olur. For yapısı x sayaç değişkenini ve onun başlangıç ve son değerlerini belirler. Next yapısı sayaç değişkenini bir artırır.

```
. . .

Sub yaz( y )

        document.write y & "<BR>"

End Sub
. . .

Sub prosedur()

Dim x

For x = 1 To 10

      yaz( x )
```

```
Next

End Sub
...
```

Step anahtar sözcüğünü kullanarak belirlediğiniz bir değere göre sayaç değişkenini artırır veya azaltabilirsiniz. Aşağıdaki örnekte, x sayaç değişkeni her bir çevrim tekrarında 3 artırılır. Çevrim tamamlandığı zaman, x in son değeri 15 olacaktır.

```
Sub adim()

Dim x

For x = 3 To 15 Step 3

    Document.Write  x  & "<BR>"

Next

    MsgBox "x in son değeri : " & x

End Sub
```

Sayaç değişkenini azaltmak için negatif bir Step değeri kullanmalısınız. Başlangıç değerinden daha az olan bir son değer belirtmek zorundasınız. Aşağıdaki örnekte, tersi sayaç değişkeni her bir çevrim tekrarında 3 azaltılır. Çevrim tamamlandığı zaman, sayacın alacağı en son değer 9 olacaktır.

```
Sub gerisayim()

Dim tersi

For tersi = 21 To 9 Step -3

    Document.write  tersi

Next

    MsgBox "tersi değişkenin son değeri " & tersi

End Sub
```

Exit For Kullanımı : Herhangi bir For...Next yapısından kullandığınız sayaç değişkeninin değeri son değerine ulaşmadan Exit For yapısını kullanarak çıkabilirsiniz. Çünkü genellikle istenen bir duruma ulaşıldıysa veya bir hata meydana geldiyse o zaman çevrim sonlandırılır.

For Each...Next Çevrimi

Bir For Each...Next çevrimi bir For...Next çevrimine benzer. Belirli bir sayıda ifadelerin tekrarlanması yerine, bir For Each...Next çevrimi bir nesnelerin toplamındaki her bir maddeyi veya bir dizinin herbir elemanı için bir grup ifadeyi tekrar eder. Bu özellikle eğer bir koleksiyonda kaçtane eleman olduğunu bilmiyorsanız faydalı olur.

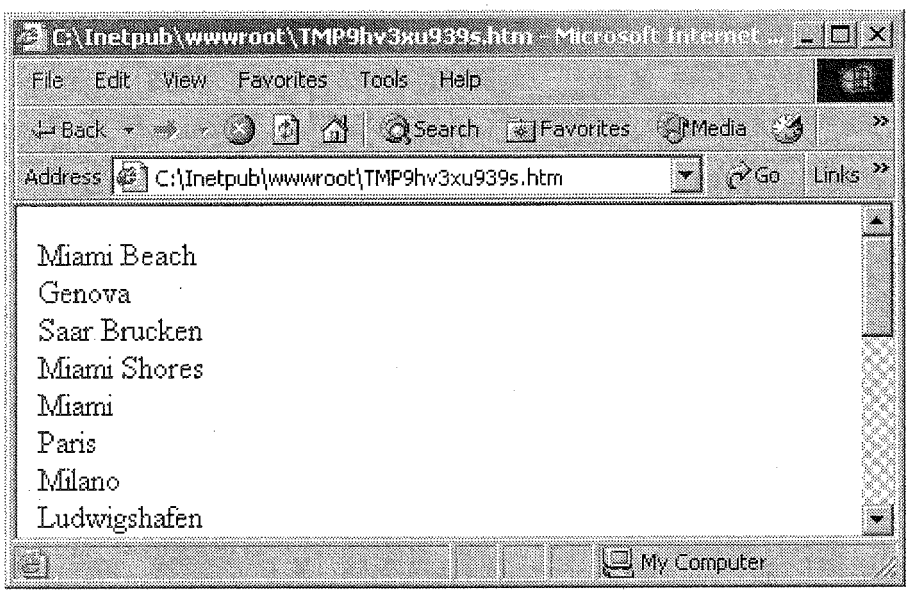
Aşağıdaki HTML kod örneğinde, Dictionary nesnesinin içerikleri document.write metodu ile doküman üzerine yazdırmak için kullanılıyor. **sozluk** değişkeni bir Dictionary nesne değişkenidir. Bu nesne değişkenine her bir madde eklenirken aynı zamanda her bir madde için bir anahtar değeri atanmaktadır. Bu nesne içindeki değerlere anahtar değerleri kullanılarak erişilecektir.

```
<HTML>
<HEAD>
<TITLE>Formlar ve Elemanlar</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub degistir_OnClick
  Dim sozluk
  Set sozluk = CreateObject("Scripting.Dictionary")
  sozluk.Add "0", "Miami Beach"
  sozluk.Add "1", "Genova"
  sozluk.Add "2", "Saar Brucken"
  sozluk.Add "3", "Miami Shores"
  sozluk.Add "4", "Miami"
  sozluk.Add "5", "Paris"
  sozluk.Add "6", "Milano"
  sozluk.Add "7", "Ludwigshafen"
  sozluk.Add "8", "Orlando"
  sozluk.Add "9", "Newyork"
```

```
  sozluk.Add "10","Monaco"
  sozluk.Add "11","Marsilya"
  sozluk.Add "12","Toluon"
  sozluk.Add "13","Nice"
  sozluk.Add "14","Cannes"
  sozluk.Add "15","Reims"
  sozluk.Add "16","Monte Carlo"
  sozluk.Add "17","Metz"
  sozluk.Add "18","Frankfurth Main"
  sozluk.Add "19","San Marino"
  sozluk.Add "20","Florida"
  sozluk.Add "21","New Jersey"
For Each i in sozluk
  Document.write  sozluk.Item(i) & "<BR>"
Next
End Sub
// -->
</SCRIPT>
</HEAD>
<BODY>
<CENTER>
<FORM NAME="form1">
<INPUT TYPE ="Button" NAME="listele" VALUE="Listele"><p>
</FORM>
</CENTER>
```

```
</BODY>

</HTML>
```

Listele düğmesine fare ile tıkladığınızda bir sonraki şekli göreceksiniz.



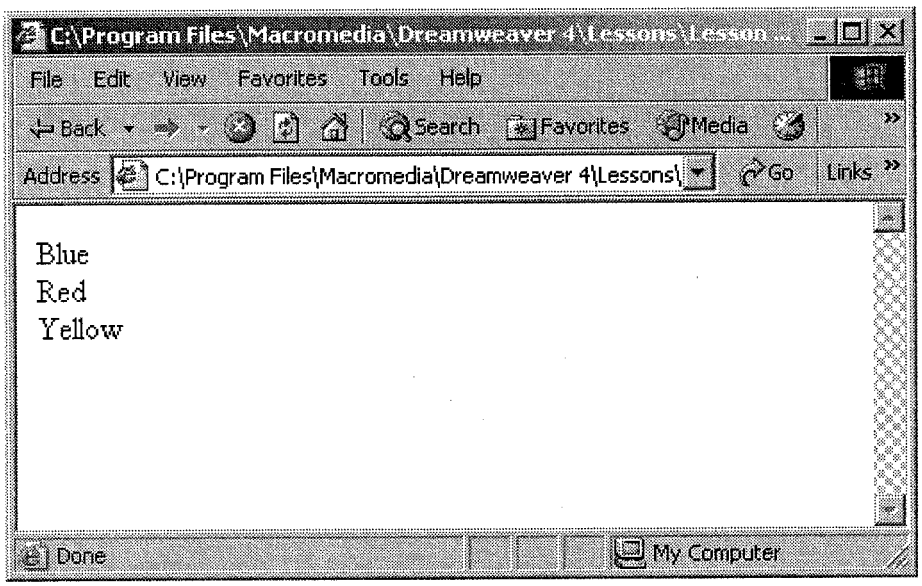
Birkaç değişik örnek daha verelim konunun iyi anlaşılması açısından,

Örnekte değişken sizin vereceğiniz bir dizi içindeki tüm elemanları gösterir, herhangi bir anda, ilk elemandan son elemana kadar erişmek için kullanılır.

```
<%
For Each sayac In dizi
  Rem buraya çevrim içinde kullanacağınız kodları yazabilirsiniz.
Next
%>
```

Örnek

```
<HTML>
<BODY>
<SCRIPT TYPE="text/vbscript">
Dim renkler(2)
renkler(0) = "Blue"
renkler(1) = "Red"
renkler(2) = "Yellow"

For Each i in renkler
  Document.write(i & "<br>")
Next
</SCRIPT>
</BODY>
</HTML>
```


C:\Program Files\Macromedia\Dreamweaver 4\Lessons\Lesson ...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Program Files\Macromedia\Dreamweaver 4\Lessons\
Go Links
Blue
Red
Yellow
Done
My Computer

BÖLÜM ON 10

VBScript Prosedürleri

- VBScript Prosedürleri
- Sub Prosedürü
- Function Prosedürü
- Prosedürlere Veri Aktarma ve Veri Alma
- Sub ve Function Prosedürlerinin Kullanımı
- VBScript İç Fonksiyonları

VBScript Prosedürleri

VBScript size bir ad verilerek kod blokları oluşturmanıza imkan sağlar, genellikle bunlar metodlar veya prosedürler olarak adlandırılır. Birçok dil, fonksiyon gibi yanlızca tek tip bir prosedüre sahiptir. VBScript içinde, iki çeşit prosedür vardır; Sub prosedürü ve Function prosedürü. Sub prosedürleri değer döndürmeyen prosedürler. Fonksiyonlar ise bir değer döndürür. Çoğu durumlarda, siz prosedür olarak ya bir fonksiyon yada bir sub prosedür yazabilirsiniz.

VBScript içinde bir prosedüre sık sık aynı şeyleri yaptırıyorsak , o zaman bir sub prosedürü kullanırız, ama bir değer döndürmek gerekiyorsa o zaman bir fonksiyon prosedürü kullanırız. Her iki prosedüre çağrı yaptığımız komut satırından parametreler yardımı ile veri aktarabiliriz veya bu prosedürlerden veri alabiliriz. Sub prosedürden verinin alınması ancak parametreler ile olur, ancak fonksiyon prosedürlerden veri döndürme işlemi fonksiyon adına döndürmek istediğimiz değeri atayarak alırız.

Sub ve Function prosedürleri tanımları yaptığımızda ilgili kodları girdikten sonra bu prosedürlerin sonunun geldiğini belirtmek için End ile birlikte sub ve function isimlerini kullanırız, bunlar End Sub ve End Function dır.

Bir Sub prosedür örneği,

```
Sub carpma(sayi1, sayi2, sonuc)
        sonuc = sayi1 * sayi2
End Sub
```

Sub prosedür *sayi1* ve *sayi2* iki argümanlarını çarpar ve sonucu *sonuc* argüman değişkenine aktarır. Bu işlem bir fonksiyon tanımlayarak daha kullanışlı hale getirebiliriz.

Bir Function prosedür örneği,

```
Function carpma(sayi1, sayi2)
        carpma = sayi1 * sayi2
End Sub
```

Fonksiyon sonucu yine bir sayısal değer olup fonksiyon adına atanmıştır. Dolaysıyla sonuca ulaşmak için fonksiyon adını kullanırız. Çoğu programlama dillerinde fonksiyondan bir değer döndürmek için *return* yapısı kullanılır. VBScript fonksiyonun adı ile aynı olan bir değişken oluşturur, dolaysıyla fonksiyonun sonucunu fonksiyon adına atayarak değeri döndürebiliriz.

Sub prosedürlerini ve fonksiyon prosedürlerini çağırma şekli farklıdır. *carpma* sub prosedürünü çağırmak için aşağıdaki yazım şekillerinden birini kullanırız:

```
carpma  sayi1, sayi2, sonuc
```

veya

```
Call carpma(sayi1, sayi2, sonuc)
```

Bir fonksiyonu çağırmak için, fonksiyon sonucunu bir değişkene atamalısınız.

Örneğin:

```
yenisonuc = carpma(sayi1, sayi2)
```

Sub prosedürler ve fonksiyon prosedürler oluşturulurken argüman tanımları yapılır. Bu durumda bu prosedürlere çağrı yapılırken argümanlar arasında uyum olması gerekir. Prosedür çağırmalarında tip uyumluluğu olması gerekmez; zaten böyle bir durumda VBScript tipler arasında uygun bir dönüşüm yapacaktır fakat bu tiplerin bizim yapmak istediğimiz şeyi karşılamalı. Eğer tipler arasında bir uyumsuzluk olursa, beklenmeyen sonuçlar alabilirsiniz.

Örneğin:

```
<%

Function carpma(sayi1, sayi2)

carpma = sayi1 *  sayi2

End Function

%>


<%

' Çarpma fonksiyonuna çağrı yapılması
```

```
Dim st

s1 = carpma( 7, 8)        ' Sonuç olarak  56 döndürülecektir
s2 = carpma("7", "8" )    ' 56 sonuç olarak döndürülür

%>
```

Yukarıdaki her iki örnekte aynı şeyi gösterir. İkinci örnek bir string değer döndürmek yerine bir sayısal değer döndürür. *carpma* fonksiyonu muhtemelen bir hataya yol açmadan karakterleri sayı olarak değerlendirip çarpma işlemini ilgili iki sayı üzerinde yapacaktır.

Değişken yazma problemleri VBScript içinde heryerdeki gibidir çünkü VBScript içinde tanımlanan değişken tipli bir değişkene sahip değildir. Bu yüzden, fonksiyonları yazarken, özellikle başka şeylerde kullanmak için fonksiyonlar yazıyorsanız, giriş tiplerini kontrol etmelisiniz. Uygun yerlerde tip değiştirmek için CInt, CBool gibi operatörleri kullanın, ve uygun olmayan yerlerde bir hata ile karşılaşabilirsiniz.

Örneğin:

```
Function  stEkle ( st1, st2 )

If varType( str1 ) <> vbString Then

  st1 = CStr ( st1)

End If


If varType( st2 ) <> vbString Then

  st2 = CStr ( st2 )

End If

strEkle = st1  &  st2

End Function


' fonksiyonun çağrılması

Dim st

st = stringEkle( 5, 9)        ' Sonuç olarak  59 döndürülecektir

st = stringEkle( "5", "9" )   ' "59" sonuç olarak döndürülür
```

Önceki örnekte her iki giriş argümanda tip kontrolu yapar ve eğer argüman tipleri string değilse onları string değerlere çevirir. Burada tip kontrolu yapmak çok zaman almayacaktır, fakat hızdaki ufak bir kayıp bir hataya neden olmaktan daha iyidir.

İzleyen örnek VBScript kullanılarak yazılmış olan ASP programının içinde bir prosedürün rekürsif (programın kendi içinde kendini çağırması) olarak çağrılmasını gösterir.

```
' program adı : rekürsif.asp

<%@ Language=VBScript %>

<% Option Explicit %>

<%

Sub buyukDizi(dizi, ilk, son)

  If ilk <= son Then

    dizi(ilk) = UCase(dizi(ilk))

    ilk = ilk+ 1

    Call buyukDizi(dizi, ilk, son)

  End if

End sub


Dim i

Dim ilk

Dim son

Dim dizi

dizi = Array("istanbul", "ankara", "siv as")

Response.Write "Orijinal dizi:<BR>"

For i = LBound(dizi) To UBound(dizi)

  Response.Write dizi(i) & "<BR>"

Next
```

```
Response.Write "<P>"

Response.Write " Tüm elemanların harflerini büyük yap<BR>"

Call buyukDizi(dizi, LBound(dizi), UBound(dizi))

For i = LBound(dizi) To UBound(dizi)

  Response.Write dizi(i) & "<BR>"

Fext

Response.Write "<P>"

Response.Write " 1. pozisyonundaki elemanın harflerini büyük yap<BR>"
dizi = Array("istanbul", "ankara", "sivas")

Call buyukDizi(dizi, 1, 1)

For i = LBound(dizi) To UBound(dizi)

  Response.Write dizi(i) & "<BR>"

Next

%>
```

ASP programları yazarken VBScript kodlarını <% ve %> arasına almak zorundayız.

Sub Prosedürü

Bir Sub prosedürü Sub ve End Sub yapıları içine alınmış aksiyonları yerine getiren ve bir değer döndürmeyen bir dizi VBScript yapılarıdır. Bir Sub prosedürü argümanlar alabilir (bir çağıran prosedür tarafından geçilen sabitler, değişkenler, veya ifadeler). Eğer bir Sub prosedürü argümanı sahip değilse, onun Sub yapısı boş bir parantez kümesi ( ) içermek zorundadır.

İzleyen Sub prosedürü iki asıl , veya iç VBScript fonksiyonları olan MsgBox ve InputBox'ın bir kullanıcının bilgi girişini sağlamak için kullanır. Sonra o bilgi üzerinden bir hesaplama sonuçlarını gösterir. Hesaplama VBScript kullanımıyla oluşturulan bir Function prosedüründe yapılır. Function prosedürü aşağıdaki gibi gösterilir.

```
<% INPUT TYPE="button" NAME="dugme" VALUE="Buraya Tıkla!" %>
```

```
< SCRIPT LANGUAGE="VBScript" >

Sub dugme_onClick

  Dim bilgi

  bilgi = InputBox("Bu bir giriş kutusudur!")

End Sub

< /SCRIPT >
```

Function Prosedürü

Bir Function prosedürü Function ve End Function yapıları içine alınmış aksiyonları yerine getiren bir dizi VBScript yapılarıdır. Bir Function prosedürü bir Sub prosedürüne benzer, fakat bir değerde döndürür. Bir Function prosedürü argümanlar alabilir (bir çağıran prosedür tarafından geçilen sabitler, değişkenler, veya ifadeler). Eğer bir Function prosedürü argümana sahip değilse, onun Function yapısı boş bir parantez kümesi ( ) içermek zorundadır. Bir Function bir veya daha çok prosedür yapısında onun adına atanan bir değer döndürür. Bir Function dönüş tipi daima bir Variant tipindedir.

Aşağıdaki örnek negatif olmayan geçerli bir sayının faktoriyel hesabını yapar.

```
<SCRIPT LANGUAGE = "VBScript">

Function faktoriyel(ByVal tamsayi)

If tamsayi <= 1 Then

  faktoriyel= 1

Else

  faktoriyel = tamsayi * faktoriyel(tamsayi - 1)

End If

End Function

</SCRIPT>
```

ve fonksiyon çağrısı aşağıdaki gibi yapılır.

```
<HTML>

<HEAD>
```

```
<TITLE>Faktoriyel hesabı</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=">
<SCRIPT LANGUAGE = "VBScript">
Function faktoriyel(ByVal tamsayi)
If tamsayi <= 1 Then
        faktoriyel= 1
Else
        faktoriyel = tamsayi * faktoriyel(tamsayi - 1)
End If
End Function
</SCRIPT>
</HEAD>

<BODY BGCOLOR="#FFFFFF"  TEXT="#000000">
<SCRIPT>
Document.Write ("8 in faktoriyeli " & faktoriyel(8))
Document.Write ("<BR>")
Document.Write ("3 ün faktoriyeli " & faktoriyel(3))
</SCRIPT>
</BODY>
</HTML>
```

Prosedürlere Veri Aktarma ve Veri Alma

Verinin herbir parçası bir argüman kullanılarak sizin prosedürlerinize geçilir. Argümanlar prosedürünüze geçmek istediğiniz veri için bir yer tutucu olarak hizmet eder. Siz argümanlarınıza herhangi bir geçerli değişken adı verebilirsiniz. Sub yapısı yada Function yapısını kullanarak bir prosedür oluşturmak istediğinizde, parantezler

prosedürün adından sonra dahil edilmek zorundadır. Bu parantezlerin içine yerleştirilen argümanlar, sayi1, ve sayi2 için Toplama fonksiyonuna geçilen değer için bir yer tutucudur.

```
<%
Function Toplama(sayi1, sayi2)
  Toplama = sayi1 + sayi2
End Function
%>
```

Bir prosedürün dışına bir veriyi çıkarmak için, siz bir Function kullanmak zorundasınız, bir Function prosedürü bir değer döndürebilir; bir Sub prosedürü döndüremez.

Aşağıdaki örnek derecenin radyana nasıl çevrildiğini gösterir. Şu an elimizde 2 tane fonksiyon var, PI sabitinin değerini 3.14 olarak göstermek için bu örnekte, tabiki bu sabit için ondalık kısmı daha çok haneli olabilir. Bu örnekte hem fonksiyona parametre ile veri aktarımı yapılıyor hemde aynı fonksiyondan veri alınmakta.

```
<SCRIPT LANGUAGE = "VBScript">
' dereceyi radyana çevir
Function Radyan(derece)
        Radyan = derece * 3.14 / 180
End Function
' radyanı dereceye çevir.
Function Derece(radyan)
        Derece = radyan * 180 / 3.14
End Function
</SCRIPT>
```

Bu fonksiyonu test etmek için işte kod.

```
<SCRIPT LANGUAGE = "VBScript">
' 90 dereceyi radyana çevir
```

```
Document.Write Radyan(90)

Document.Write ("<BR>")

' 0.78 radyanı dereceye çevir

Document.Write Derece(0.78)

</SCRIPT>
```

Çıktılar:

1.57

44.7133757961783

Sub ve Function Prosedürlerinin Kullanımı

Kodunuzdaki bir Function daima bir değişkenin sağ tarafında veya bir ifade içinde kullanılır.

Örneğin:

```
Miktar = Toplam( 10, 20 )
```

veya

```
MsgBox "Miktar " & Toplam( 10, 20) & " kadardır."
```

Bir başka prosedürden bir Sub prosedürü çağırmak için, bir virgül ile herbiri ayrılmış gerekli herhangi argümanlar ile prosedürün adı yazılmalı. Call yapısı gerekli değil. fakat eğer siz bunu kullanırsanız, siz argümanları parantez içine almak zorundasınız.

Aşağıdaki örnek *giris* prosedürüne iki çağrı yapar. biri kod içindeki Call yapısını kullanır; diğeri kullanmaz. Herikiside tam olarak aynı şeyi yapar.

```
Call giris(kullanıcı, sifre)

giris kullanıcı, sifre
```

Call yapısı kullanılmadığında parantezlerin çağrı içinden çekilip alındığına dikkat ediniz.

VBScript İç Fonksiyonları

İç fonksiyonların hepsi zaten liste halinde verildi, fakat siz muhtemelen onlardan birkaçını kullanımının nasıl olduğunu görme ihtiyacı duyacaksınız, özellikle tarih ve zaman fonksiyonları gibi.

Bunların en basiti Now() fonksiyonudur. Now o anki güncel tarih ve zamanı bir Date değişkeni içinde döndürür.

Örneğin:

```
Response.Write Now()  'Güncel tarih ve zamanı yazar
```

Now() fonksiyonuna yakın bir fonksiyon olan Date() zaman bilgisi olmaksızın sadece tarih bilgisini döndürür.

"Bugünden İki hafta sonrası" gibi bilinen bir tarihten bir tarih tabanı (ofset) hesaplamak için ihtiyaç duyduğunuzda, DateAdd fonksiyonunu kullanın. Siz başlangıç tarihini , aralık, ve ofsetin büyüklüğünü sağlarsınız.

Örneğin:

```
' Bugünden İki hafta sonrası

Response.Write DateAdd('ww', 2 , Now() )
```

Siz ofset süresini bir string olarak belirtirsiniz.

Geçerli ofset değerler:

Ofsetaralık değeri	Anlamı
s	Saniye
n	Dakika
h	Saat
D	Gün
w	Haftanın günü
Ww	Yılın haftası
y	Yılın günü
m	Ay
q	Dörtte bir
yyyy	Yıl

Bilinen bir tarihten daha önceki bir tarihi hesaplamak için, negatif bir ofset değeri kullanın. Örneğin, aşağıdaki kod parçası tam olarak bir yıl öncesinin nasıl hesaplanacağını gösterir:

```
Response.write DateAdd( "yyyy" , Now() , -1 )
```

Bununla ilgi diğer bir fonksiyon ise bilinen iki tarih arasındaki aralığı hesaplayan DateDiff fonksiyonudur. Örneğin, Ocak 1, 1900 dan bu yana günlerin sayısını bulmak için aşağıdaki gibi bir kod yazarız:

```
Response.write DateDiff("d",#1/1/1900#, Now() )

Response.write DateDiff("d","1/1/1900", Now() )
```

DateDiff fonksiyonu daima ilk tarihi ikinci tarihten çıkarır; bu yüzden eğer siz iki tarihin yerlerini değiştirirseniz, fonksiyon negatif bir sayı döndürür.

Örneğin:

```
Response.write  DateDiff ( "d" , Now() , #1/1/1900#)
```

FormatDateTime fonksiyonu tarihleri bilinen formatlarda göstermemizi sağlar. Siz tarih ve bir formatlama (tarihin nasıl gösterilmesi gerektiğini gösteren bir sabit değer) şeklini argüman olarak vermelisiniz.

Örneğin:

```
Response.write FormatDateTime( Now ,vbGeneralDate)
```

VBScript içinde tanımlı iç tarih ve zaman gösterme formatları yanlızca beş tanedir. Aşağıdaki tablo sabitleri ve Ocak 20, 2002 tarihi için örnek bir sonuç gösterir.

Sabit	Sonuç
vbGeneralDate	01/20/2002 04:22:01 PM
vbLongDate	Çarşamba, Ocak 20, 2002
vbShortDate	01/20/02
vbLongTime	04:22:01 PM
vbShortTime	04:22

Bunlar size VBScript dilinin tarih ve zaman bilgilerini içinde nasıl sakladığı hakkında yardımcı olur. Bir VBScript DateTime (Date alt tipi) değişkeni bir Double tipindedir

(8-byte uzunluğunda bir değer) . Tamsayı kısmı tarihi gösterir ve kalan kısım zamanı gösterir. Bu yüzden siz yanlızca tarih bilgisini almak istediğinizde DateTime değişkenini tarih bilgisini kesip (truncate) alabilirsiniz. Buna ek olarak, eğer bir DateTime değişkeni üzerinde toplama veya çıkarma yapmak isterseniz, günleri değiştirebilirsiniz. Örneğin, eğer güncel tarihe 1 eklemek isterseniz:

```
Response.write Now() + 1
```

VBScript aynı anda bir sonraki güne ait tarihi döndürür. Benzer olarak, şöyle yazabilirsiniz:

```
Response.write FormatDateTime( CLng( Now() ),vbShortDate)
```

Değeri VBScriptin bir tarih olarak yorumlaması için FormatDateTime fonksiyonunu kullanarak bir Long dönüşüm sonucunu formatlamalısınız. Buna alternatif olarak, CDate fonksiyonunu kullanarak Long dönüşümünün sonucunu bir tarihe dönüştürebilirdiniz:

```
Response.write CDate(CLng ( Now() ) + 1 )
```

Sayıları formatlama VBScriptte tarihleri ve zamanları formatlamaya çok benzer. FormatNumber, FormatPercent, ve FormatCurrency fonksiyonları alışkın olduğumuz formatlar ile çalışmamızı sağlar. Bunların hepsi birbirine benzer olarak çalışırlar. Formatlamak için değer veya ifade sağlamak zorundasınız. Siz onlu noktadan sonra gösterilmesini istediğiniz dijitlerin sayısını , negatif değerlerin etrafında parantez kullanmak isterseniz, ve sayıların grupları arasına virgül (veya bilgisayarınızın bölgesel ayarlarına bağlı olarak başka karakterler), koymak istediğinizde, bunların hepsini verebilirsiniz.

Yazım şekli:

```
FormatCurrency(Ifade ,OnluYerler, _AnaHaneEkle,
                    ParantezİcindekiNegatifSayilar, _AyiricilarEkle)
```

En son üç seçimlilik argümanların hepsi Tristate sabitler olarak bilinen sabitlerin bir gruptaki değerlerden birini alırlar.

Sabitler	Value	Tanımı
TristateTrue	- 1	True
TristateFalse	0	False
TristateUseDefault	- 2	bilgisayarınızın bölgesel ayarlarını kullanın

Örneğin, 1009.7386 değerini currency olarak formatlamak için, en yakın penny e yuvarlandı.

```
<%
Dim s
s = FormatCurrency( 1009.7386, 2, True, True, True )
Response.write s
' $1.009.74 olarak bir çıktı olur
%>
```

Aynı kodlar -.0246 değerine uygulanırsa sonuç olarak $0.02 sonucunu alırız. Sonuç parantezler arasındadır çünkü sonuç negatif bir sayı ve fonksiyon çağrısı *ParantezİcindekiNegatifSayilar* argümanını True olarak ayarlar.

BÖLÜM 11
ON BİR

VBScript Hataları Kotarma

- Hata Kotarma
- Çalışma Zamanı ve Yazım Hata Kodları

VBScript Hataları Kotarma

Hata Kotarma

Herhangi bir dilde program yazarken veya yazdığımız program kodlarını çalıştırırken hatalar meydana gelebilir. Durum böyle olunca VBScript ile yazdığımız script kodlarımızın da bir hataya neden olması gayet doğal olacaktır. Önemli olan bizim olabilecek bu hatalar meydana geldiğinde ne yapmamız gerektiğini bilmek ona göre hata kotarıcı yani hatayı sezmek ve kullanıcıya mümkün olduğu sürece yansıtmadan hatayı gidermektir. Hataya neden olabilecek çok şey vardır önemli olan bunları önceden tahmin ederek kullanıcının bu hatayı yapmasını engellemek lazım, örneğin bir dosyayı açmadan kullanmak bir hataya neden olur.

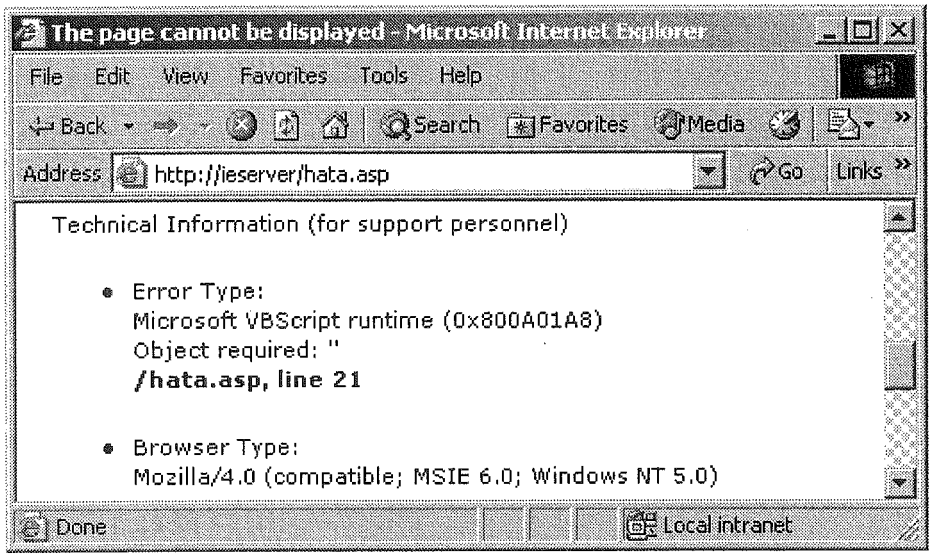
VBScript çalışma-zamanı hataları VBScript sistemin yürütemediği bir aksiyonu yapmaya çalıştığı zaman ortaya çıkan hatalardır. Çalışma-zamanı hataları scriptiniz yürütülürken ortaya çıkar; değişken ifadeleri değerlendirildiği zaman, ve bellek dinamik olarak tahsis edildiğinde.

Çalışma Zamanı ve Yazım Hata Kodları

Örnek çalışma zamanı hata kodları:

429	ActiveX bileşeni nesne oluşturamaz
507	Bir istisna meydana geldi
449	Argüman seçimlilik değil
17	İstenilen işlem yapılamaz
430	Class otomasyonu desteklemez
506	Class tanımlı değil
11	Sıfıra bölme hatası
48	DLL yükleme hatası
5020	Düzenli ifadede beklenilen ')'

Bir önceki sayfada çalışma zamanı hatalarının bir kısmı verilmiştir. Bu hatalar kodlarımızın sunucu üzerinde yürütülürken meydana gelebilecek hatalardır.



VBScript yazım hataları sizin VBScript yapılarınız bir veya daha çok VBScript script dili yazım kurallarına uymuyorsa bu tip hatalar meydana gelir. VBScript yazım hataları program yürütülmeden önce derleme aşamasında meydana gelir.

Örnek yazım hata kodları:

1052 Bir Sınıf içinde çok sayıda metod/özelliğe sahip olmaz

1044 Bir Sub prosedürü çağrıldığında parantez kullanılamaz

1053 Sınıf başlatma veya sonlandırma argümanlara sahip olmaz

1058 'Default' belirtimi yanlızca Property Get üzerinde olur

1057 'Default' belirtimi 'Public' ğide belirtmeli

1005 Beklenilen '('

1006 Beklenilen ')'

1011 Beklenilen '='

1021 Beklenilen 'Case'

Bir önceki sayfada yazım hatalarının bir kısmı verilmiştir. Bu hatalar kodlarımızın yazım aşamasında meydana gelebilecek hatalardır.

Çalışma zamanı hatalarını kotarmak için yazım yanlışı yapılmamış bir scriptimiz olmalı. Bu tür hatalardan yazmış olduğumuz program kodlarının çalışma zamanı ortaya çıkması mutlaka engellenmelidir, çünkü yazım hataları programınızın çalışma esnasında tamamen sona ermesine neden olacaktır. Yazım hatalarından kurtulduktan sonra yine bir problemle karşılaşabiliriz o da çalışma zamanı hatalarıdır. Çalışma zamanı hataları kullanıcıdan veya kullanıcı bilgisayarından kaynaklanabilir. Biz bu tür hatalardan kurtulmak veya bu hataları kontrol etme imkanına sahibiz.

On Error GoTo ifadesi yazmış olduğumuz kodların çalışması esnasında çıkan hataları kotarmamıza izin verir.

On Error Resume Next

On Error GoTo 0

Eğer kodunuzun herhangi bir yerinde On Error Resume Next ifadesini kullanmıyorsanız, meydana gelebilecek herhangi bir çalışma zamanı hatası gösterilmek üzere bir hata mesajına neden olur ve yürütme işlemi durur. Bununla birlikte, kodlarımızın üzerinde çalıştığı host makine davranışın ne olacağını belirler. Sunucular aynı hatayı farklı şekillerde kotarabilir. Bazı durumlarda, script debug ediciye hatanın çıktığı noktada başvurulur. Diğer durumlarda, herhangi bir hata olmamış gibi kullanıcıya herhangi bir mesaj verilmeyebilir. Bu açık bir şekilde meydana gelen hataları sunucuların nasıl kotardığının bir fonksiyonudur.

Herhangi bir prosedür içinde, yığın çağırma boyunca biryerde hata-kotarma izinli olmadığı sürece bir hatanın çok ciddi olması gerekmez. Eğer lokal hata-kotarma bir prosedürde izinli değilse ve bir hata meydana gelirse, kontrol hata-kotarma izinli bir prosedür bulunana kadar çağırma yığınına geri geçilir ve o noktada hata kotarılır. Eğer çağırma yığınında hata-kotarma için izinli bir prosedür bulunamaz ise, bir hata mesajı o noktada gösterilir ve yürütme durur veya sunucu uygun bir şekilde hatayı kotarır.

- On Error Resume Next çalışma zamanı hatasına neden olan ifadeyi izleyen ifadeye hemen geçerek, veya On Error Resume Next içeren en sık çağrılan prosedürü izleyen ifadeyi yürütmenin devam etmesini sağlar. Bu bir çalışma hatasına rağmen yürütmenin devam etmesini sağlar. Siz o zaman prosedürünüz içinde hata kotarma rutinleri yazabilirsiniz.

Bir On Error Resume Next ifadesi başka bir prosedür çağrıldığında pasif hale geçer, bundan dolayı siz o rutin içinde satır içi hata kotarma istiyorsanız herbir çağrılan rutinde bir On Error Resume Next ifadesini yürütmelisiniz. Bir prosedürden çıktığınızda, hata kotarma yeteneği çıkılan prosedüre girmeden önceki yerdeki olan hata kotarmaya döner. Eğer önce On Error Resume Next kullanımına izin verdiyseniz, hata kotarmayı izinsiz hale getirmek için On Error GoTo 0 kullanın.

Err nesnesinin metodlarını ve özelliklerini kullanarak hataları kontrol edebiliriz. Err nesnesinin metodları , Clear metodu nesnenin tüm güncel değerlerini temizler, Raise metodu ise çalışma esnasında bir hata oluşturur.

Err nesnesi özellikleri; Description, bir hata koduna karşılık gelen tanımlama kısmını string olarak içerir, Number, olabilecek bir hatanın sayısal değerini içerir, Source ise meydana gelen hatanın kaynağını belirtir.

Bir hata meydana geldiğinde , Err nesnesi başka hatalar için kullanılmayan ve hatayı belirleyen bir sayı ataması yapar. On Error Resume Next ifadesi kullanıldığında, hatanın sahip olduğu değerler temizlenir ve bu hatanın dikkate alınmamasını sağlar. Bu işlemi Clear metodunu kullanarak yaparız.

Meydana gelen hatayı belirlemek için onun numarasının ne olduğunu bilmemiz gerekli, çünkü hatayı kontrol etmek için bu numarayı kullanacağız. Hatanın numarasını belirlemek için Err.Number özelliğini kullanırız.

Örnek 1:

Aşağıdaki örnek On Error Resume Next ifadesinin kullanımını gösterir.

```
On Error Resume Next

Err.Raise 7

MsgBox "Hata # " & CStr(Err.Number) & " " & Err.Description

Err.Clear
```

Yukarıda örnekte Err.Raise 7 ifadesi yetersiz bellek hatasına neden olur. Err.Clear ifadesi ise bir sonraki olabilecek hatayı sezmek için o an oluşan hata bilgilerini temizler.

CStr() fonksiyonu bir sayının stringe dönüştürülmesini sağlar.

Örnek 2:

İşte başka bir örnek, bu örnekte meydana gelebilecek bir hatanın kullanıcıya mesaj olarak bildirilmesini sağlamak ve hata kodlarını ayırmak için Select Case yapısını kullanıyoruz.

```
. . .

Select Case Err.Number
```

```
        Case 0
                MsgBox "Herhangi bir hata yok! "
        Case 5
                MsgBox "Geçersiz prosedür çağrısı veya argüman"
        Case 7
                MsgBox "Yetersiz bellek"
        Case 11
                MsgBox "Sıfıra bölme hatası"
        . . .
        Case Else
                MsgBox "Hata: " & CStr(Err.Number) & " " & Err.Description
End Select
. . .
```

Başka bir örnek

```
<HTML>
<HEAD>
<TITLE>Hata Kotarma</TITLE>
<SCRIPT LANGUAGE="VBScript">
Function hatayaz(numara)
Select Case numara
        Case 1
                hatayaz = "Alan numarası bir olan boş, lütfen doldurun!"
        Case 2
                hatayaz = "Yazdığınız e-mail adresi geçersiz!"
        Case 3
                hatayaz = "SQL veritabanı güncelleme hatası!"
```

```
        Case 4
                hatayaz = "Alan numarası iki olan  boş, lütfen doldurun!"
        Case 5
                hatayaz = "Belirtilmeyen bir hata!"
End Select
End Function
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#000000"
VLINK="#000000">
<SCRIPT>
hatayaz(CInt(TRIM(Request.Form("kod"))))
</SCRIPT>
<BR><BR><BR>
<FORM ACTION="yaz.asp" METHOD="POST">
Yeni bir KOD değeri seçin
<SELECT NAME="kod">
<OPTION VALUE="1">1
<OPTION VALUE="2">2
<OPTION VALUE="3">3
<OPTION VALUE="4">4
<OPTION VALUE="5">5
</SELECT>
<INPUT TYPE="Submit" VALUE="Submit">
</FORM>
```

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

Verilen örnek programın tarayıcı üzerideki görünümü şekildeki gibidir.

BÖLÜM
ON İKİ

12

VBScript Fonksiyonları

- Tüm VBScript Fonksiyonları

VBScript Fonksiyonları

Aşağıda Visual Basic Script diline ait tüm fonksiyonlar liste halindedir. Daha iyi anlaşılması açısından bu fonksiyonları kullanarak değişik örnekler her fonksiyon başlığı altında verilmiştir.

Visula Basic Script Fonksiyonları

Abs	GetLocale	RGB
Array	GetObject	Right
Asc	GetRef	Rnd
Atn	Hex	Round
CBool	Hour	RTrim
CByte	InputBox	ScriptEngine
CCur	InStr	ScriptEngineBuildVersion
CDate	InStrRev	ScriptEngineMajorVersion
CDbl	Int	ScriptEngineMinorVersion
Chr	IsArray	Second
CInt	IsDate	SetLocale
CLng	IsEmpty	Sgn
Cos	IsNull	Sin
CreateObject	IsNumeric	Space
CSng	IsObject	Split
CStr	Join	Sqr
Date	LBound	StrComp
DateAdd	LCase	String
DateDiff	Left	StrReverse
DatePart	Len	Tan
DateSerial	LoadPicture	Time
DateValue	Log	Timer
Day	LTrim	TimeSerial
Eval	Mid	TimeValue
Exp	Minute	Trim
Filter	Month	UBound
Fix	MonthName	UCase
FormatCurrency	MsgBox	VarType
FormatDateTime	Now	Weekday
FormatNumber	Oct	WeekdayName
FormatPercent	Replace	Year

Abs

Bir sayının mutlak değerini döndürür.

Abs(sayı)

sayı argümanı herhangi geçerli bir ifade olabilir. Eğer sayı Null içeriyorsa, Null döndürülür; eğer ilk değer atanmamış değer ise, sıfır döndürülür.

```
<%
Dim x
x = Abs( -10 )
Response.Write  x
'sonuç 10 olur
%>
```

Bir sayının mutlak değeri onun işaretsiz halidir. Örneğin, Abs(-8) ve Abs(8) her ikiside 8 döndürür.

Aşağıdaki örnek bir sayının mutlak değerini hesaplamak için Abs fonksiyonunu kullanır.

```
<%
Dim sayı
sayı = Abs(45.2)  ' 45.2 değeri döner.
sayı = Abs(-45.2) ' 45.2 değeri döner
%>
```

Array

Variant bir dizi oluşturur.

Array(argumanlistesi)

Gerekli *argumanlistesi* argümanı Variant ile içerilen bir dizinin elemanlarına atanan değerlerin bir virgül ile ayrılmış listesidir. Eğer argüman belirtilmemiş ise, sıfır uzunluklu bir dizi oluşturulur.

```
<%

Dim dizi

dizi = Array(10,20,30)

dizi2 = dizi(2)   ' dizi2 şimdi 30 dur.

%>
```

Sabit diziler

Tek boyutlu bir dizi bir grup maddeye ulaşmanın bir yoludur siz dizi indislerini kullanarak herbir maddeye bireysel olarak ulaşabilirsiniz.

```
Dim komutlar(2)
```

Bu sizin dizi içinde üç elemana sahip olduğunuz anlamına gelir, bu elemanlar komut(0) ,komut(1) , ve komut(2) dir. Burada indisin ilk değerinin 0 dan başladığına dikkat edin.

Dinamik diziler

```
<%

Dim komut()

KomutSayisi = InputBox("Kaç tane komut var")

ReDim komut(KomutSayisi)

%>
```

Asc

Bir karakterin ASCII kodunu döndürür.

```
Asc(string)
```

string argümanı herhangi bir geçerli bir ifadedir. Eğer string karakterler içermiyorsa, bir çalışma zamanı hatası meydana gelir.

Aşağıdaki örnek, Asc herbir stringin ilk karakterinin ANSI karakter kodunu döndürür.

```
<%

Dim numara
```

```
numara = Asc("B")     ' 66 döndürür.

numara = Asc("b")     ' 98 döndürür.

numara = Asc("Araba") ' 65 döndürür.

%>
```

Atn

Bir sayının ark tanjantını döndürür.

Atn(sayı)

sayı argümanı geçerli bir sayısal ifade olabilir.

Atn fonksiyonu bir sağ taraf açının (sayı) iki tarafının oranını alır ve radyan olarak açıya benzeyeni döndürür. Oran açıya ayarlanan kenarın uzunluğu ile bölünen açının ters tarafının uzunluğudur. Sonuç bölgesi radyan olarak –pi/2 den pi/2 ye kadardır.

Dereceleri radyanlara çevirmek için, pi/180 ile dereceler çarpılır. Radyanları dereceleri çevirmek için ise radyanlar 180/pi ile çarpılır.

Aşağıdaki örnek pi nin değerini hesaplamak için Atn fonksiyonunu kullanır.

```
<%

Dim pi

pi = 4 * Atn(1)    'pi nin değeri hesaplanır.

%>
```

CBool

Boolean alt tipinin (ya True ya da False) bir Variant döndürür. Cbool fonksiyonu herhangi bir sayıyı Boolean alt tipine dönüştürür. Boolean çıkışı ya true yada false olur. Eğer dönüşüm başarılı ise, çıkış true olacaktır. Eğer dönüşüm başarısız ise, çıkış false olacak veya bir hata mesajı olacaktır.

Cbool(sayi)

```
<% birsayi=7.77 %>

<% =CBool(birsayi) %>
```

Çıkış: True olacaktır

<% karekter=abc %>

<% =CBool(karekter) %>

Çıkış False olacaktır.

CByte

Siz VarType() fonksiyonunu kullanarak alt tipi belirleyebilirsiniz.

Cbyte(sayi)

Cbyte fonksiyonu 0 ile 255 arasındaki herhangi bir sayıyı Byte alt tipine çevirir.

<% sayi=(9.876) %>

<% =CByte(sayi) %>

Çıkış 10 olacaktır

<% sayi=(255) %>

<% =CByte(sayi) %>

Çıkış 255 olacaktır

CCur

Boolean alt tipinin (8 byte) bir Variant döndürür. Siz VarType() fonksiyonunu kullanarak alt tipi belirleyebilirsiniz.

CCur(sayi)

CCur fonksiyonu herhangi bir sayıyı veya sayısal bir stringi Currency alt tipine dönüştürür.

Currency değerlerini dönüştürme aralığı :

-922,337,203,685,477.5808 to 922,337,203,685,477.5807

<% sayi=(12345) %>

<% =CCur(sayi) %>

Çıkış 12345 olacaktır

<% sayisalstring=("98765") %>

<% =CCur(sayisalstring) %>

Çıkış 98765 olacaktır

CDate

Boolean alt tipinin (8 byte) bir Variant döndürür. Siz VarType() fonksiyonunu kullanarak alt tipi belirleyebilirsiniz.

CDate (tarih)

Cdate fonksiyonu herhangi geçerli bir tarih ve zaman ifadesini Date alt tipine dönüştürür.

Default tarih çıkış formatı " **/**/** ", ve zaman çıkış formatı ise " **:**:** *M " şeklinde olacaktır. *M ifadesi ya AM yada PM olabilir. Dönüştürülebilecek tarih değerleri Ocak 1, 100 ve Aralık 31, 9999 arasındadır.

<% tarih = "June 26, 1943" %>

<% =CDate(tarih) %>

Çıkış 6/26/43 olacaktır

<% tarih = #6/26/43# %>

<% =CDate(tarih) %>

Çıkış 6/26/43 olacaktır

<% zaman="2:23:59 PM" %>

<% =CDate(zaman) %>

Çıkış 2:23:59 PM olacaktır

CDbl

CDbl fonksiyonu herhangi bir sayıyı Double alt tipine dönüştürür.

CDbl(sayi)

Dönüştürülecek değer aralığı: negatif değerler -1.79769313486232E308 ile -4.94065645841247E-324 arası ve pozitif değerler 4.94065645841247E-324 ile 1.79769313486232E308 arasındadır.

<% sayi=2348.9 %>

<% =CDbl(sayi) %>

Çıkış 2348.9 olacaktır.

Chr

Bir ASCII tamsayı değerini gösteren karakteri döndürür. Chr(32) space (boşluk) karakterini gösterir. Bu fonksiyon bir ANSI karakter kodunu bir karaktere dönüştürür.

<% =Chr(98) %>

Çıkış b olacaktır.

CInt

CInt fonksiyonu herhangi bir sayıyı Integer alt tipine dönüştürür. Dönüştürülecek değerler –32,768 ile 32,767 arasındadır.

<% sayi=1234.567 %>

<% =CInt(sayi) %>

Çıkış 1235 olacaktır

CLng

CLng fonksiyonu herhangi bir sayıyı Long alt tipine dönüştürür, burada kayan noktalı sayıları yuvarlar.

CLng(sayi)

Dönüştürülecek değerler -2,147,483,648 ile 2,147,483,647 arasındadır.

<% sayi=1234.6 %>

<% =CLng(sayi) %>

Çıkış 1235 olacaktır

Cos

Cos fonksiyonu bir sayı (açı) için kosinüs değerini döndürür.

```
Cos( sayi )
```

```
<% =Cos(45.0) %>
```

Çıkış 0.52532198881773

Açı olarak negatif bir sayıda kullanabilirsiniz.

```
<% =Cos(-45.0) %>
```

Çıkış 0.52532198881773

CreateObject

Bir otomasyon nesnesine bir referans döndürür veya oluşturur.

```
CreateObject(sunucuadı.tipadı [, lokasyon])
```

Sunucuadı gerekli olup nesneyi sağlayan uygulamanın adıdır, *tipadı* ise oluşturmak istediğiniz nesnenin sınıfı veya tipidir, *lokasyon* ise nesnenin oluşturulacağı yerdeki ağ sunucusunun adıdır.

Bir otomasyon nesnesi oluşturmak için, bir nesne değişkenine CreateObject tarafından döndürülen nesne atanır,

```
Dim ExcelSheet

Set ExcelSheet = CreateObject("Excel.Sheet")
```

Bu kod nesneyi oluşturan uygulamayı başlatır. Bir nesne oluşturulduğu için, sizin nesne değişkenini kullanarak tanımladığınız kod içinde ona işaret eder. Örnekte gösterildiği gibi, nesne değişkenini , ExcelSheet gibi, kullanarak yeni nesnenin özelliklerine ve metodlarına erişebilirsiniz, ve diğer Excel nesneleri, uygulama nesnesi ve ActiveSheet.Cell koleksiyonuda buna dahildir.

```
' Uygulama nesnesi içinde Excel görünür yapma.

ExcelSheet.Application.Visible = True

' Sheet içinde ilk hücreye text bilgi koyabiliriz.
```

```
ExcelSheet.ActiveSheet.Cells(1,1).Value="Sütun A ve satır 1"

' Sheet saklanıyor.

ExcelSheet.SaveAs "C:\OFFICE\EXCEL\DENEME.XLS"

' Uygulama nesnesindeki Quit metodu ile Excel kapatıyoruz.

ExcelSheet.Application.Quit

' Nesne değişkenini serbest bırakıyoruz.

Set ExcelSheet = Nothing
```

Uzak bir sunucu üzerinde bir nesne oluşturma internet güvenlik durumu devre dışı bırakıldığında oluşturulabilir. Siz CreateObject sunucuadı argümanına bilgisayar adını vererek uzak bir ağ bilgisayarı üzerinde bir nesne oluşturabilirsiniz. Verilen isim bir paylaşım adının makine ismi ile aynıdır. \\sunucu\genel adlı bir ağ paylaşımı için, sunucuadı "sunucu" dur. Buna ek olarak DNS adı veya IP adresini kullanarak sunucu adını belirtebiliriz.

Aşağıdaki kod "sunucu" adlı uzak bir ağ bilgisayarı üzerinde çalışan Excel için bir tutamacın versiyon numarasını döndürür.

```
Function GetirVersiyon
  Dim EXCELuyg
  Set EXCELuyg = CreateObject("Excel.Application","sunucu")
  GetirVersiyon = EXCELuyg.Version
End Function
```

Eğer belirtilen uzak sunucu yoksa veya bulunamıyorsa bir hata meydana gelir.

Örnek

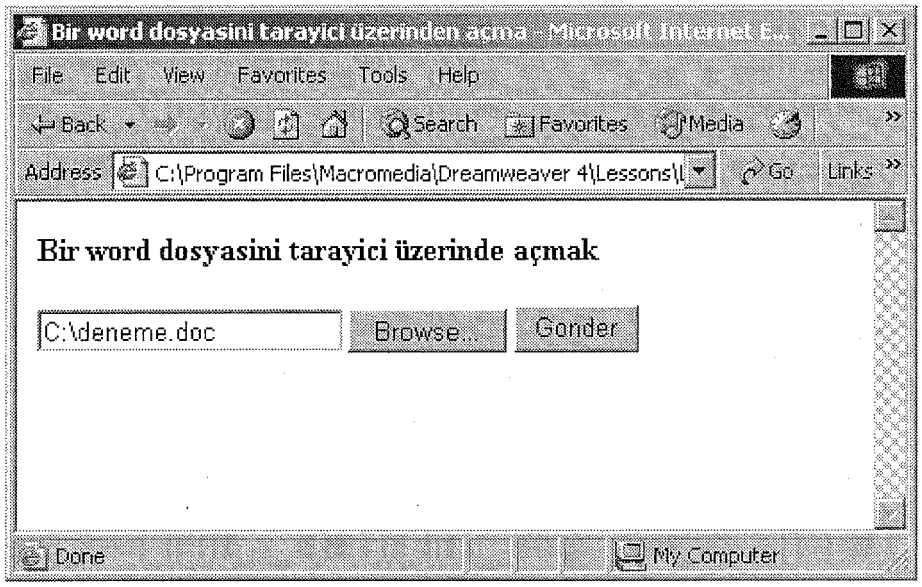
Microsoft Winword dosyasının VBScript kullanılarak tarayıcı üzerinden açılması.

```
<HTML>
<HEAD>
<TITLE>Bir word dosyasını tarayıcı üzerinden açılması</TITLE>
<SCRIPT LANGUAGE="VBScript">
```

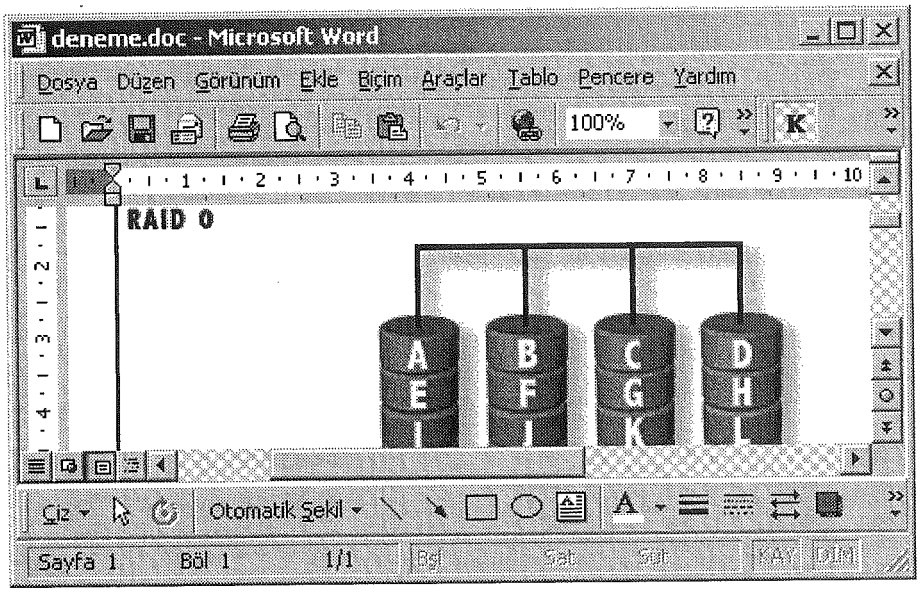
```
<!--
Sub Submit_OnClick
Dim wordUygulama
Dim wordForm
Dim mesaj
Set wordForm = Document.wForm
mesaj = MsgBox("OPEN: "+wordForm.dosya.Value,  vbOKCancel +
vbInformation,  "Word dosyası aç")
If mesaj = vbCancel Or wordForm.dosya.Value = "" Then
      Exit Sub
End If
' Bir otomasyon nesnesi oluşturmak için, bir nesne değişkenine CreateObject
' tarafından döndürülen nesne atanır
Set wordUygulama = CreateObject("Word.Application")
' Seçmiş olduğumuz Word dosyasını açıyoruz
wordUygulama.Documents.Open(wordForm.dosya.Value)
' Uygulama nesnesi içinde Word görünür yapma.
wordUygulama.Visible = True
' Word dosyamızı aktif hale getiriyoruz
wordUygulama.Documents(1).Activate
' Nesne değişkenini serbest bırakıyoruz.
Set wordUygulama = Nothing
End Sub
-->
</SCRIPT>
</HEAD>
```

```
<BODY>
<BR>
<B>Bir word dosyasını tarayıcı üzerinde açmak </B>
<BR>
<BR>
<FORM NAME="wForm" METOD="POST" ACTION="">
<INPUT TYPE="file" NAME="dosya">
<INPUT TYPE="BUTTON" NAME="Submit" VALUE="Gonder">
</FORM>

<BODY>
</HTML>
```

Verilen programın tarayıcı üzerindeki görünümü aşağıdaki şekildeki gibidir.



Aşağıda gönder düğmesine tıkladığınızda deneme.doc word dosyasının açılmış halini görürsünüz.



CSng

CSng fonksiyonu herhangi bir sayıyı Single alt tipine dönüştürür. Dönüştürülecek negatif değerler -3.402823E38 ile -1.401298E-45 arasında ve pozitif değerler 1.401298E-45 ile 3.402823E38 arasındadır.

```
<% sayi=5678.123 %>

<% =CSng(sayi) %>
```

Çıkış 5678.123

CStr

CStr fonksiyonu bir ifadeyi String alt tipine dönüştürür.

```
<% ifade=5678 %>

<% =CStr(ifade) %>
```

Çıkış 5678

Date

Date fonksiyonu bilgisayar sisteminizdeki bölgesel ayarlara göre o günkü tarihi döndürür.

Tarihsel olarak, iki dijit yıl bilgisi (DD/MM/YY) döndürülür. Bununla birlikte, millennium için şimdi dört dijitlik yıl bilgisi (DD/MM/YYYY) döndürülür. Aynı şeyler DateValue fonksiyonu içinde geçerlidir.

```
<%

' Günün tarihini 18/01/2002 şeklinde gösterir.

Response.Write Date()

%>
```

veya

```
<% =Date %>
```

çıktı olarak 27/01/2002 tarih bilgisi döner

DateAdd

DateAdd fonksiyonu herhangi bir tarihe bir zaman aralığını ekler, bu şekilde geçmiş veya ileri yöndeki bir tarihi elde edebiliriz.

DateAdd(Aralik, Sayi, Tarih)

aralik tarih üzerine eklemek istediğiniz zaman aralığının tipini tanımlayan argümandır. Siz çift tırnak arasına ayarlamaları yerleştirmeniz gerekir. Yanlızca aşağıdaki ayarlamaları kullanabilirsiniz.

Ayarlama	Tanımı
YYYY	Yıl
Q	Dörtte bir
M	Ay
Y	Yılın günü
D	Gün
W	Hafta günü
WW	Yılın haftası
H	Saat
N	Dakika
S	Saniye

sayi argümanı aralık için bir çarpandır (kaç gün, kaç hafta, kaç ay, ...). Eğer bu pozitif bir sayı ise, siz zaman olarak ileri yönde gideceksiniz. Negatif bir sayı zaman olarak geri yöne doğru sizi götürecektir.

tarih argümanı bir tarih veya zamanı gösterir.

Eğer siz o günkü tarih ve zamanı kullanmak istiyorsanız, basit olarak Date veya Now fonksiyonunu kullanabilirsiniz.

Bilinen bir tarihten daha önceki bir tarihi hesaplamak için, negatif bir ofset değeri kullanın. Örneğin, aşağıdaki kod parçası tam olarak bir yıl öncesinin nasıl hesaplanacağını gösterir:

```
Response.Write DateAdd("yyyy", Now() ,-1 )
```

Diğer örnekler,

```
<% =Date %>                          16/03/99
<% =DateAdd("WW", 6, Date) %>  27/04/99
<% =Now %>                           16/03/99 12:14:00 PM
<% =DateAdd("S", 30, Now) %>   16/03/99 12:14:30 PM
<% ="1/1/2001" %>                                  1/1/2001
<% =DateAdd("YYYY", 1000, "1/1/2001") %>   1/1/3001
```

DateDiff

Bilinen iki tarih arasındaki aralığı hesaplayan DateDiff fonksiyonudur.

```
DateDiff(Aralik, Tarih1, Tarih2, HaftaninIlkGunu, YilinIlkHaftasi)
```

DateDiff fonksiyonu iki farklı tarih arasındaki zaman olarak miktarı hesaplar. Üç tane önemli argümanı var.

Aralik argümanı zaman farkını hesaplamak için kullanmak istediğiniz zaman aralığının tipini tanımlar. Yanlızca aşağıdaki ayarlamalar kullanılabilir. Ayarlamaları çift tırnak arasına yerleştirmek zorundasınız.

Ayarlama	Tanımı
YYYY	Yıl
Q	Dörtte bir
M	Ay
Y	Yılın günü

D	Gün
W	Hafta günü
WW	Yılın haftası
H	Saat
N	Dakika
S	Saniye

tarih1 argümanı ilk tarih, *tarih2* argümanı ise ikinci tarihi gösterir.

<%=DateDiff("d", Date, "1/1/2000") %> Çıktısı 291 olur

İki tarihin sırası aradaki farkın pozitif veya negatif olmasını belirler.

<% eskiTarih = #6/26/43# %>

<% simdikiTarih = Now %>

<% =DateDiff("s", simdikiTarih, eskiTarih) %> Çıktısı -3130748108

DateDiff fonksiyonunun iki tane seçimlilik olan argümanı var. HaftaninIlkGunu argümanı yanlızca aşağıdaki Tarih ve Zaman Sabitlerinde tanımlı olan sabitleri veya değerleri kullanır.

Sabit	Değer	Tanımı
vbSunday	1	Pazar
vbMonday	2	Pazartesi
vbTuesday	3	Salı
vbWednesday	4	Çarşamba
vbThursday	5	Perşembe
vbFriday	6	Cuma
vbSaturday	7	Cumartesi
vbUseSystemDayOfWeek	0	Haftanın ilk günü için sistem ayarlarınızda belirtilen haftanın gününü belirtir.
vbFirstJan1	1	Ocak 1 deki hafta için kullanın.
vbFirstFourDays	2	Yeni yıldaki en az dört güne sahip ilk haftası için kullanın.
vbFirstFullWeek	3	Yılın tam ilk haftası için kullanın.

Bu örnekte, Salı haftanın ilk günü olarak tanımlıdır. Şu anki tarih ve 1/01/2001 tarih arasında kalan, salı gününden başlayarak şimdi tanımlı olan kaç tane haftanın olduğu hesaplanıyor.

<% =DateDiff("w", Date, "1/1/2001", 3) %> Çıktısı 93 hafta olur

<% =DateDiff("w", Date, "1/1/2001", VBTUESDAY) %> Çıktısı 93 hafta olur

YilinIlkHaftasi argümanı yanlızca yukarıda listelenen Tarih ve Zaman sabitlerinde tanımlı olan sabitler veya değerleri kullanabilir.

<%=DateDiff("ww",Date,"1/1/2001",1,VBFIRSTFOURDAYS)%>

Çıktısı 94 olur

Örneğin, Ocak 1, 1900 dan bu yana günlerin sayısını bulmak için aşağıdaki gibi bir kod yazarız:

Response.Write DateDiff("d",#1/1/1900#, Now())

DatePart

Bir tarihin parçasını, veya gün, haftanın günü, ay, dörtte bir, yıl, dakika, saniye, veya saat ile gösterilen zamanı döndürür.

DatePart(Aralik, Tarih, HaftaninIlkGunu, YilinIlkHaftasi)

İki tane zorunlu argümanı var. *Aralik* argümanı zaman aralığının tipini gösterir ve yanlızca aşağıdaki ayarlamalar kullanılabilir. Ayarlamaları çift tırnak arasına yerleştirmek zorundasınız

Ayarlama	Tanımı
YYYY	Yıl
Q	Dörtte bir
M	Ay
Y	Yılın günü
D	Gün
W	Hafta günü
WW	Yılın haftası
H	Saat
N	Dakika
S	Saniye

Tarih argümanı sizin belirttiğiniz tarih ve zamandır.

<% =Date %> Çıktısı 16/03/99 olur

<% =DatePart("D", Date) %> Çıktısı 16 olur

Haftanın günü Tarih ve Zaman sabitlerinde verilen bir sayıdır. Örneğin 7 rakamı Cumartesi gününü gösterir.

<% =DatePart("W", 1/1/2000) %> Çıktısı 7 olur

DatePart fonksiyonu iki seçimlilik argümana sahiptir.

HaftaninIlkGunu argümanı yanlızca aşağıdaki Tarih ve Zaman Sabitlerinde tanımlı olan sabitleri veya değerleri kullanır.

Bu örnekte, Salı haftanın ilk günü olarak tanımlıdır. Bu yüzden, Cumartesi haftanın 5 nci günü olur.

<% =DatePart("W", "1/1/2000", 3) %>

Çıktısı 5 olur

<% =DatePart("W", "1/1/2000", VBTUESDAY) %>

Çıktısı 5 olur

yilinIlkHaftasi argümanı yanlızca yukarıda listelenen Tarih ve Zaman sabitlerinde tanımlı olan sabitler veya değerleri kullanabilir.

Bu örnekte 1999 yılında kaç tane tam 7 güne sahip hafta sayısını döndürür.

<% =DatePart("WW", "12/31/1999", 1, VBFIRSTFULLWeek) %>

Çıktısı 52 olur

DateSerial

DateSerial fonksiyonu bir argümanı Date alt tipinin bir varyantına dönüştürür.

DateSerial(yıl, ay, gün)

Üç adet argüman var. *yıl* argümanı bir string veya tamsayı olarak yıl bilgisi, *ay* argümanı bir string veya tamsayı olarak ay bilgisi, *gün* argümanı bir string veya tamsayı olarak gün bilgisidir.

<% =DateSerial(1943, 6, 26) %>

Çıktısı 26/06/43 olur

<% =DateSerial("43", "6", "26") %>

Çıktısı 26/06/43 olur

DateValue

DateValue fonksiyonu bir argümanı Date alt tipinin bir varyantına dönüştürür.

DateValue(Tarih)

tarih argümanı tarih formatında geçerli bir string olmalı yoksa bir hata mesajı gelecektir.

<% birSting = "26/06/1945" %>

<% =DateValue(birString) %>

Çıktısı 26/06/1945 olur

<% =DateValue("1/1/2002") %>

Çıktısı 1/1/2002 olur

Day

Ayın gününü gösteren 1 ve 31 arasında tam bir sayı döndürür.

Day(tarih)

tarih argümanı bir tarih ile gösterilen herhangi bir ifadedir. Eğer tarih Null içeriyorsa, Null döndürülür.

Aşağıdaki örnek belirtilen bir tarihten ayın gününü almak için Day fonksiyonunu kullanır.

Dim gun

gun = Day("October 10, 2002") ' gun 10 olur.

<% =Day(Date) %> Çıktısı 16 olur

<% =Day("6/26/1943") %> Çıktısı 26 olur

Eval

Bir ifade olarak fonksiyona geçilen scripti değerlendirir. Bir anda yanlızca bir ifade değerlendirebilirsiniz. Çok sayıda satırdan oluşan kodları yürütmek için Execute ifadesini kullanın.

[sonuc =]Eval(ifade)

Argüman olarak verilen sonuc seçimlilik olup değer atamasının yapılacağı değişkendir. Eğer sonuc belirtilmemişse , Execute yapısı yerine kullanılabileceğini gözönünde bulundurun. İfade herhangi bir uygun VBScript ifadesi içeren stringdir.

VBScript içinde, x=y iki şekilde yorumlanabilir. İlki y nin değerinin x e atandığını gösteren bir değer atama yapısı. İkincisi ise x ile y aynı değeri içeriyormu diye test etmeyi gösterir. Eğer sonuç doğru ise True , aksi taktirde False olur. Eval metodu daima ikinci yorumu kullanır, Execute yapısı ise daima ilk yorumu kullanır.

Eval fonksiyonunun kullanımı:

```
Sub birTahmin
  Dim tahmin, rasgelesayı
  rasgelesayı = Int((10) * Rnd(1) + 1)
  tahmin = CInt(InputBox("Tahminiz nedir:",,0))
  Do
    If Eval("tahmin = rasgelesayı") Then
      MsgBox "Tebrikler! Tahminiz doğru!"
      Exit Sub
    Else
      tahmin = CInt(InputBox("Tekrar deneyin lütfen.",,0))
    End If
  Loop Until tahmin = 0
End Sub
```

Exp

Bir üstel kuvvete yükseltilen logaritmaların (e) doğal tabanını döndürür.

Exp(sayı)

sayı argümanı herhangi bir geçerli sayısal ifade olabilir. Eğer sayının değeri 709.782712893 sayısını geçerse hata meydana gelir. e sabiti yaklaşık olarak 2.718282 dir.

Örnekler:

<% =Exp(3.269) %>

Çıktısı 26.2850411552082 olur.

<% =Exp(-3.269) %>

Çıktısı 0.038044452511799 olur.

Aşağıdaki örnekte radyan olarak bir açı tanımlar ve hiperbolik sinüs değerini hesaplar.

```
<%
Dim aci, asin
aci = 1.3
asin = (Exp(aci)-Exp(-1 * aci))/2
%>
```

Filter

Parametre olarak geçilen koşul tabanlı bir string dizisinin bir alt kümesini döndürür.

Filter(string, altstring, dahil, karsılastır)

Filter fonksiyonu bir veya daha çok karakter dizisini ilk indis değeri sıfırdan başlayan bir dizinin elemanları içinde araştırır, ve uyum gösteren diziyi içeren elemanlar ile yada bu elemanlar olmaksızın yeni bir dizi oluşturur, Siz Split fonksiyonunu kullanarak sıfır tabanlı bir string dizi oluşturabilirsiniz. Join fonksiyonu Filter fonksiyonu uygulandıktan sonra tekrar bir düzenleme için kullanılır.

Filter fonksiyonu zorunlu iki argümana sahip olup , diğerleri seçimliliktir.

string argümanı sıfır tabanlı bir string dizinin adı, *altstring* dizi içinde araştırılacak olan bir veya daha çok karakterden oluşan bir alt dizini oluşturur,

dahil argümanı True veya False olabilir. Eğer True ise, yanlızca arama alt dizinini içeren değerleri taşıyan bir dizi döndürülür, eğer False ise, yanlızca arama alt dizinini içermeyen değerleri taşıyan bir dizi döndürülür.

karsılastır bir seçimlilik argüman olup , ya sabit yada aşağıdaki sabitlerden birini alır.

Sabit	Değer	Tanımı
VBBinaryCompare	0	İkili karşılaştırma
VBTextCompare	1	Text bazında karşılaştırma
VBDataBaseCompare	2	Karşılaştırma bilgisi veritabanı içinde

Fix

Bir tamsayı değere kesilmiş bir sayıyı döndürür. Sayının kesirli kısmını atar ve geri kalan tamsayı kısmını döndürür. Aynı işi yapan bir Int fonksiyonuda var ve bu fonksiyonda tamsayı bir sonuç döndürür, fakat Int ve Fix fonksiyonları arasında bir fark vardır. Int fonksiyonu negatif sayıları daha küçük bir değere yuvarlar , Fix fonksiyonu ise negatif sayıları ise daha üst bir sayıya yuvarlar.

Fix(sayı)

sayı argümanı geçerli sayısal bir değer olabilir. Eğer sayı Null taşıyorsa, Null döndürülür.

Eğer sayı negatif ise, o zaman fix sayıya eşit veya daha büyük ilk negatif tamsayıyı döndürür. örneğin, -7.3 değerine fix fonksiyonu uygulandığında –7 değerini elde ederiz.

```
<%
Dim toplam
toplam = fix(55.2) ' 55 döner.
toplam = fix(-24.7) ' –24 döner.
toplam = fix(-23.3) ' –23 döner.
toplam = Int(-24.7) ' -25 döner.
%>
```

FormatCurrency

Belirtilen kritere göre currency değerini formatlamamızı sağlayan fonksiyon.

Örneğin,

1009.7386 değerini currency olarak formatlamak için, en yakın penny e yuvarlandı.

```
Dim s

s = FormatCurrency( 1009.7386, 2, True, True, True )

Response.write s

' $1.009.74 olarak bir çıktı olur
```

FormatDateTime

FormatDateTime fonksiyonu tarihleri bilinen formatlarda göstermemizi sağlar. Siz tarih ve bir formatlama (tarihin nasıl gösterilmesi gerektiğini gösteren bir sabit değer) şekli argüman olarak vermelisiniz.

Örneğin:

```
MsgBox FormatDateTime(Now, vbGeneralDate) '15/6/99 5:15:20 PM

vbLongDate Monday, June 15, 1999

vbShortDate 15/6/99

vbLongTime 5:15;20 PM

vbShortTime 17:15
```

FormatNumber

FormatNumber fonksiyonu sayısal bir ifade için formatlı bir sayı değeri döndürür.

FormatNumber(ifade, ondalikhanesayisi, dahiltemelkisim, negatifsayiicinparantezkullan, gruphanesi)

Bir tane zorunlu argümanı vardır.

ifade argümanı formatlı bir sayıya dönüştürülecek olan sayıyı gösterir.

<% =FormatNumber(12345) %>

Çıkış : 12,345.00

bu fonksiyon değerleri yuvarlar.

<% =FormatNumber(12345.67899) %>

Çıkış : 12,345.68

4 tane seçimlilik argümanı vardır.

ondalikhanesayisi seçimlilik olan bu argüman ondalık kısmında kaç hane olacağını gösterir.

<% =FormatNumber(12345.67899, 4) %>

Çıkış : 12,345.6790

dahiltemelkisim argümanı temel olarak sıfırı dahil eder.

Bu argüman için yanlızca ya sabitleri yada Tristate sabitleri kullanmak zorundayız.

Sabit	Değer	Tanımı
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Default ayarları kullan

<% =FormatNumber(.77, 4, -1) %>

Çıkış : 0.7700

negatifsayiicinparantezkullan argümanı negatif bir sayıyı parantez içine almayı sağlar.

Bunu sağlamak için yanlızca sabitleri yada Tristate sabitlerini kullanmak zorundasınız.

Sabit	Değer	Tanımı
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Default ayarları kullan

<% =FormatNumber(-12345.67899, 2, 0, -1) %>

Çıkış : (12,345.68)

gruphanesi argümanı bilgisayar ayarlarına göre bir sayının gösterilmesini sağlar.

Bunu sağlamak için yanlızca sabitleri yada Tristate sabitlerini kullanmak zorundasınız.

Sabit	Değer	Tanımı
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Default ayarları kullan

<% =FormatNumber(12345.67899, 2, 0, -1, -1) %>

Çıkış : 12,345.68

FormatPercent

Yüzde değerler olarak sayıları veya sayısal ifadeleri formatlamak için kullanılır.

FormatPercent(ifade, ondalikhanesayisi, dahiltemelkisim,
negatifsayiicinparantezkullan, gruphanesi)

FormatPercent fonksiyonu (%) işareti eklenmiş olan sayısal bir ifade için formatlı bir yüzde değeri döndürür.

Bir tane zorunlu parametresi vardır.

ifade argümanı formatlı bir yüzdeye dönüştürülecek sayıdır.

<% =FormatPercent(.77) %>

Çıkış : 77.00%

Bu fonksiyon yuvarlama yapar.

<% =FormatPercent(.678999) %>

Çıkış : 67.90%

Dört tane seçimlilik argüman vardır.

ondalikhanesayisi argümanı ondalık noktadan sonraki hanelerin sayısını seçmenize izin verir.

<% =FormatPercent(.123456789, 4) %>

Çıkış : 12.3457%

dahiltemelkisim argümanı sıfır temel alınmasını sağlar.

Bu argüman için tablodaki sabitler veya bunların değerlerini kullanmalısınız.

Sabit	Değer	Tanımı
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Default ayarları kullan

<% =FormatPercent(.0098, 4, -1) %>

Çıktısı : 0.9800%

negatifsayiicinparantezkullan argümanı negatif bir sayının parantez içine alınmasını sağlar.

Bu argüman için tablodaki sabitler veya bunların değerlerini kullanmalısınız.

Sabit	Değer	Tanımı
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Default ayarları kullan

<% =FormatPercent(-.77, 2, 0, -1) %>

Çıkış : (77.00%)

gruphanesi parametresi formatlamanın bilgisayar ayarlarına göre yapılmasını sağlar. Tristate sabitleri kullanılır.

Sabit	Değer	Tanımı
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Default ayarı kullan

<% =FormatPercent(.77, 2, 0, -1, -1) %>

Çıkış : 77.00% olur.

GetLocale

Scriptin üzerinde çalıştığı bilgisayarın LocaleID sini döndürür.

GetLocale

GetLocale fonksiyonu LCID değerini döndürür.

LCID coğrafik yerlere göre tek olarak tanımlanmış kısa bir string, onlu değer, veya onaltılık değerdir. Değişik coğrafik yerler kullanıcı dili, ülke, ve kültürüne göredir. Örneğin, "English-United States" ya "en-us", veya "1033", yada "0x0409" olarak belirtilir.

<% =GetLocale() %>

Çıkış : 1033 olur.

GetObject

Bir dosyadan yüklenen bir nesne için bir nesne referansı döndürür. Siz dosya adını, ve seçimlilik olarak ProgID sini sağlarsınız. Bir ProgID bir ProjeAdı.SınıfAdı şeklindedir, Excel.Worksheet, veya Word.Application gibi.

GetObject (yol, sinif)

GetObject fonksiyonu bir dosyadaki bir otomasyon nesnesine erişmek için ve bir nesne değişkenine bir nesne atamak için kullanılır.

İki tane seçimlilik argümanı vardır.

(yol, sinif)

yol argümanı tam yolu gösterir ve erişilecek olan nesne dosyasını gösterir.

Eğer yol argümanı kullanılmaz ise, sinif argümanını kullanmak zorundasınız.

sinif argümanı nesne tarafından sağlanan uygulamanın adıdır veya oluşturulacak nesnenin adıdır.

```
<% yol = "C:\deneme\nesne.txt" %>

<% Set nesne = GetObject(yol) %>

<% Set nesne = GetObject(yol, "Uygulama.BirNesne") %>
```

GetRef

Bir DHTML olaya bağlayabileceğiniz bir fonksiyon işaretçisini döndürür.

GetRef (FonksiyonVeyaAltrutin)

GetRef fonksiyonu bir fonksiyon veya alt rutini bir DHTML sayfa üzerindeki bir olaya bağlar.

NesneAdi gerekli olup DHTML olayının bir DHTML nesnesinin adı.

OlayAdi gerekli olup fonksiyon veya alt rutin ilişkilendirileceği bir DHTML olayının adı.

FonksiyonVeyaAltrutin zorunlu olup DHTML olayının birleştirileceği bir VBScript fonksiyonu veya alt rutini adıdır.

Yukarıda verilenler aşağıdaki şekilde Set anahtar sözcüğü ile kullanılmalıdır.

Set NesneAdi.OlayAdi = GetRef(FonksiyonVeyaAltrutin)

Hex

Bir onaltılık string olarak bir sayısal değeri döndürür.

Hex (sayi)

Hex fonksiyonu bir tamsayının onaltılık tabandaki değerini döndürür.

```
<% =Hex(123) %>
```

Çıkış : 7B

siz negatif bir tamsayıyıda kullanabilirsiniz.

```
<% =Hex(-123) %>
```

Çıkış : FF85

Ondalık kısmı olan bir sayı (kayan noktalı sayı) için, onlu noktadan sonraki sağdaki haneler atılır.

```
<% =Hex(123.256) %>
```

Çıkış : 7B

Aşağıdaki örnek verilen bir sayının Hex olarak döndürülmesini sağlayan bir fonksiyon içeriyor.

```
<SCRIPT LANGUAGE="VBScript" TYPE="text/vbscript">
<!--
  Function cevirHex(sayi)
    cevirHex = Hex(sayi)
  End Function
-->
</SCRIPT>
```

Hour

Belirli bir zaman ifadesinden saat bilgisini döndürür.

```
<%
Dim Saat
' Now fonksiyonundan günün saat bilgisini getir.
```

```
Saat = Hour(Now)

' Eğer saat bilgisi 12 den küçükse o zaman bir mesaj göster

If Saat < 12 then

   Response.Write "Öğleden önce"

Else

Response.Write "Öğleden sonra"

End if

%>
```

InputBox

Örneğin, VBScript kodu kullanıcının adını ister ve onu web sayfanız üzerinde bir mesaj kutusu içinde gösterir ve isminizi büyük harflerle yazar.

```
InputBox(prompt[,title][,default][,xpoz][,ypoz][,helpfile,context])


<HTML>

<HEAD>

<TITLE> "VBScript InputBox örneği"</TITLE>

<SCRIPT LANGUAGE="VBScript">

<!--

' Scriptinizi buraya yazın.

Adınız = InputBox("Adınız nedir?")

MsgBox "Adınız"

' veya adınızı şöylede gösterebilirsiniz:

Document.Write " <H1>"

Document.Write Adınız

Document.Write " <VH1>"
```

- ->

</SCRIPT>

</HEAD>

</BODY>

</HTML>

InStr

Seçimlilik olan başlangıç , arama işleminin nereden başlayacağını gösterir, durum ise aramanın büyük-küçük harf ayırımı yapılıp yapılmayacağınıda belirleyebilirsiniz.

Instr([başlangıç,]orijinalString,altString[,durum])

birString = "Bu bir denemedir"

Pozisyon = InStr(birString, "de") 'Pozisyon = 8

Pozisyon = InStr(8, birString, "ir") 'Pozisyon = 15

Son satırdaki ifadede arama işlemi 5. pozisyondaki "ir" yerine 8. pozisyondan itibaren arama işlemi başlamış olup ikinci "ir" alt stringi getirilmiştir.

InStrRev

InStrRev(string, altstring, baslangic, karsilastirma)

Verilen argüman değerlerine göre bir alt string getirir. Eğer bir uyum bulanamaz ise, o zaman sıfır çıkışı gösterir.

İki tane zorunlu argümanı vardır.

string argümanı içinde arama yaptığınız string.

altstring argümanı aranmasını istediğiniz string.

<% =InStrRev("ABCDE ABCDE", "C") %>

Çıkış : 9

İki tane seçimlilik argümanı vardır.

baslangic argümanı aramanın soldan itibaren nerden başlayacağını belirten sayısal bir değer.

Örnekte, arama soldan pozisyon 4 te başlar, ve arama sağdan sola doğru yapılıyor.

<% =InStrRev("ABCDE ABCDE", "C", 4) %>

Çıkış : 3

karsilastir argümanı arama işlemi sırasında karşılaştırmanın hangi şekilde yapılacağını belirtir. Bunun için aşağıdaki sabitleri veya değerlerini kullanabilirsiniz.

Sabit	Değer	Tanımı
VBBINARYCOMPARE	0	Binary karşılaştırma (büyük-küçük harf ayırımı yapar)
VBTEXTCOMPARE	1	Text karşılaştırma (büyük-küçük harf ayırımı yapmaz)
VBDATABASECOMPARE	2	Veritabanı içi bilgi karşılaştırması

Örnekte, karşılaştırma argümanı olarak VBBinaryCompare veya 0 kullanılarak,Tüm büyük/küçük harf ayırımlarına her iki aramadada uyulur.

<% =InStrRev("ABCDE ABCDE", "c", 4, 0) %>

Çıkış : 0 olur.

Örnekte, karşılaştırma argümanı olarak VBTextCompare veya 1 kullanılarak,Tüm büyük/küçük harf ayırımlarına her iki aramadada uyulmaz.

<% =InStrRev("ABCDE ABCDE", "c", 4, VBTextCompare %>

Çıkış : 3 olur.

Başka bir örnek,

```
birString = "Bu bir denemedir"

Pozisyon = InStrRev(birString, "d") ' Pozisyon = 8 olur.
```

Int

Bir değişkenin değerini bir Tamsayı değerine dönüştürür. Bir string değeri bir tamsayıya değiştirmek veya reel değerlerin kesirli kısmını kesip atmak için kullanılır.

int (sayı)

Eğer sayı negatif ise, o zaman sayı eşit veya daha küçük ise int fonksiyonu ilk negatif tamsayıyı döndürür, örneğin, -5.4 sayısı int fonksiyonu sonucu –6 olarak döner.

sayı argümanı geçerli sayısal bir değer olabilir. Eğer sayı Null taşıyorsa, Null döndürülür.

```
sayı = Int(25.8)   ' 25 döner.
sayı = Int(-25.8)  ' –26 döner.
sayı = Int(-25.2)  ' –26 döner.

<%
Dim x
Dim y
x = 201.45
y = int(x)
'y değeri 201 olacaktır.
%>
```

IsArray

Eğer bir değişken bir dizi alt tipi ise True değerini döndürür.

IsArray(degisken)

degisken argümanı herhangi bir değişken olabilir. Eğer değişken bir dizi ise IsArray True değerini döndürür, diğer halde False döner. IsArray özellikle dizileri taşıyan variantlar ile faydalıdır.

```
<%
Dim degisken
Dim dizi(2)
dizi(0) = "BMW"
dizi(1) = "Mercedes"
degisken = IsArray(dizi) 'degisken "True" değerini taşır.
%>
```

IsDate

Eğer argüman alt tip Date bir variantı ise veya bir Date alt tipine dönüştürülebiliyorsa True değerini döndürür, aksi taktirde False döndürür.

IsDate(ifade)

ifade argümanı herhangi bir tarih ifadesi veya bir tarih veya zaman olarak kabul edilebilir string ifade olabilir.

```
<%
Dim tarihim
Dim tarihin
Dim tarihsiz
Dim tarih
tarihim = "Mayıs 13, 1999"
tarihin = #5/13/99#
tarihsiz = "Ne haber"
tarih = IsDate(tarihim)    ' True döner.
tarih = IsDate(tarihin)  ' True döner.
tarih = IsDate(tarihsiz) ' False döner.
%>
```

IsEmpty

Bir değişkene ilk değer ataması yapılıp yapılmadığını gösteren bir boolean olarak True veya False değerlerinden birini döndürür.

```
IsEmpty(ifade)
```

ifade argümanı herhangi bir ifade olabilir. Bununla birlikte, bireysel değişkenlerinin ilk değer alıp almadığını belirlemek için IsEmpty kullanıldığı için, ifade argümanı daha çok tek bir değişken adıdır.

```
<%
Dim degisken
Dim sonuc
sonuc = IsEmpty(degisken)  'True döner.
degisken = Empty           'Empty ataması yapılıyor.
sonuc= IsEmpty(degisken)   'True döner.
%>
```

IsNull

Eğer ifade argümanı geçersiz bir veri (Null) içeriyorsa, True değerini aksi taktirde False değerini döndürür. İfade argümanı herhangi bir ifade olabilir.

```
IsNull(ifade)
```

ifade birden fazla değişken içeriyorsa, Null içeren bir değişken tüm ifade için True değerinin dönmesine neden olur.

```
<%
Dim degisken
Dim sonuc
degisken = Null            'Null ataması yapılıyor.
sonuc = IsNull(degisken)   'True döner.
%>
```

IsNumeric

Eğer argüman bir sayı veya bir sayıya döndürülmüş ise True değerini döndürür.

```
IsNumeric(ifade)
```

ifade argümanı herhangi bir ifade olabilir.

Eğer tüm ifade bir sayı olarak değerlendirilebilirse, True değeri döner, aksi taktirde, False döner. Eğer ifade bir tarih ifadesi ise, IsNumeric fonksiyonu False değerini döndürü.

```
<%
Dim degisken
Dim sonuc
degisken = 16   ' bir değer atanıyor.
sonuc = IsNumeric(degisken)   ' True döner.
degisken = "25. sokak"          ' bir değer atanıyor.
sonuc = IsNumeric(degisken)   ' False döner.
%>
```

IsObject

Eğer argüman alt tip nesneye işaret ediyorsa, True değerini döndürür aksi taktirde False değerini döndürür.

```
IsObject(ifade)
```

İfade argümanı herhangi bir ifade olabilir. Eğer ifade Object alt tipi değişkeni veya kullanıcı tanımlı bir nesne ise , IsObject True döndürür; yoksa False döndürür.

Aşağıdaki örnek eğer bir tanımlayıcı bir nesne değişkenini gösteriyorsa IsObject fonksiyonunu bunu belirlemek için kullanıyor.

```
Dim tamsayi
Dim kontrol
```

```
Dim nesne

Set nesne = Me

kontrol = IsObject(nesne)   ' True döner

kontrol = IsObject(tamsayi)   ' False döner
```

Join

Belirtilen bir ayırıcı ile ayrılmış bir string veya bir stringler dizisini kabül eder. Join fonksiyonu, Split fonksiyonunun tersi bir işlem yapar. String birleştirme işlemini & operatörü ile de yapabiliriz.

```
Join ( string )
```

```
<%

Response.Write "Zaman şu an " & time & " dır."

%>
```

Sonuç olarak "Zaman şu an 4:28:31 PM dır".

```
<%

Dim dizi(2)

dizi(0) = "Bugün "

dizi(1) = "hava çok "

dizi(2) = "güzel."

Sonuc = Join(dizi)

%>
```

Sonuç olarak "Bugün hava çok güzel." tam stringi elde ederiz.

Lbound

Bir dizinin alt-sınırını döndürür.

```
Dim renkler(1 to 10 )
```

Burada 10 elemanlı bir dizi var ve ilk indisi 1 den (0 değil) başlamakta olup son indiside 10 dur.

Geçerli dizi indis değerleri bölgesi dizinin sınırlarıdır. Siz alt sınır ve üst sınır değerlerini LBound(DiziDegiskeni) ve UBound(DiziDegiskeni) fonksiyonlarını ile bulabilirsiniz.

Örneğin,

```
Dim Renk( 1 to 8 )

Response.write "Alt sınır değeri=" & LBound(Renk)&"<BR>"

Response.write "Üst sınır değeri=" & UBound(Renk)&"<BR>"
```

LCase

Verilen bir stringin tüm karekterlerini küçük harflere çevrilmiş hali ile döndürür.

```
LCase ( string )

Lcase( "KASAbanıN SIRrı") 'Sonuç "kasabanın sırrı" olacaktır.

<% =LCase("İstanbul Teknik Üniversitesi") %>
```

Çıkış istanbul teknik üniversitesi

Bir string değişken adınıda kullanabilirsiniz

```
<% astring=" İstanbul Teknik Üniversitesi" %>

<% =LCase(astring) %>
```

Çıkış: istanbul teknik üniversitesi

Left

Bir string içinde başlangıç olarak belirtilen bir indeksden başlayarak ve başlangıcıda dahil olmak üzere bir string döndürür.

```
Left("Bugün hava çok güzel", 5)
```

Sonuç olarak "Bugün" alt stringi döner.

Left(string, uzunluk)

Left fonksiyonu gösterilen bir uzunluk için belirtilen stringin sol taraftaki kısmını döndürür.

İki tane zorunlu argümanı vardır.

string argümanı ayırmak istediğiniz stringin adıdır.

uzunluk argümanı stringin sağ taraftan sol tarafa doğru kalması gereken karakterlerin sayısıdır.

```
<% =Left("abcde fghij klmno pqrst uvwxyz", 13) %>
```

Çıkış :

```
abcde fghij k
```

Len

String argümanının uzunluğunu döndürür. VBScript aynı tip string işlemlerine izin verir, fakat onun yerine iç fonksiyonları kullanır.

```
<%
Dim Tampon
Tampon = "vbscript"
%>
```

Stringin uzunluğunu almak için , Len( ) fonksiyonunu çağırın:

```
<%
uzunluk = Len(Tampon)
' uzunluk değeri 7 olacaktır.
%>
```

LoadPicture

Bir resim nesnesini döndürür ve yanlızca 32 bitlik platformlarda kullanılabilir. Komut bir resim dosyasını diskten yükler. Bu fonksiyon sunucu üzerinde çalışır, fakat herhangi bilinen özellikler ve metodları kabul etmez.

LoadPicture(resimadı)

Bitmap(.bmp) dosyaları , icon (.ico) dosyaları, run-length encoded (.rle) dosyaları, metadosya (.wmf) dosyaları , geliştirilmiş metadosyalar (.emf), GIF (.gif) dosyaları, be JPEG (.jpg) dosyaları dahil olmak üzere grafik formatları LoadPicture tarafından kabul edilir.

```
<% LoadPicture(ornek.gif) %>
```

Log

Bir sayının doğal logaritmasını döndürür.

Log(number)

Sayı argümanı 0 dan büyük herhangi bir geçerli sayısal ifade olabilir. Doğal logaritma e tabanına göre logaritmadır. e sabiti yaklaşık olarak 2.718282 dır.

Siz herhangi bir sayı x için n tabanına göre logaritmasını n nin doğal logaritması tarafından x in doğal logaritmasına bölerek hesaplayabilirsiniz.

Log*n(x)* = Log(*x*) / Log(*n*)

Aşağıdaki örnek 10 tabanına göre logaritma hesaplayan bir fonksiyonu gösteriyor.

```
Function Log10(X)
  Log10 = Log(X) / Log(10)
End Function
```

Ltrim

```
birString = "  Bu bir denemedir "
yeniString = Ltrim(birString)
' yeni yeniString = "Bu bir denemedir "
```

Örnekte görüldüğü gibi string içindeki değerin solundaki boşluk değeri atılarak yeni string elde edilmiştir.

Mid

Belirtilen bir indeksden başlayan bir string içindeki alt stringi ve belirtilen sayı ise uzunluk olarak karakterlerin sayısını döndürür.

```
Mid("Bugün düğün var",7,5)
```

Sonuç olarak "düğün" alt stringi döner.

Minute

Minute fonksiyonu bir argüman olarak Time fonksiyonunu kullandığımızda dakikayı döndürür. 0 ile 59 arasında bir değer alır.

```
<%=Minute(Time())%> '8 döner
```

Month

1 ile 12 arasında o yıla ait olan ayın numarasını döndürür.

```
Month(tarih)
```

tarih argümanı bir tarih ile gösterilen herhangi bir ifadedir. Eğer tarih Null içeriyorsa, Null döndürülür.

Aşağıdaki örnek güncel olan ay bilgisini getirir.

```
<%

Dim ay

ay = Month(Now) 'Güncel aya benzer bir sayı içerir.

%>
```

MonthName

Bir argüman olarak geçilen yılın ayına ait numaraya göre ismini döndürür.

```
MonthName(ay[, kısaltma])
```

Ay argümanı gerekli olup ayın sayısal karşılığıdır. Örneğin Mayıs 5 , Haziran 6 ve öyle devam eder. Kısaltma ise seçimlilik olup ay adının kısa şekilde gösterilip

gösterilmeyeceğini gösteren bir boolean değerdir. Eğer bu parametre yazılmassa, varsayılan olarak False değerini alır ve ay uzun adı ile yazılır.

Aşağıdaki örnekte ay adı kısa olarak elde edilmesi için kısaltma parametresi True olarak girilmiştir.

```
<%

Dim ay

ay = MonthName(10,True) 'ay "Oct" stringini içerir.

%>
```

MsgBox

Belirtilen mesaj, başlığı, ve ikonu veya düğmeleri içeren bir pencere mesaj kutusunu gösterir. Kullanıcının tıkladığı düğmeye göre sabit değer döndürülür. Bu istemci tarafı script için faydalıdır. Sunucu tarafında ise, VBScript MsgBox mesajlarını NT/2000 uygulama log dosyasına yazar.

```
MsgBox(prompt[,dugme][,baslik][,yardımdosyası,baglam])
```

Örneğin:

```
MsgBox "Bir mesaj." VbOKCancel + vbCritical "Mesaj başlığı"
```

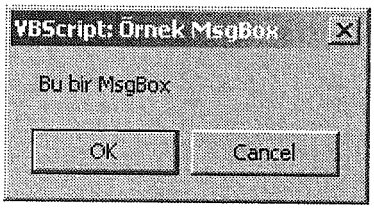
VBScript MsgBox Sabitleri

Sabit	Değer	Tanımı
vbOKOnly	0	Yanlızca OK düğmesi görünür.
vbOKCancel	1	OK ve Cancel düğmeleri görünür.
vbAbortRetryIgnore	2	Abort, Retry, ve Ignore düğmeleri görünür.
vbYesNoCancel	3	Yes, No, ve Cancel düğmeleri görünür.
vbYesNo	4	Yes ve No düğmeleri görünür.
vbRetryCancel	5	Retry ve Cancel düğmeleri görünür.
vbCritical	16	Kritik mesaj ikonu görünür.
vbQuestion	32	Uyarı Sorgu ikonu görünür.
vbExclamation	48	Uyarı mesaj ikonu.
vbInformation	64	Bilgi Mesaj ikonu görünür.
vbDefaultButton1	0	İlk düğme varsayılan.
vbDefaultButton2	256	İkinci düğme varsayılan.
vbDefaultButton3	512	Üçüncü düğme varsayılan.

vbDefaultButton4	768	Dördüncü düğme varsayılan.
vbApplicationModal	0	Kullanıcı o anki uygulamaya çalışmaya devam etmeden önce mesaj kutusuna cevap vermek zorunda.
vbSystemModal	4096	Win 16 sistemlerinde kullanıcı mesaj kutusuna cevap verene kadar tüm uygulamalar durdurulur. Win 32 sistemlerinde, bu sabit sadece mesaj için bekleyen uygulama durdurulur fakat diğer uygulamalar normal çalışmasına devam eder.

Bir MsgBox JavaScript alert() fonksiyonuna benzer. İşte örnek bir kod.

```
<SCRIPT LANGUAGE= "VBScript">

<!--

        MsgBox"Bu bir MsgBox", vbOKCancel , "Örnek MsgBox"

//-->

</SCRIPT>
```



İlk argüman sizin yazacağınız text bilgidir, ikinci argüman gösterilecek düğmeleri belirleyen seçimlilik bir argümandır. En son argüman ise pencere başlığında gösterilecek olan text bilgiyi gösterir.

Now

Now o anki güncel tarih ve zamanı bir Date değişkeni içinde döndürür.

Örneğin:

```
<%

Response.write Now()
```

```
'Güncel tarih ve zamanı yazar

%>
```

Now() fonksiyonuna yakın bir fonksiyon olan Date() zaman bilgisi olmaksızın sadece tarih bilgisini döndürür.

Oct

Bir sayının sekizli değerini gösteren bir string döndürür.

```
Oct(sayı)
```

Sayı argümanı herhangi geçerli bir ifadedir. Eğer sayı tam bir sayı değilse, değerlendirilmeden önce en yakın tam bir sayıya yuvarlanır.

Eğer sayı ... ise	Oct' nin döndürdüğü sonuç
Null	Null.
Empty	Sıfır (0).
Herhangi bir başka bir sayı	11 sekizli karaktere kadar

Siz sekizli tabandaki sayıları &O ile doğrudan uygun bölgedeki sayıları gösterebilirsiniz. Örneğin, &O10, 8 in sekizli tabanda gösterim şeklidir.

Aşağıdaki örnek bir sayının sekizli değerini döndürmek için Oct fonksiyonunu kullanıyor.

```
<%

Dim sekiz

sekiz = Oct(5)    ' 5 döner.

sekiz = Oct(8)    ' 10 döner.

sekiz = Oct(459)  ' 713 döner.

%>
```

Replace

Farklı bir alt string ile bir string içinde belirtilen bir veya daha çok alt stringlerin yerdeğiştirmesini sağlar. Yerdeğiştiren alt stringler orijinal alt string ile aynı uzunlukta olmalıdır.

```
<%

Replace("Bu harika bir site# ya sana göre#","#","!")

%>
```

Sonuç olarak "Bu harika bir site! ya sana göre!" stringi döner. Burada "#" karakteri ile "!" karakteri yerdeğiştirmiş olduğuna dikkat edin.

RGB

Üç bireysel renk değerinden oluşan bir kümeyi RGB formatında tek bir Long renk değerine çevirir.

```
RGB(kırmızı, yesil, mavi)
```

kırmızı argümanı gerekli olup kırmızı renk bileşeni için 0 ile 255 arasında bir sayıdır.
yesil argümanı gerekli olup yeşil renk bileşeni için 0 ile 255 arasında bir sayıdır.
mavi argümanı gerekli olup mavi renk bileşeni için 0 ile 255 arasında bir sayıdır.

```
Function tersiRGB(kırmızı,yesil,mavi)

  tersiRGB= CLng(kırmızı +(yesil * 256) +(kırmızı * 65536))

End Function
```

Veya

RGB ye herhangi bir argüman için değer 255 değerini geçemez. Eğer tamsayı argüman 255 değerini geçerse, o zaman sanki 255 değerindeymiş gibi davranır. 256*256*256 = 16,777,215 farklı renk kombinasyonu oluşturulabilir.

```
<%; = RGB(0, 0, 0) %>

<%; = RGB(94, 71, 177) %>

<%; = RGB(255, 255, 255) %>
```

Çıkışlar:

```
0

11618142

16777215
```

Right

Bir stringin sağ-el tarafından belirtilen karakterlerin sayısını döndürür.

```
Right("Kasabanın sırrı",7 )
```

Sonuç olarak "n sırrı" alt stringi döner.

Rnd

0 ve 1 arasında rasgele bir sayı döndürür.

```
<%
RANDOMIZE
kucukSayi = 1
buyukSayi = 10
rasgeleSayi = Int((buyukSayi-kucukSayi+1)*Rnd+kucukSayi)
Response.write  rasgeleSayi
' Sonuçta 0,1,2,3,4,5,6,7,8,9,10 rasgele sayılar üretilir.
%>
```

Rasgele bir sayı getirmek için Randomize ve Rnd fonksiyonlarını kullanırız.

```
<SCRIPT LANGUAGE="VBScript">
<!--
Option Explicit
Dim rasgele
Randomize
rasgele = (CInt(Rnd * 6 )+1)
Document.Write (rasgele)
//-->
</SCRIPT>
```

Bu örnekte biz 1 ile 7 arasında tüm rasgele sayıları alabileceğiz.

Round

Kayan-noktalı sayıları kesirli kısmı olmayan onlu yerleşimine göre belirli bir sayıya yuvarlar.

```
<%

Dim x

Dim y

x = Round( 10.42 )

y = Round( 19.7 )

Response.write x

' Sonuç olarak x değeri 10 olacaktır.

' Sonuç olarak y değeri 20 olacaktır.

%>
```

Rtrim

```
birString =" Bu bir denemedir "

yeniString =Rtrim(birString)

' yeniString =" Bu bir denemedir"
```

Örnekte görüldüğü gibi string içindeki değerin sağındaki boşluk değeri atılarak yeni string elde edilmiştir.

ScriptEngine

O anki yürüten script motorunun adını içeren bir stringi döndürür.

```
ScriptEngine

<% =ScriptEngine %>

Çıkış  VBScript
```

ScriptEngineBuildVersion

O anki yürüten script motorunun versiyon bilgisini içeren bir stringi döndürür.

ScriptEngineBuildVersion

<% =ScriptEngineBuildVersion %>

Çıkış 2926

ScriptEngineMajorVersion

O anki yürüten script motorunun major versiyon bilgisini içeren bir stringi döndürür.

ScriptEngineMajorVersion

<% =ScriptEngineMajorVersion %>

Çıkış 4

ScriptEngineMinorVersion

O anki yürüten script motorunun minor versiyon bilgisini içeren bir stringi döndürür.

ScriptEngineMinorVersion

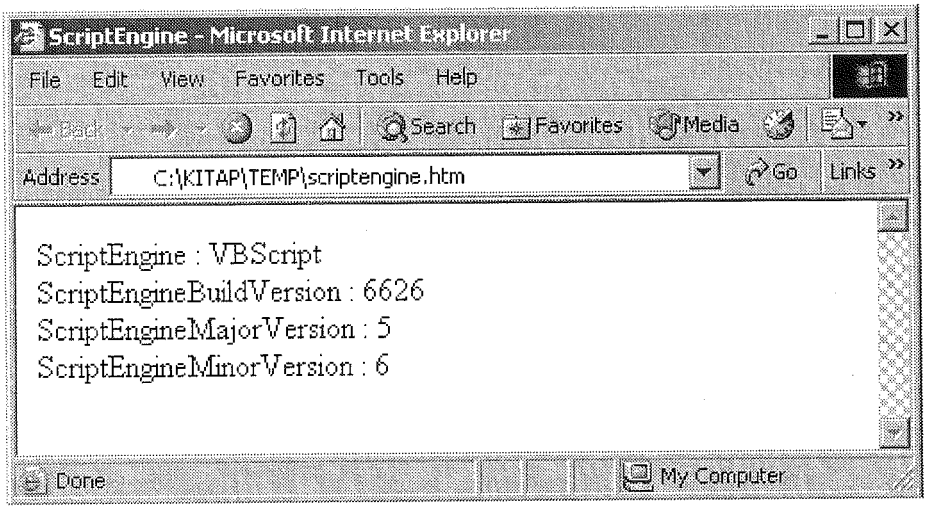
<% =ScriptEngineMinorVersion %>

Çıkış 0

ScriptEngine ile ilgili örnek program kodu

```
<HTML>
<HEAD>
<TITLE>ScriptEngine</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=">
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<SCRIPT LANGUAGE="VBScript">
```

```
Document.Write "ScriptEngine          : " & ScriptEngine & "<BR>"

Document.Write "ScriptEngineBuildVersion : " & ScriptEngineBuildVersion &
               "<BR>"

Document.Write "ScriptEngineMajorVersion : " & ScriptEngineMajorVersion &
               "<BR>"

Document.Write "ScriptEngineMinorVersion : " & ScriptEngineMinorVersion &
               "<BR>"
</SCRIPT>

</BODY>

</HTML>
```



Second

Second fonksiyonu bir argüman olarak Time fonksiyonunu kullandığımızda saniyeyi döndürür. 0 ile 59 arasında bir değer alır.

```
Second(Time)

<%=Second(Time())%> '  56 değeri döner
```

SetLocale

Script bağlamı için LocaleID yi ayarlar. Siz bunu atanılan LocaleID için olan formattaki çıkış tarihleri, zamanları ve currency değerleri için kullanırsınız.

SetLocale(LCID)

Zorunlu bir tane argümanı vardır , LCID coğrafik yerleri belirten kısa bir string, onlu , veya onaltılık (hex) değerler olabilir. Değişik coğrafik yerler kullanıcı dilini, ülkesini, ve kültürünü gösterir. Örneğin, "English-United States" "en-us" veya "1033", veya "0x0409" şeklinde olur.

Eğer LCID sıfır olarak ayarlanırsa, yerel bilgiler sistem tarafından ayarlanır.

<% =SetLocale(0) %>

Çıkış 1033 olur

Sgn

Bir sayının işaretini döndürür.

Sgn (sayi)

-1 negatif sayıları gösterir.
0 sıfır sayısını gösterir.
1 pozitif sayıları gösterir.

<% =Sgn(-127.89) %>

Çıkış -1

<% =Sgn(0) %>

Çıkış 0

<% =Sgn(127.89) %>

Çıkış 1

Sin

Bir açının sinüsüni döndürür.

Sin (sayi)

<% =Sin(45.0) %>

Çıkış 0.850903524534118

negatif bir sayıyıda açı olarak kullanabilirsiniz.

<% =Sin(-45.0) %>

Çıkış -0.850903524534118

Space

Uzunluk olarak karakterlerin belirli bir sayısı kadar boşluğu string içine doldurulur.

Space (sayi)

HTML aynı satırda sadece bir boşluk gösterir.

<% ="üst boşluk." %>

<% =Space(13) %>

<% ="alt boşluk." %>

Çıkış

üst boşluk.

alt boşluk.

Split

Belirli bir ayıraca bağlı olarak bir stringi bir dizinin alt stringlerine ayırır.

Split (ifade, ayirici, sayi, karsilastir)

Sadece bir tane zorunlu argümanı vardır.

ifade argümanı bir string ifadedir.

<% metin = "Kontrol ve Bilgisayar Mühendisliği" %>

<% metin= Split(metin) %>

<% =dizi(0) %>

<% =dizi(1) %>

<% =dizi(2) %>

```
<% =dizi(3) %>
```

Çıkış :

```
Kontrol
ve
Bilgisayar
Mühendisliği
```

Üç tane seçimlilik argüman vardır.

ayirici argümanı ifadeyi ayırmak kullanmak istediğiniz (boşluklar dahil) karakterleri belirtir. Default olarak tek bir boşluk " " içerir.

```
<% metin = "How now brown cow?" %>

<% metin = Split(mystring, "ow") %>

<% =dizi(0) %>

<% =dizi(1) %>

<% =dizi(2) %>

<% =dizi(3) %>

<% =dizi(4) %>
```

Çıkış :

```
H
n
br
n c
?
```

sayi argümanı döndürülecek elemanların sayısını belirtir.

```
<% metin = "How now brown cow?" %>

<% metin = Split(metin, " ", 2) %>

<% =dizi(0) %>

<% =dizi(1) %>
```

Çıkış

```
How
now brown cow?
```

karsilastir argümanı yanlızca sabitleri veya karşılaştırma sabitlerini kullanırlar.

Sabit	Değer	Tanımı
VBBinaryCompare	0	İkili karşılaştırma
VBTextCompare	1	Text karşılaştırma
VBDataBaseCompare	2	Veritabanının iç kısımdaki bilgiyi karşılaştırma

Örnekte, karsilastir argümanı için VBTextCompare kullanarak, ayırma işlemi b de olur ve ayırıcı "B" argümanı arasındaki fark büyük harf durumu göz ardı edilir.

```
<% metin = "How now brown cow?" %>

<% metin = Split(metin , "B", 2, 1) %>

<% =dizi(0) %>

<% =dizi(1) %>
```

Çıkış

```
How now
rown cow?
```

```
<% metin = "How now brown cow?" %>

<% dizi = Split(mystring, "B", 2, VBTextCompare) %>

<% =dizi(0) %>

<% =dizi(1) %>
```

Çıkış

```
How now
rown cow?
```

Sqr

Bir sayının karesini hesaplamamızı sağlayan bir fonksiyondur.

```
<%

Dim sayi
```

```
Dim sonuc

sayi= 25

sonuc = Sqr ( sayi )' sonuc = 625 dir.

%>
```

StrComp

İki stringi karşılaştırmak için kullanılır. Eğer ilk string ikinci stringden daha düşük ise –1 değerini döndürür. Eğer eşit ise 0 değerini döndürür, ve ilk string ikincisinden daha büyük ise 1 değerini döndürür. Siz karşılaştırma işleminin küçük-büyük harflere karşı duyarlı olup olmamasını seçebilirsiniz.

StrComp kullanım şekli StrComp(string1, string2) dir ve eğer onlar aynı ise sonuç 0 olacaktır yoksa 1. StrComp büyük-küçük harf ayırımına karşı duyarlıdır, fakat siz bunu parametre ile değiştirebilirsiniz: StrComp(string1, string2, 1)

İki stringi karşılaştırmak için, siz "= " operatörünü kullanabilirsiniz, eğer karşılaştırmada büyük-küçük harf ayırımı yapılıyorsa o zaman "kara" = "KARA" karşılaştırma sonucu False olacaktır, fakat UCase("kara") = UCase("KARA") karşılaştırma sonucu True olacaktır.

```
StrComp( "kara", "KARA", 1 ) 'sonuç 0 olacaktır.
```

String

Çok kez tekrarlanmış bir karakter ile doldurulmuş bir string döndürür.

```
String ( sayi, karakter )
```

İki argümanıda girmek zorunludur.

sayi argümanı bir tamsayı olup kaç kez tekrarlanacağını belirtir.

karakter argümanı tek bir karakter olup çift tırnak arasında olmak zorundadır.

```
<% =String(33, "!") %>
```

Çıkış :

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

StrReverse

Verilen bir stringin karakterleri tersine çevrilmiş bir string olarak döndürür.

StrReverse("Mutlu Günler Sana")

Sonuç string "anaS relnüG ultuM" şeklinde olacaktır.

Aşağıdaki örnek StrReverse fonksiyonunu kullanarak bir stringin tersinden nasıl yazdırılacağını göreceksiniz.

```
<SCRIPT LANGUAGE="VBScript">
<!--
Option Explicit
Dim strText
strText = "Bu bir deneme"
document.write (strReverse(strText))
//-->
</SCRIPT>
```

Çıktısı: emened rib uB

Tan

Bir açının tanjantını döndürür.

Tan (sayi)

Tan fonksiyonu bir sayının (açı) tanjantını döndürür.

```
<% =Tan(45.0) %>
```

Çıkış 1.61977519054386

sayi olarak negatif bir değerde kullanabilirsiniz.

```
<% =Tan(-45.0) %>
```

Çıkış -1.61977519054386

Time

Time fonksiyonu bilgisayarınızda tanımlı olan o anki zamanı döndürür.

<%=Time %>

Çıktısı 2:07:56 PM olur

Aşağıdaki örnekte VBScript ile o anki tarih ve zamanı gösteren fonksiyonlar kullanılıyor.

```
<SCRIPT LANGUAGE="VBScript">
<!--
Document.write ("Tarih: " & Date())
Document.write ("<br/>")
Document.write ("Zaman: " & Time())
//-->
</SCRIPT>
```

Tarih: 1/29/2002

Zaman: 12:13:32 PM

Timer

Timer fonksiyonu (12:00:00 AM) geceyarısından bu yana geçen saniyelerin miktarını döndürür.

Timer

<% =Now %>

<% =Timer %>

Çıkış :

12/17/99 2:14:36 PM

51276.99

TimeSerial

TimeSerial fonksiyonu argümanları alt tip olan Date tipinin bir varyasyonuna dönüştürür.

TimeSerial(saat, dakika, saniye)

Üç zorunlu argümanı vardır. Saat argümanı bir string veya tamsayı olarak saat bilgisi, dakika argümanı bir string veya tamsayı olarak dakika bilgisi,ve saniye argümanı bir string veya tamsayı olarak saniye bilgisidir.

```
<%
Response.Write TimeSerial(12, 11, 10) '12:11:10 PM olarak döner
%>
```

Aşağıdaki örnek mutlak zaman sayılarının yerine ifadeler kullanıyor.

```
<%
Dim zaman
zaman = TimeSerial(12 - 6, -15, 0)
Response.Write zaman  ' Returns 5:45:00 AM.
%>
```

TimeValue

Argümandan zamanı döndürür. Eğer argüman hem tarih hemde zaman bilgisini taşıyorsa, TimeValue fonksiyonu yanlızca zamanı döndürür.

TimeValue(String)

String argümanı geçerli tarih formatında bir string olmalı yoksa bir hata mesajı alırsınız. String argümanı 0:00:00 (12:00:00 AM) den 23:59:59 (11:59:59 PM) e kadar içerebilir. Bununla birlikte, string argümanı ilgili bölgedeki bir zamanı gösteren bir ifade olabilir. Eğer string Null ise, Null döndürülür.

Aşağıdaki örnek TimeValue fonksiyonunu bir stringi bir zamana dönüştürmek için kullanır.

```
<%
Dim zaman
zaman = TimeValue("4:35:17 PM")
Response.Write zaman
' zaman 4:35:17 PM içerir.
%>
```

Trim

```
<%
birString = " Bu bir denemedir "
yeniString = Trim(birString)
' yeniString = "Bu bir denemedir"
%>
```

Örnekte görüldüğü gibi string içindeki değerin solundaki ve sağındaki boşluklar atılarak yeni string elde edilmiştir.

Ubound

Bir dizinin üst-sınırını döndürür.

```
Dim renkler(1 to 10 )
```

Burada 10 elemanlı bir dizi var ve ilk indisi 1 den (0 değil) başlamakta olup son indiside 10 dur.

Geçerli dizi indis değerleri bölgesi dizinin sınırlarıdır. Siz alt sınır ve üst sınır değerlerini LBound(DiziDegiskeni) ve UBound(DiziDegiskeni) fonksiyonları ile bulabilirsiniz.

Örneğin:

```
<%

Dim Renk( 1 to 8 )

Response.write "Alt sınır değeri=" & LBound(Renk)&"<BR>"

Response.write "Üst sınır değeri=" & UBound(Renk)&"<BR>"

%>
```

UCase

Verilen bir stringi tüm karakterleri büyük harflere çevrilmiş bir string halinde döndürür.

```
UCase("Kasabanın Sırrı") 'Sonuç "KASABANIN SIRRI" olacaktır.
```

Bu örnekte Lcase ve Ucase fonksiyonları sırası ile bir text stringi önce küçük harflere ve sonra büyük harflere çeviriyor.

```
<SCRIPT LANGUAGE ="VBScript">
<!--

Dim metin

metin = "VBScript programlama"

' orijinal text bilgiyi göster

Document.Write ("<br/>")

Document.Write ("Orjinal text : " & metin)

' küçük harflere çevir

Document.Write ("<br/>")

Document.Write ("Küçük harfli text : " & LCase(metin))

' büyük harflere çevir

Document.Write ("<br/>")

Document.Write ("Büyük harfli text : " & UCase(metin))

//-->
</SCRIPT>
```

Çıktısı

Orjinal text : VBScript programlama

Küçük harfli text : vbscript programlama

Büyük harfli text : VBSCRIPT PROGRAMLAMA

VarType

Bir sabit veya bir değişkenin VBScript iç tipini gösteren sabitlerin kombinasyonunu (diziler için) döndürür. Bir variant değişkenin alt tipini belirlemenin tek yolu VarType fonksiyonunu kullanmak ve bu sabit değerlerden biri için test etmektir. Belirli bir değişken tipi dizileri için , vbArray sabiti bir boolean ve koşul kullanarak test eder. Örneğin, stringleri içeren bir variant dizi olup olmadığını test etmek için:

```
<%
Dim d
d =Array("1", "2", "3", "4" )          'd bir dizi değişkeni.
If ((varType(d)and vbArray)=vbArray and (varType(d) and vbString)=vbString
Then Response.write "Bir string dizi"
End If
%>
```

VarType fonksiyonu sabitleri

Sabit	Değer	Tanımı
vbEmpty	0	İlk değer atanmamış
vbNull	1	Geçersiz bir değer taşır
vbInteger	2	Integer alt tipi
vbLong	3	Long alt tipi
vbSingle	4	Single alt tipi
vbDouble	5	Double alt tipi
vbCurrency	6	Currency alt tipi
vbDate	7	Date alt tipi
vbString	8	String alt tipi

vbObject	9	Nesne
vbError	10	Error alt tipi
vbBoolean	11	Boolean alt tipi
vbVariant	12	Variant(yanlızca variant dizileri için kullanılır)
vbDataObject	13	Veri erişim nesnesi
vbDecimal	14	Onlu alt tipi
vbByte	17	Byte alt tipi
vbArray	8192	Dizi

WeekDay

Bir Date argümanını alır ve argümanın gün kısmına göre haftanın gününü 1 den 7 ye kadar olan bir sayı döndürür.

WeekDay(Tarih)

Çıkış Tarih ve zaman sabitleri tarafından tanımlanır. Bir argümanı vardır. Tarih argümanı geçerli bir tarih ifadesi ve siz Date ve Now fonksiyonlarınıda kullanabilirsiniz.

```
<%
Response.write "Haftanın günü = " & WeekDay(Now())
' Haftanın günü= 1 olur
%>
```

WeekDayName

WeekdayName fonksiyonu haftanın gününün adını tam olarak döndürür.

WeekdayName(haftagunu, kisaltma, haftaninilkgunu)

Zorunlu bir tane argümanı vardır.

haftagunu argümanı haftanın gününü belirten sayıdır, bu sayılar izleyen tabloda verilmiştir.

Tarih ve Zaman Sabitleri

Sabit	Değer	Tanımı
vbSunday	1	Pazar
vbMonday	2	Pazartesi
vbTuesday	3	Salı
vbWednesday	4	Çarşamba
vbThursday	5	Perşembe
vbFriday	6	Cuma
vbSaturday	7	Cumartesi
vbUseSystemDayOfWeek	0	Haftanın ilk günü için sistem ayarlarınızda belirtilen haftanın gününü belirtir.
vbFirstJan1	1	Ocak 1 deki hafta için kullanın.
vbFirstFourDays	2	Yeni yıldaki en az dört güne sahip ilk haftası için kullanın.
vbFirstFullWeek	3	Yılın tam ilk haftası için kullanın.

```
<% =WeekdayName(7) %>
```

Çıkış :

Cumartesi

İki tane seçimlilik argümanı vardır.

kisaltma argümanı ilk üç karakteri ile kısaltılmış olan gün adının döndürülmesi için ya true yada false değerini alır. Eğer true ise ad kısaltılır, eğer false ise ad kısaltılmayacaktır.

```
<% =WeekdayName(7, True) %>
```

Çıkış :

Cmt

haftaningunu argümanı yanlızca sabitleri veya tabloda tanımlanmış olan değerleri alabilirler.

Bu örnekte, Pazartesi haftanın ilk günü olarak tanımlıdır. Bu yüzden, Cumartesi 6 olacaktır.

```
<% =WeekdayName(6, False, 2) %>
```

Çıkış :

Cumartesi

<% =WeekdayName(6, False, VBMonday) %>

Çıkış :

Cumartesi

Year

Bir tarih bilgisini alır ve argümanın yıl kısmına bağlı olarak bir tamsayı döndürür.

Year(tarih)

tarih argümanı bir tarihi gösteren herhangi bir ifadedir. Eğer tarih Null içeriyorsa, Null döndürülür.

Aşağıda tarih bilgisinden yıl bilgisinin nasıl alındığını gösteren örnek verilmiştir.

```
Dim tarih, yıl

tarih = #Ocak 20,1969# 'Bir tarih ataması yapılıyor.

yıl = Year(tarih)      'yıl 1969 değerini alır.
```

BÖLÜM
ON ÜÇ

13

VBScript Nesne Modeli

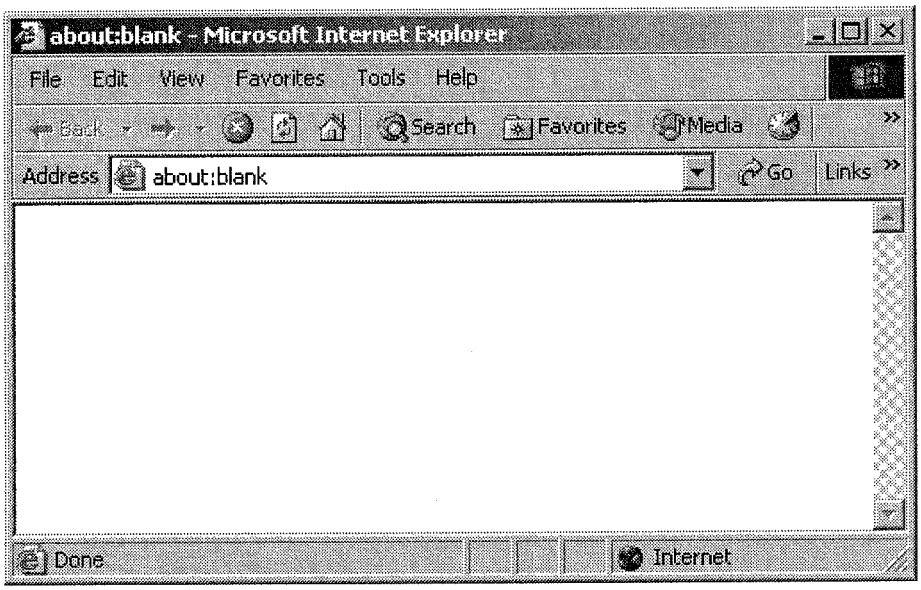
- Microsoft Internet Explorer Nesne Modeli
- Tarayıcı Nesneleri
- Window
- Document
- History
- Location
- Link
- Anchor
- Navigator
- Form
- Elements
- Çerçeveler,Formlar ve Elemanlar

VBScript Nesne Modeli

Bu bölümde , Internet Explorer Script yazma ile ilgili nesne modelini göreceğiz. Bu şekilde VBScript standard HTML bileşenleri (örneğin, formlar, alanlar, çerçeveler, ve dökümanlar gibi) ile etkileşim içine girer.

Microsoft Internet Explorer Nesne Modeli

VBScript dilini kullandığınız için, Netscape ve Internet Explorer kullanıcıları için ayrı bir versiyona sahip olduğunuz anlamına gelir. VBScript dilini sadece temel olmayan sayfa özellikleri veya Internet Explorer kullanan kullanıcılar için kullanınız. VBScript dilinin yeteneklerini diğerlerinden ayıran örnekler ileride verilecektir ve eğer bir istemci Internet Explorer kullanıyorsa o zaman ekstra özellikleri çağıran bir script dili olacaktır.



Microsoft Internet Explorer Tarayıcı

JavaScript dilinin hem Netscape hemde Internet Explorer içinde kullanılabileceğini unutmayınız.

Eğer bu dört kategoriden birine uyarlayabilirseniz, VBScript ve diğer IE-özel özellikler, örneğin, kayan çerçeveler, ActiveX kontroller, IE Dinamik HTML gibi, kullanabilirsiniz. Burada, herkesin Internet Explorer kullandığını varsayarak görüntüleme yapılacağını varsayıyorum ve bundan dolayı kayan çerçeveleri kullanmak hedefimize ulaşmak için uygun bir yol olacaktır.

Çok sayıda kod VBScript içindedir. Kodun birkaç tipi daima VBScript ifadeleri, operatörleri, ve iç fonksiyonlar olarak mevcuttur. Bunlara ek olarak, web sayfalarındaki VBScript de tarayıcı nesne modeline erişime sahiptir. Bu tarayıcının nasıl çalıştığına bakışın yapısal bir yoludur. Bu özelliklere, olaylara ve metodlara erişim verir. Olaylar size neler olduğu hakkında bilgi verir, örneğin, fare tıklaması gibi, ve bu olaya cevap vermenize izin verir. Özellikler arka plan rengi gibi bilgileri saklar, ve sık sık değişebilir. Değiştirilemeyen özellikler yanlızca okunabilir özellikler olarak adlandırılır. Metodlar nesnelerin yapabildikleri şeylerdir. Metodlar için örnek olarak bir mesajı gösteren Alert metodunu ve bir dökümana bilgi yazan Write metodunu gösterebiliriz.

Internet Explorer 3 ve daha üst versiyonları VBScript dilini destekler, fakat herbir versiyon farklı bir nesne modeline sahiptir. Bir üst versiyon bir önceki versiyonun nesne modelinden daha geniş bir nesne modeline sahiptir. En son versiyon yeteneklerinin hepsini kullanmak bir önceki versiyona göre daha çekicidir, fakat genellikle mümkün olduğunca eski tarayıcıları kullanan kullanıcılarıda göz önünde tutarak yazmak en iyisidir.

İşte basitleştirilmiş IE nesne modeli, fakat gerçekte çok daha karmaşıktır.

* Window
    * Navigator
    * Location
    * History
    * Frames...
    * Document
        * Links...
        * Anchors...
        * Forms...
            * Elements...

(. . .) o nesneden daha çok nesne olduğunu gösterir. Herbir nesne bir Window nesnesi gibi davranır, ve başka pencerelerde içerebilir. Nesneler nokta notasyonu kullanılarak işaret ettirilir, yani bir nokta ile nesneler ayrılır. Örneğin, Alert metoduna işaret etmek için, Window.Alert şeklinde kullanabilirsiniz. Dökümanın arka plan rengine işaret etmek için, siz Window.Document.BgColor şeklinde kullanabilirsiniz. Çoğu durumlarda, Window nesnesi atılır. Eğer window nesnesi atılmış ise, bu tarayıcı tarafından anlaşılır, çünkü Window en üst seviye nesnedir. Bundan dolayı, önceki örneklerde sadece Alert, ve Document.BgColor şeklinde yazabilirdiniz.

Tarayıcı Nesneleri

Tarayıcı on tane ana nesneye sahiptir.

- Window
- Document
- History
- Navigator
- Location
- Link
- Anchor
- Form
- Element

Burada her bir nesne detaylı bir şekilde incelenmekten ziyade, herbir nesne ve bu nesnelere ait olan metod ve özellikleri hakkında kısa bilgi vereceğim.

Window

VBScript ve JavaScript içindeki bir window fiziksel bir tarayıcı penceresini gösterir, ve doküman nesnesi için bir taşıyıcı rolü oynar. Pencereler script yazılarak açılabilir veya kapatılabilir. Buna ek olarak, bir window giriş ve çıkış mekanizmaları baz alınarak bir diyalog sağlanabilir.

Özellikler

document

Kullanıcı tarafından gösterilen güncel doküman. Doküman özelliği pencere içindeki beyaz alanı ve üzerindeki elemanları içerir. Bu elemanlar bir text, resim, veya diğer nesneler olabilir.

Window.**Document**.Write "document özelliği Window nensesine aittir."

Veya

Document.Write "document özelliği Window nensesine aittir."

Örnektede görüldüğü gibi Window nesnesine ait olan hem özellikler hemde metodlardan önce Window yazmayabiliriz, bu bir hata değildir.

Metodlar

alert (string)

Bir string başlık ile bir uyarı diyalog kutusunu gösterir.

Örnekte, form içinde tanımlanan dugme1 form elemanı üzerine tıklandığı zaman click olayı meydana gelir, bu olay sonucu olarak ekranda istenen mesaj alert metodu ile gösterilir.

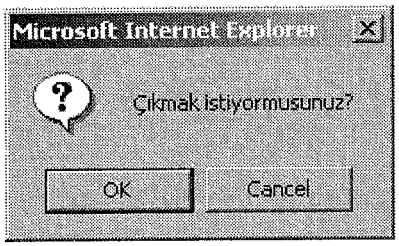
```
<SCRIPT TYPE="text/vbscript" LANGUAGE="VBScript">
<!- -
Sub dugme1_OnClick()
   alert "Bu bir mesaj.", vbExclamation, "VBScript Testi"
   ' veya Window.Alert "Bu bir mesaj.", vbExclamation, "VBScript Testi"
End Sub
- ->
</SCRIPT>
```

confirm (string)

Bir string başlık ile bir OK/Cancel onayı için kullanıcıya sunar. True veya False değerlerinden birini döndürür.

```
<HTML>
<HEAD>
<TITLE>Confirm metodu kullanımı</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=">
</HEAD>

<BODY BGCOLOR="#FFFFFF"  TEXT="#000000">
<SCRIPT LANGUAGE="VBScript">
```

```
sonuc = confirm("Çıkmak istiyormusunuz?")
Document.Write sonuc
</SCRIPT>
</BODY>
</HTML>
```



prompt (string)

Bir string başlık ile bir text cevabı girilmesi için kullanıcıya sunar. Bir string döndürür.

Örneğin,

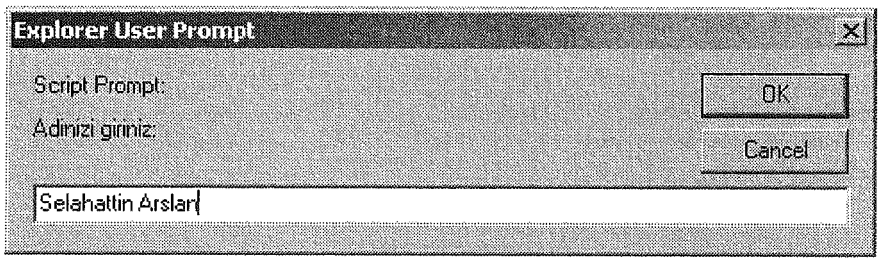
```
<SCRIPT LANGUAGE="VBScript">
' Değişken tanımlama
Dim str
' str değişkenine bir değer atama
str = Prompt ("Lütfen bir string girin", "(Buraya Yazınız)")
Document.Write "String " & len(str) & " karakter uzunluğunda"
</SCRIPT>
```

Başka bir örnek

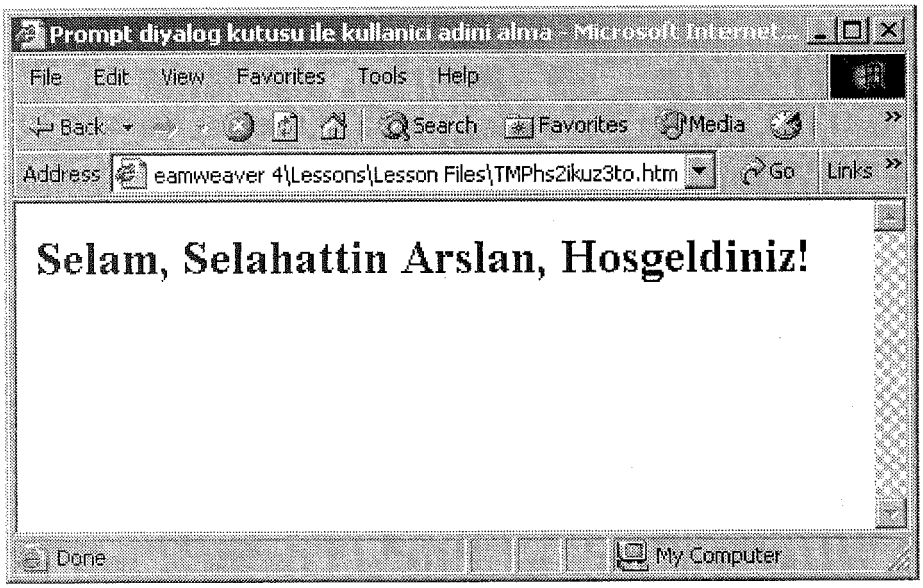
```
<HTML>
<HEAD>
```

```
<TITLE>Prompt diyalog kutusu ile kullanıcı adını alma</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<SCRIPT LANGUAGE="VBSCRIPT">
<!--
Dim adi
adi = ""
Do While adi = ""
    adi = Prompt("Adınızı giriniz:", "")
Loop
    Document.Write "<H2>Selam, " & adi & ", " & "Hoşgeldiniz!</H2>"
-->
</SCRIPT>
</BODY>
</HTML>
```

HTML programını tarayıcı üzerinde çalıştırdığınızda karşınıza aşağıdaki prompt gelecektir.



Promt penceresi üzerindeki OK düğmesine tıkladığınız zaman aşağıdaki görünümü elde edersiniz.



window open (url, pencer_adi)

Verilen lokasyonda ve belirtilen bir pencere adı ile yeni bir pencere açar.

```
<SCRIPT LANGUAGE="VBScript">
pencere = Window.Open("http://www.abc.com","yenipen")
</SCRIPT>
```

close ()

Bir pencereyi kapatır.

```
<SCRIPT LANGUAGE="VBScript">
Window.Close()
</SCRIPT>
```

o anki güncel pencereyi kapatır.

navigate (url)

Belirli bir URL için pencereyi açar

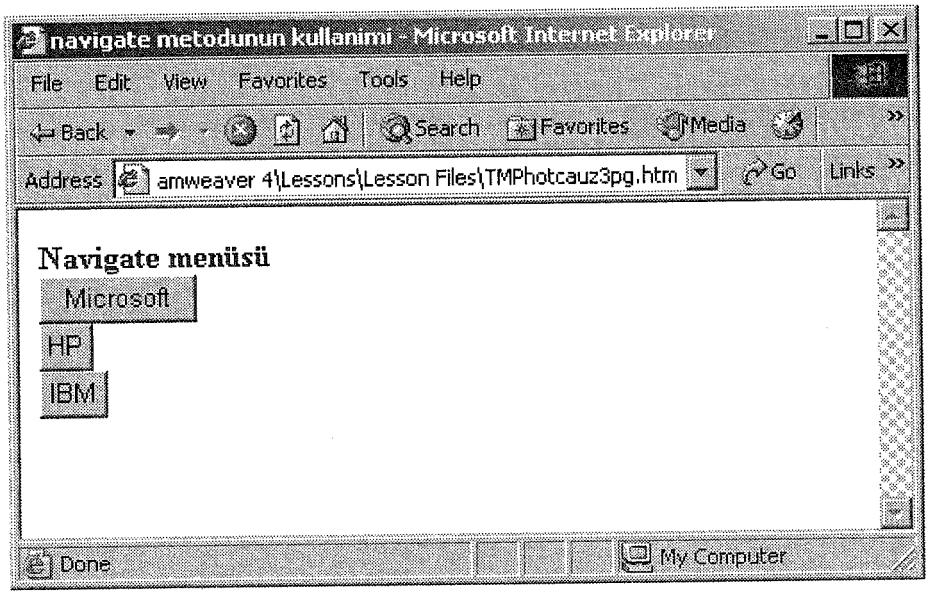
```
<HTML>
<HEAD>
<TITLE>navigate metodunun kullanımı</TITLE>
</HEAD>
<SCRIPT LANGUAGE="VBSCRIPT">
<!--
        Sub dug1_onClick
                navigate("http://www.msn.com")
        End Sub
        Sub dug2_onClick
                navigate("http://www.hp.com")
        End Sub
        Sub dug3_onClick
                navigate("http://www.ibm.com")
        End Sub
-->
</SCRIPT>
<BODY BGCOLOR="#FFFFFF">
<FORM><B>Navigate menüsü</B>
<BR>
<INPUT TYPE=BUTTON ID="dug1" VALUE="Microsoft">
<BR>
<INPUT TYPE=BUTTON ID="dug2" VALUE="HP.">
<BR>
```

<INPUT TYPE=BUTTON ID="dug3" VALUE="IBM">

</FORM>

</BODY>

</HTML>



Document

Bir doküman bir HTML dökümanını gösterir ve belirli bir pencere ile ilişkilendirilmiştir. Dökümanlar formlar, hiperbağlar, resimler, appletler, ActiveX kontrolleri, ve dahasını taşır. Bunların herbiri dönüşümlü olarak yazmış olduğumuz scritp kodu tarafından erişilir.

Özellikler

title

Dökümanın takıları arasında taşınan text string. Bu text bilgi tarayıcı penceresinin başlık kısmında görünür.

<HTML>

<HEAD>

```
<TITLE>Döküman title özelliğinin kullanımı</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<SCRIPT LANGUAGE="VBScript">
<!--
    Document.Write "<P>Doküman başlığı <I>" & Document.Title & "</I>"
-->
</SCRIPT>
</BODY>
</HTML>
```

location

Dökümanın location nesnesi.

Aşağıdaki örnek dökümanın lokasyon bilgisini verir.

```
<SCRIPT LANGUAGE = "VBScript">
        lokasyon = Document.Location
        MsgBox lokasyon
</SCRIPT>
```

BgColor, LinkColor, vLinkColor, aLinkColor, FgColor

Dökümanın renk şeması için Text veya RGB değerleridir.

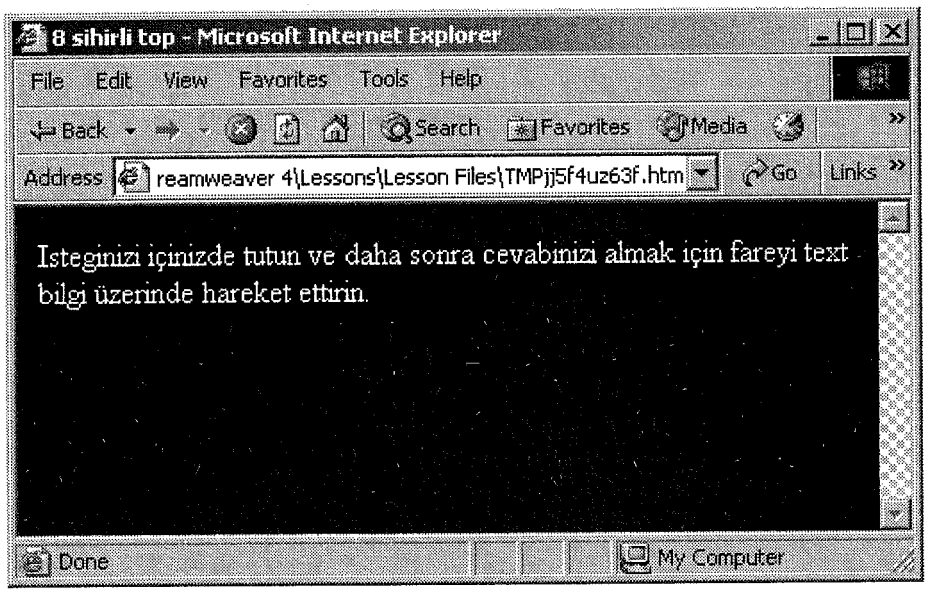
Aşağıdaki örnek dökümanın bağ rengini beyaz renk yapar.

```
<SCRIPT language = "VBScript">
        Document.LinkColor = "White"
</SCRIPT>
```

Başka bir örnek

```
<HTML>
<HEAD>
<TITLE>8 sihirli top</TITLE>
</HEAD>
<STYLE>
<!--
        A { text-decoration: none; }
        A:hover { text-decoration: underline; color: none; }
-->
</STYLE>
<SCRIPT LANGUAGE="VBSCRIPT">
<!--
Sub Renkler_onMouseOver
        top.document.fgColor = "WHITE"
        top.document.linkColor = "BLACK"
        top.document.vlinkColor = "BLACK"
        top.document.alinkColor = "BLACK"
End Sub
Function SihirliSekizTop()
        Dim tahmin(7), numara
        Randomize
        tahmin(0) = "Evet, kesinlikle"
        tahmin(1) = "Olabilir"
        tahmin(2) = "Daha sonra tekrar deneyin"
```

```
        tahmin(3) = "Olmayabilir"

        tahmin(4) = "Muhtemelen"

        tahmin(5) = "Eminim"

        tahmin(6) = "Bahse girerim"

        numara = Int(Rnd() * ((7-1) + 1))

        SihirliSekizTop = tahmin(numara) & "<BR><FONT SIZE=2>Başka bir
                          istek için refresh yap</FONT>"
End Function

-->

</SCRIPT>


<BODY BGCOLOR="BLACK" TEXT="Black" LINK="White" VLINK="White"
ALINK="White">

<A HREF="ornek.htm" ID="Renkler"> İsteğinizi içinizde tutun ve daha sonra
cevabınızı almak için fareyi text bilgi üzerinde hareket ettirin.
</A>


<SCRIPT LANGUAGE="VBScript">

<!--

Document.write "<BR><FONT SIZE=7>" & SihirliSekizTop & "</FONT>"

-->

</SCRIPT>

</BODY>

</HTML>
```



Text bilgi üzerinden fareyi hareket ettirdikten sonra aşağıdaki görüntü elde edilir.

links

Doküman içinde taşınan URL leri gösteren bir string (0...links.length) dizisi.

Aşağıda bir doküman üzerinde 3 tane bağ (link) tanımlanmıştır, bu bağlar Links dizisi üzerinden indis kullanılarak erişilir.

```
<A HREF="http://www.ibm.com">IBM</A>
<A HREF="http://www.hp.com">HP</A>

<A HREF="http://www.microsoft.com">Microsoft</A>
```

Doküman üzerindeki bağlara aşağıda script içinden nasıl erişileceği gösterilmiştir.

```
<SCRIPT LANGUAGE="VBScript">

Document.Write Document.Links(0)

Document.Write Document.Links(1)

Document.Write Document.Links(2)

</SCRIPT>
```

forms

Bir form nesneleri (0 .. forms.length) dizisi. Doküman üzerinde tanımlanan formlar 0 dan başlayarak numaralandırılırlar. 0 indisi ile forms dizisinden ilk form işaret edilir.

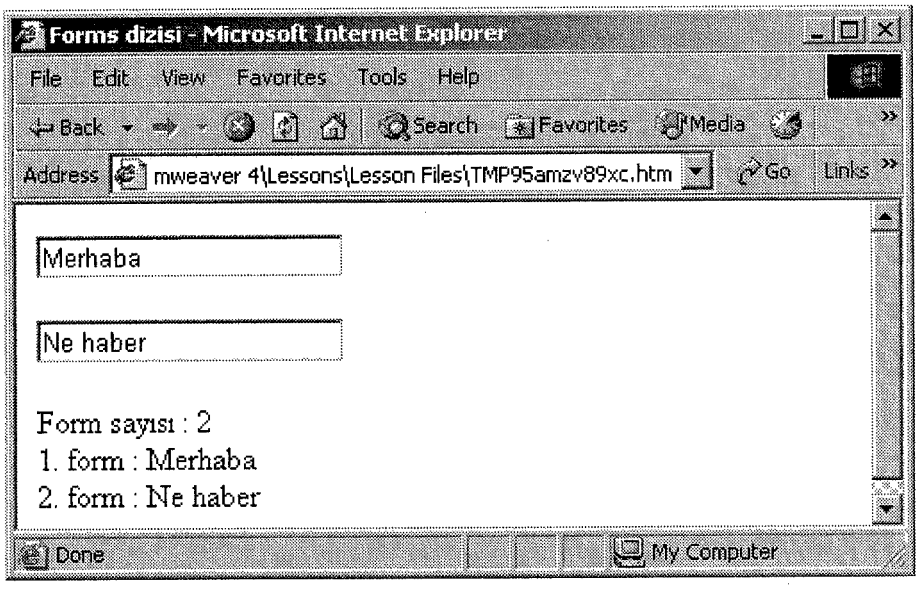
```
<FORM NAME="form1" METHOD="post" ACTION="">

<SCRIPT LANGUAGE="VBScript">

Document.Write Document.Forms.Length

' Yukarıdaki komut doküman üzerindeki form sayısını verir.

</SCRIPT>

</FORM>
```

Başka bir örnek

```
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">

<FORM NAME="form1" METHOD="post" >
```

```
<INPUT TYPE="text" NAME="bilgi1" VALUE="Merhaba">

</FORM>

<FORM NAME="form2" METHOD="post" >

<INPUT TYPE="text" NAME="bilgi2" VALUE="Ne haber">

</FORM>

<SCRIPT LANGUAGE="VBScript">
Document.Write "Form sayısı : " & document.Forms.Length & "<BR>"

Document.Write "1. form : " & Document.Forms(0).bilgi1.Value & "<BR>"

Document.Write "2. form : " & Document.Forms(1).bilgi2.Value & "<BR>"

</SCRIPT>
```

Verilen örnek programın tarayıcı üzerindeki görünümü şekildeki gibidir.



lastModified

Dökümanın en son değiştirildiği tarih.

Aşağıdaki örnekte, dökümanın en son değiştirildiği tarihi ve ilgili sayfanın yüklendiği zamanı verir.

```
<SCRIPT TYPE="text/vbscript" LANGUAGE="VBScript">

<!--

Document.Write "Güncelleme tarihi: " &  Document.LastModified  & ".<BR>"

Document.Write "Sayfanın yüklendiği zaman: " & Time()

-->

</SCRIPT>
```

Metodlar

clear ()

Dökümanın içeriğini temizler. Buradaki temizleme işlemi bizim tarafımızdan yapılan değişiklikler üzerinde geçerlidir , dolaysıyla orijinal doküman üzerinde bir değişim olmaz.

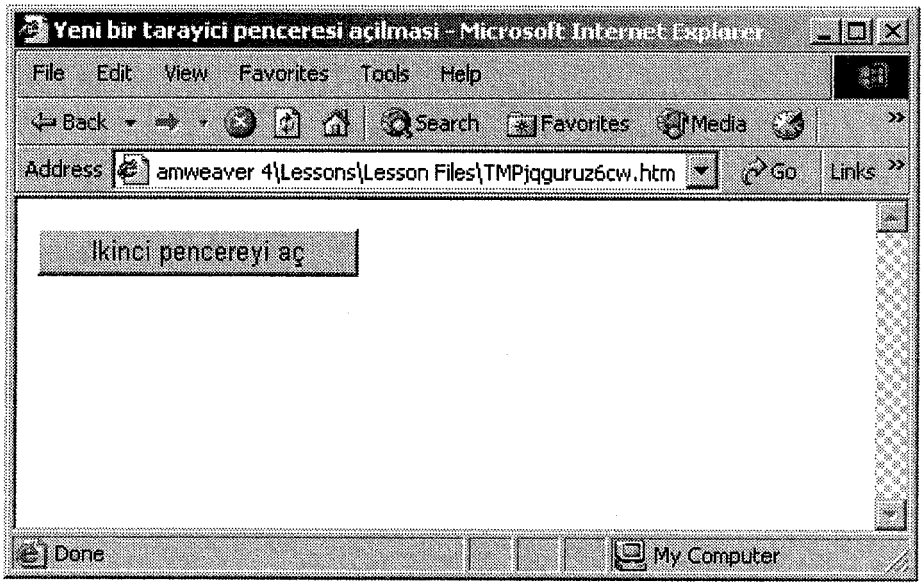
```
Document.clear()
```

open ()

Yazma işlemi için dökümanı açar.

```
<HTML>

<HEAD>

<TITLE>Yeni bir tarayıcı penceresi açılması</TITLE>

<SCRIPT LANGUAGE="VBScript">

<!--

Sub dugme_OnClick
  Open "http://www.deneme.com", "pencere", "toolbar=no, menubar=no,
        width=640, height=480"
End Sub

-->
```

```
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FORM>
<INPUT ID="dugme" TYPE=BUTTON VALUE="İkinci pencereyi aç">
</FORM>
</BODY>
</HTML>
```



close ()

Döküman içine yazma işlemini sonlandırır ve içeriğini tarayıcı üzerinde gösterir.

```
<HTML>
<HEAD>
<TITLE>Tarayıcı penceresini kapatma</TITLE>
</HEAD>
```

```
<SCRIPT LANGUAGE="VBSCRIPT">
<!--
Sub dugme_OnClick
        close
End Sub
-->
</SCRIPT>
<BODY BGCOLOR="#FFFFFF">
<FORM>
<INPUT TYPE=BUTTON ID="dugme" VALUE="Tarayıcı penceresini kapat">
</FORM>
</BODY>
</HTML>
```

write (string), writeln (string)

Döküman kapatılmadan gösterilecek olan text bilginin doküman içine yazılmasını sağlar. Örneğin,

```
<SCRIPT LANGUAGE="VBScript">
        Document.Write "<H1>İşte bir deneme</H1>"
</SCRIPT>
```

Olaylar

onLoad

Doküman yüklendiğinde bir olay tetiklenir.

onUnload

Doküman yok edildiğinde bir olay tetiklenir.

```
<HTML>

<HEAD>

<TITLE>onLoad ve onUnload kullanımı</TITLE>

</HEAD>

<SCRIPT LANGUAGE="VBSCRIPT">

<!- -

' Ziyaret etmek istediğiniz sayfa tarayıcınıza yüklenirken aşağıdaki alt rutin aktif
' olur ve size bir mesaj verir.

Sub window_onLoad

        MsgBox "Web sayfama hoşgeldiniz!",,"Hoşgeldiniz!"

        Status = "Web sayfama hoşgeldiniz!"

End Sub

' Ziyaret etmiş olduğunuz web sayfası penceresini kapattığınızda aşağıdaki alt
' rutin aktif olur ve size mesaj verir.

Sub window_onUnload

        MsgBox "Teşekkürler, görüşmek üzere!",,"Görüşürüz!"

End Sub

-->

</SCRIPT>

<BODY BGCOLOR="#FFFFFF">

</BODY>

</HTML>
```

History

History nesnesi script kodları ile sayfalar içinden ileri veya geri gitmek için üç tane metod sağlar. Bunun yanında o bu dökümanların lokasyonlarına erişim izni vermez.

Özellikler

length

History tarafından tutulan dökümanların sayısını verir.

```
<SCRIPT LANGUAGE="VBScript">
Document.Write "History uzunluğu : " & History.Length
</SCRIPT>
```

Metodlar

back ()

Bir doküman geriye gider.

forward ()

Bir doküman ileriye gider.

History nesnesine ait olan back() ve forward() metodlarının kullanımı ile ilgili örnek aşağıda verilmiştir.

```
<HTML>
<HEAD>
<TITLE> History üzerinde ileri ve geri gitme</TITLE>
</HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
        Sub Geri_onClick
                top.history.back
        End Sub
```

```
        Sub Ileri_onClick
                top.history.forward
        End Sub
-->
</SCRIPT>
<BODY BGCOLOR="#FFFFFF">
<FORM>
<INPUT NAME="Geri" VALUE="Geri git" TYPE=BUTTON>
<INPUT NAME="Ileri" VALUE="İleri git" TYPE=BUTTON>
</FORM>
</BODY>
</HTML>
```

go (tamsayi)

Tamsayı ile belirlenen sayı kadar ileri veya gerideki bir doküman gider.

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

' yukarıda verilen komut ile history üzerinde bir önceki doküman URL sine gidilir.
```

Olaylar

onLoad

Doküman yüklendiğinde bir olay tetiklenir.

onUnload

Doküman yok edildiğinde bir olay tetiklenir.

Location

Location nesnesi bir URL kapsar, ve URL elemanlarının (host, port, ve path) geçişine kolay erişim sağlar.

Özellikler

Protocol

Kaynağa erişmek için kullanılan protokol (örneğin: http, ftp, gopher gibi)

hostname

URL nin sunucu adı kısmını verir.

port

URL nin port kısmını verir.

pathname

URL nin yol (path) kısmını verir.

```
<HTML>

<HEAD>
```

```
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#CCFFFF">
<SCRIPT LANGUAGE="VBScript">
        Document.Write Location.protocol & "<BR>"
        Document.Write Location.hostname & "<BR>"
        Document.Write Location.port & "<BR>"
        Document.Write Location.pathname & "<BR>"
</SCRIPT>
</BODY>
</HTML>
```

Link

Link nesnesi bir URL bağı kapsar, ve URL bağ elemanlarının (host, port, ve path) geçişine kolay erişim sağlar.

```
<A HREF="Deneme.htm#deneme">Denemeler</A>
```

URL şekli aşağıdaki gibidir.

```
<protocol>//<host>[:<port>]/<pathname>
```

Özellikler

protocol

Kaynağa erişmek için kullanılan protokol (örneğin: http, ftp, gopher gibi)

hostname

URL nin sunucu adı kısmını verir.

port

URL nin port kısmını verir.

pathname

URL nin yol (path) kısmını verir.

target

Bağın (link) hedef pencerisini belirler.

Olaylar

OnMouseMove

Bir fare kürsörü bağ (işaret edilen bir hiperbağın hangi kısmı olduğunu belirlemek için faydalıdır) üzerinde hareket ederken tetiklenen bir olay.

OnMouseOver

Bir fare kürsörü bir bağ üzerinden ilk kez hareket ettiğinde tetiklenen bir olaydır.

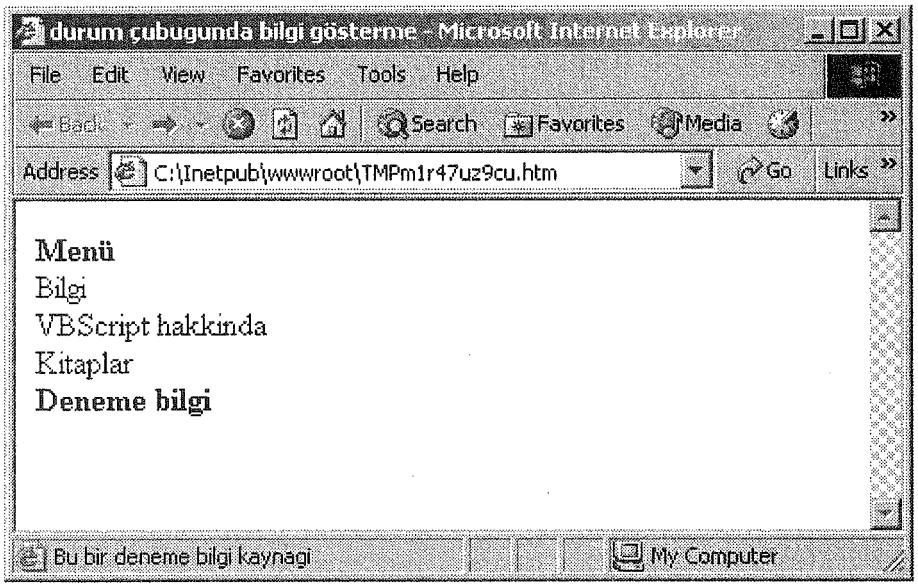
OnClick

Bir bağ (link) üzerine tıklandığı zaman meydana gelen olay.

```
<HTML>
<HEAD>
<TITLE>durum çubugunda bilgi gösterme</TITLE>
</HEAD>
<STYLE>
<!--
        A { text-decoration: none; }
```

```
        A:hover { text-decoration: underline; color: none; }
        A:visited { color: blue; }
-->
</STYLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub window_onLoad
        window.defaultStatus = ""
End Sub
Sub penbil_onMouseMove
        window.status = "Pencere bilgi kaynağı"
End Sub
Sub vbs_onMouseMove
        Window.Status = "VBScript bilgi kaynağı!"
End Sub
Sub kitap_onMouseMove
        Window.Status = "Kitaplar hakkında herşey"
End Sub
Sub deneme_onMouseOver
        Window.Status = "Bu bir deneme bilgi kaynağı"
End Sub
-->
</SCRIPT>
<BODY BGCOLOR="#FFFFFF">
<B>Menü</B>
```

```
<BR>
<A ID="bilgi" NAME="penbil" HREF="http://www.bilgi.com">Bilgi</A>
<BR>
<A ID="vbs" HREF="http://www.vbs.com">VBScript hakkinda</A><BR>
<A ID="kitap" HREF="http://www.kitap.com">Kitaplar</A><BR>
<B ID="deneme">Deneme bilgi</B>
</BODY>
</HTML>
```



Anchor

Bir doküman içindeki bir işaretleme noktasını gösteren bir köprü.

Özellikler

name

Köprü adı, bu köprü adı hedef doküman içinde yer alan bir nesnedir.

<A NAME="deneme_anchor">Deneme Yazılar</A>

Navigator

Navigator nesnesi o anki tarayıcı hakkında bilgi verir. Kullanılabilecek olan imkanları belirlemek için faydalı olabilecek bilgiler (tarayıcı adı ve versiyonu gibi).

Özellikler

appCodeName

Tarayıcının kod adını verir.

appName

Tarayıcının tam adını verir.

appVersion

Tarayıcının güncel versiyon bilgisini verir.

userAgent

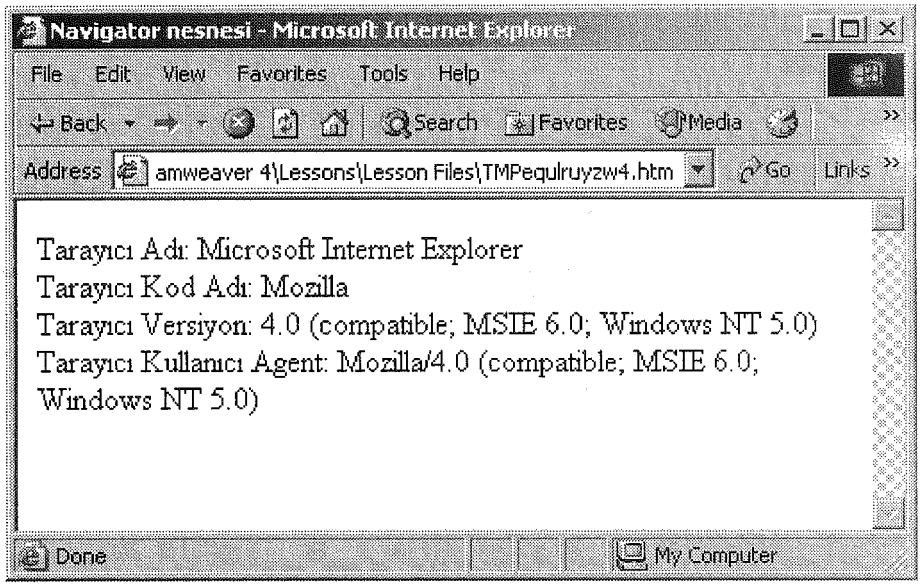
Bir varolan başlığın bir parçası olarak HTTP sunucularına gönderilen userAgent stringi.

Örnek

```
<HTML>
<HEAD>
<TITLE>Navigator nesnesi</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=">
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<SCRIPT LANGUAGE="VBScript">
Document.Write "Tarayıcı Adı: " & Navigator.AppName & "<BR>"
```

```
Document.Write "Tarayıcı Kod Adı: " & Navigator.AppCodeName & "<BR>"
Document.Write "Tarayıcı Versiyon: " & Navigator.AppVersion & "<BR>"
Document.Write "Tarayıcı Kullanıcı Agent: " & Navigator.UserAgent & "<BR>"
</SCRIPT>
</BODY>
</HTML>
```

Örnek HTML sayfasının tarayıcı üzerindeki görünümü şekildeki gibidir.



Form

Bir HTML formunu gösteren bir form nesnesi.

Özellikler

element

Form elemanlarından oluşan bir dizi (0 dan element.length uzunluğunda).

Metodlar

submit ()

Bir formu sunucuya gönderir.

Olaylar

onSubmit

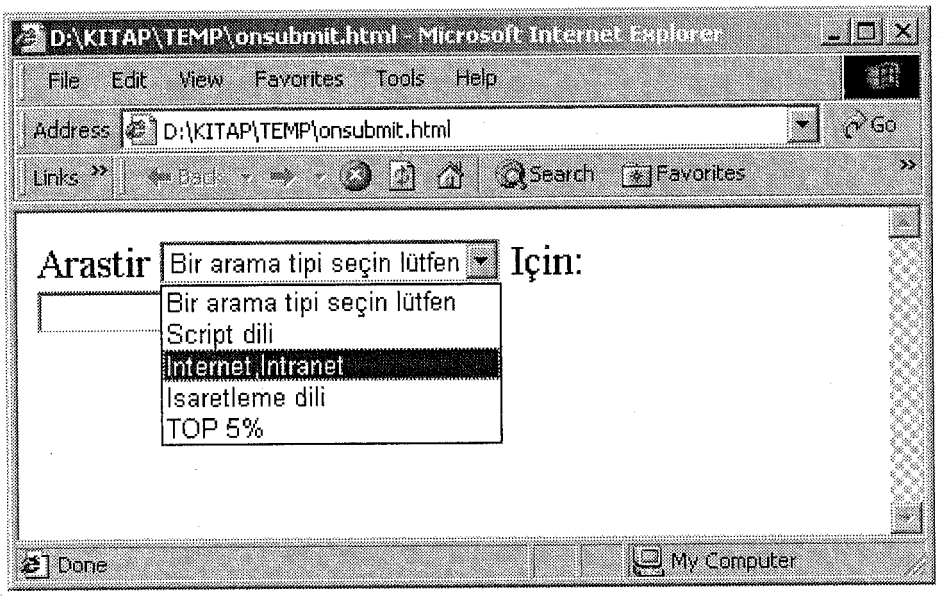
Bir form sunucuya gönderildiğinde meydana gelen olay.

```
<HTML>
<HEAD>
<TITLE> OnSubmit olay kotarıcı örneği</TITLE>
<SCRIPT TYPE="text/vbscript" LANGUAGE="VBScript">
<!- -
Function tForm_OnSubmit()
   If Len(tForm.sorgula.Value) = 0 Then
      tForm_OnSubmit = False
      Alert "Bir sözcük girmek zorundasınız."
      Exit Function
   End If
   If tForm.kitap.Value = "null" Then
      tForm_OnSubmit = False
      Alert "Bir arama tipi seçmek zorundasınız."
      Exit Function
   End If
End Function
```

```
'- ->

</SCRIPT>

</HEAD>

<BODY>

<FORM ACTION="http://www.pcs.com/script/ara.asp" METHOD=GET
NAME="tForm">

Araştır

<SELECT NAME="kitap">

<OPTION value="JavaScript">Bir arama tipi seçin lütfen</OPTION>

<OPTION value="VBScript">Script dili</OPTION>

<OPTION value="Network">Internet,Intranet</OPTION>

<OPTION value="XML">İşaretleme dili</OPTION>

<OPTION value="ASP">TOP 5%</OPTION>

</SELECT>

İçin:<INPUT TYPE="text" NAME="sorgula" VALUE="" SIZE=22>

<INPUT TYPE="submit" VALUE="Getir">

</FORM>

</BODY>

</HTML>
```

Bir kitap aramak için olan bir form , fakat eğer bir arama tipi, ve bir sözcük girmezseniz form submit edilmeyecektir.

Örnek programın çıktısı aşağıda verilmiştir.



Elements

Script kodları tarafından erişilebilen ve değiştirilebilen bir form elemanını (bir text alanı, düğme, çekme listeler, onay kutuları veya radyo düğmeleri gibi) gösteren bir nesne.

Özellikler

value

Form elemanın değeri (çoğu form elemanı için geçerlidir.)

name

Form elemanının adı.

Metodlar

click ()

Bir click olayını tetikler.

focus ()

İlgili pencere için aktif olarak elemanı odaklamayı yapar.

blur ()

İlgili eleman için aktif haldeki odaklamayı kaldırır.

select ()

Elemanın içeriğini seçer.

Olaylar

onChange

Eleman değiştirildiğinde tetiklenen olay.

onClick

Bir eleman üzerine tıklandığı zaman tetiklenen olay.

Aşağıdaki örnekte OnClick olay kotarıcısı bir form elemanı olan düğme üzerine tıklandığında cevap vermek için kullanılıyor.

```
<HTML>

<BODY>

<P>Click olayı için örnek.<BR>

<FORM>

<SCRIPT LANGUAGE="VBScript"></SCRIPT>

<INPUT TYPE=BUTTON VALUE="Tıkla" NAME=Dugme1 OnClick="Alert 'Selam
Herkese.'">

</FORM>

</BODY>

</HTML>
```

onFocus, onBlur

Elemanlar odaklandığında / odaklama kaldırıldığında tetiklenen olay.

onSelect

Text elemanı seçildiğinde tetiklenen olay (birkaç tarayıcı için bu olay geçerli değildir).

Örnek

```
<HTML>
<HEAD>
<TITLE>Bir text elemanı üzerinde olayların kullanımı</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FONT FACE="Verdana, Arial, Helvetica" SIZE=2>
<SCRIPT LANGUAGE="VBSCRIPT">
<!--
        Sub bilgi_onFocus
                alert "Odaklanma meydana geldi"
        End Sub
        Sub bilgi_onBlur
                alert "Odaklanma kalktı"
        End Sub
        Sub bilgi_onChange
                alert "Text bilgide değişme oldu"
        End Sub
        Sub bilgi_onSelect
                alert "Text bilgi seçildi"
        End Sub
```

```
-->

</SCRIPT>

</FONT>

<FORM NAME="Form1">

Text elemanı üzerinde birşeyler yaparak sonuçları görebilirsiniz:

<BR>

<INPUT TYPE=TEXT NAME="bilgi" VALUE=" Buraya birşeyler yazın" SIZE=50
MAXLENGTH=50>

</FORM>

</BODY>

</HTML>
```

Çerçeveler , Formlar ve Elemanlar

Script yazarak en çok kullanımlarından biri bir form üzerinde oluşan olaylara cevap vermektir, örneğin,bir text alanı içine yazmak veya Submit düğmesine tıklamak gibi. Bir form elemanına işaret etmek için iki yol vardır. İlk yöntem, nesne modeline göre işaret edebilirsiniz. Document.Form(0).Elements(1).Value sayfa üzerindeki ilk formun ikinci elemanın özelliğinin değerine işaret eder. Document.Forms.Count kullanarak sayfadaki formların sayısını belirleyebilirsiniz, ve Document.Forms(0).Elements.Count kullanarak ilk formdaki elemanların sayısını belirleyebilirsiniz.

Bu elemanlara işaret etmenin zor yöntemidir. Eğer form ve elemanların HTML takılarındaki NAME özelliğini dahil ederseniz, onunla işaret edebilirsiniz.

```
<FORM NAME="xForm">
<INPUT TYPE="TEXT" NAME="txtSelam">

</FORM>
```

Bu elemanın değeri xForm.txtSelam.Value kullanarak işaretlenebilir.

Özellikle IE4+ için yazmıyorsanız, form elemanlarını bir form içine yerleştirmelisiniz, ve bu elemanlar için ID özelliğinden ziyade NAME özelliğini kullanmalısınız. Form elemanları hem özellikleri hemde olayları açığa çıkarır. İzleyen örnek bunu gösteriyor.

```
Sub txtSelam_OnFocus()
  With xForm.txtSelam
    If .Value = "Merhaba" Then
      .Value = "Hoşçakal"
    Else
      .Value = "Merhaba"
    End If
  End With
End Sub
```

OnFocus olayı eleman giriş için aktif olduğunda meydana gelir, tıklanarak veya tab ile geçiş ile. With bağıl referansların kullanımına izin veren bir sözcüktür. Normal olarak xForm.txtSelam.Value şeklinde yazmamız gereklidir, fakat With. şeklini kullanmak daha basittir. Bunu yapmak için başka bir yol ise Set sözcüğünü kullanmaktır. Bu sözcük bir değişkeni başka bir nesne içine bir kısayol oluşturmayı sağlar.

Örneğin,

```
Sub txtSelam_OnFocus()
  Dim aKont
  Set aKont = aForm.txtSelam
    If aKont.Value = "Merhaba" Then
      aKont.Value = "Hoşçakal"
    Else
      aKont.Value = "Merhaba"
    End If
  End With
End Sub
```

Burada şunu görmelisiniz, olaylara bir Sub prosedürden ziyade bir fonksiyon olarak baktım. Eğer bir olayı iptal edebilmek istiyorsanız, onu nasıl yapabilirsiniz, onu bir

fonksiyon yapmakla ve sonra ya True yada False döndürürsünüz, False olayın iptal edilmesini sağlar.

Window.Frames Document.Forms yapısına benzer, isim ve indeks ile başvurulabilen çok sayıda tutamaç içerebilir ve bir Count özelliğine sahiptir. Fark ise Forms elemanlar taşırken, Frames ise bir doküman içerir.

Çerçeveler pencerelere benzerler. Bir doküman içermek ve diğer çerçeveleri taşıyabilir. Gerçekte, neredeyse tüm yöntemlerde, siz o bir pencereymiş gibi davranabilirsiniz.

Siz bir çerçeveden diğer bir çerçeveye Window.Parent nesnesini kullanarak ulaşabilirsiniz. O üzerindeki pencereye işaret eder.

Çerçeveler NN2+ ve IE3+ tarafından desteklenir, ve HTML 4 belirtimlerinin bir parçasıdır. Script yazmak için, kayan çerçeveler düzenli çerçevelerden daha kullanışlıdır. IFRAME takısı size bir sayfanın ortasında bir resim gibi duran bir çerçeve yapmanıza izin verir. Bu IE3+ tarafından desteklenir, ve bir HTML 4 belirtimi değildir. Onlar aynı şekilde düzenli çerçeveler olarak başvurulur.

Aşağıdaki örnek bir çerçeve kümesinin nasıl tanımlanacağını ve çerçeveler arası etkileşimin nasıl olacağını gösterir.

```
<HTML>

<HEAD>

<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">

<TITLE>Bir çerçeveye başka bir çerçeveden erişim</TITLE>

</HEAD>

<FRAMESET COLS="100,*">

        <FRAME SRC="arastir.htm">

        <FRAME SRC="main.htm">

</FRAMESET>

</HTML>
```

main.htm dosyasının arka plan rengi çerçevelerin yüklenmesi esnasında , arastir.htm sayfası içindeki script kod tarafından farklı bir renge boyanmaktadır. Buradada görebileceğiniz gibi farklı bir çerçeveden başka bir çerçeve üzerinde değişiklik yapabiliyoruz.

------ Main.htm sayfası -------

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Zemin rengi değişen sayfa</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
        Sub Window_onLoad
                window.top.frames(0).document.bgColor = "#CCFF66"
        End Sub
-->
</SCRIPT>
</HEAD>

<BODY>
<FONT FACE="Verdana, Arial, Helvetica" SIZE=2>
<P>
<HR WIDTH=100% SIZE=1 COLOR="ORANGE">
```

Çerçeve kümesinden birinin içeriği olan main.htm sayfası zemin rengi, değişime uğramıştır.

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

arastir.htm dosyası aşağıdaki kodları içerir.

```
------- arastir.htm --------
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
        Sub Window_onLoad
                window.top.frames(1).document.bgColor = "#66FFCC"
        End Sub
-->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

BÖLÜM
ON DÖRT

14

VBScript Nesneleri

- Class
- Dictionary
- Drive
- Drives Collection
- Err
- File
- Files Collection
- FileSystemObject
- Folder
- Folders Collection
- Match
- Matches Collection
- RegExp
- TextStream
- Nesneler için For Each...Next Çevrimi

VBScript Nesneleri

VBScript içinde yer alan nesneler aşağıda liste halinde verilmiştir. Bu nesnelere ait geniş açıklamalar izleyen kısımlarda yer almaktadır. VBScript içinde hazır olan nesneleri kullanabileceğiniz gibi aynı zamanda kendi nesnelerinizi oluşturabilirsiniz, bu nesneler üzerinde özellikler ve metodlar tanımlayabilirsiniz. Nesne oluşturmak için Class yapısını kullanırız. Bu konu ile detaylı bilgiler izleyen kısımlarda verilmiştir.

Visula Basic Script nesneleri :

- Class
- Dictionary
- Drive
- Drives Collection
- Err
- File
- Files Collection
- FileSystemObject
- Folder
- Folders Collection
- Match
- Matches Collection
- RegExp
- TextStream

Class Nesnesi

Class yapısını kullanarak bir sınıf tanımlaması oluşturduğunuzda döndürülen nesne. Terim olarak nesne Class . . . End Class ifadesini kullanarak oluşturmak istediğiniz herhangi bir nesneye işaret eder. Bu yetenek size kendi nesnelerinizi oluşturmak ve VBScript dilinin çok yönlülüğünün önemli uzantısını gösteren geçerli bir VBScript kod kullanarak işlemler yapmanıza izin verir. Class bir anahtar sözcük olduğu için bir sınıf adı olarak kullanamazsınız.

Sınıflar VBScript 5 nci versiyonunun diğer bir imkanıdır. Düzenli ifadelerin yazım şekli gibi değildir, VBScript sınıfları dilin yapısına uyar. Bir sınıf bir birim olarak birlikte çalışacak özellikler ve kod içerir. Çoğu sınıflar sizin ayarlayabileceğiniz ve alabileceğiniz değerler taşır ve çağrılabilir fonksiyonlar veya alt rutinler olarak da metodlar içerir. Sınıflar hem gizli (private) hemde görünen (public) özellikler ve metodlar taşır.

VBScript içindeki çoğu diğer nesnelere benzemez, siz doğrudan sınıf nesneleri

oluşturabilirsiniz. Sınıf için sağladığınız kod yerine ve VBScript sınıf nesnesine bir referans döndürür. Siz sınıfları hem istemci tarafı hemde sunucu tarafı kod yazmak istediğinizde çok faydalı olacaktır.

Class...End Class kullanarak bir sınıf tanımı yapabilirsiniz. Class yapısının yazım şekli aşağıdaki gibidir.

```
Class sınıfadı
```

Burada sınıfadı bizim vermiş olduğumuz sınıf adıdır. Sınıflara isim verme VBScript değişkenlerine isim verme kurallarına uyar.

Sınıflar değişkenler, özellikler, metodlar, ve olaylar içerebilir. Bunların kaç tane olduğu ve tiplerinin ne olduğu size bağlıdır. Özellikler veya metodlar içermeyen sınıflar oluşturmak mümkün ve yanlızca iki tane olayı desteklemeside mümkün. O zaman bu sınıf faydalı bir sınıf olmayacaktır.

Aşağıda classTest adlı bir sınıf tanımını göreceksiniz, bu sınıf içinde ilgili string bilginin üzerinde yapılan özellikler ve işlemler tanımlanmıştır.

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Class classTest

        Private m_strTest

        Private Sub Class_Initialize()

                m_strTest = "VBScript Programlama Harika!"

        End Sub


        Private Function PrivateTest(strTest)

                PrivateTest = strReverse(strTest)

        End Function

        Public Property Get TestProperty()

                TestProperty = m_strTest

        End Property

        Public Property Let TestProperty(strTest)

                m_strTest = strTest
        End Property
```

```
        Public Sub Test()
                m_strTest = PrivateTest(m_strTest)
        End Sub
End Class
. . .
```

Siz bir sınıf adını bildirmek için Class ifadesini kullandıktan sonra, o zaman aşağıda gösterildiği gibi Set ve New kullanarak o sınıfa bir tutamaç tanımlayabilirsiniz.

```
. . .
Dim  nesneTest
Set  nesneTest = New classTest
. . .
```

nesneTest değişkeni bir nesneye işaret eden bir referans, classTest sınıfının yeni bir tutmacını tutar. Siz şimdi nesnenin özelliklerini ayarlayabilirsiniz.

```
. . .
Response.Write nesneTest.TestProperty & "<BR>"
nesneTest.Test
Response.Write nesneTest.TestProperty & "<BR>"
nesneTest.TestProperty = "Nesneleri nasıl!"
Response.Write nesneTest.TestProperty & "<BR>"
nesneTest.Test
Response.Write nesneTest.TestProperty & "<BR>"
Set nesneTest = Nothing
. . .
</SCRIPT>
```

Başka bir örnek

Aşağıdaki örnek basit bir Class oluşturur, "kullanici" adı bir class olup tek bir özelliğe sahiptir, KullaniciAdi, ve bir metoda sahip, "KullaniciAdiniGoster".

```
<%
Class Kullanici
```

```
' private class değişken tanımı

Private m_kullaniciAdi

' özellik tanımı


Public Property Get KullaniciAdi

  KullaniciAdi = m_kullaniciAdi

End Property


Public Property Let KullaniciAdi (strKullaniciAdi)

  m_kullaniciAdi = strKullaniciAdi

End Property

' metod tanımlama ve bildirme

Sub KullaniciAdiGoster

  Response.Write KullaniciAdi

End Sub

End Class

%>
```

Sınıf tanımını yaptıktan sonra, aşağıda gösterildiği gibi, siz oluşturduğunuz class için bir tutamaç oluşturabilirsiniz, bunu yapmak Set ve New komutlarını kullanırız.

```
<%

Dim nesneKullanici

Set nesneKullanici = New Kullanici

' KullaniciAdi özelliğini ayarlama

nesneKullanici.KullaniciAdi = "Administrator"

' KullaniciAdiGoster metodunu çağırma
```

```
' Kullanıcı adını yazdırma

nesneKullanici.KullaniciAdiGoster

%>
```

Çıkış:

```
"Administrator"
```

VBScript Class nesnesi üzerinde tanımlı olan olaylar

- Initialize
- Terminate

Initialize

Bir VBScript Class nesnesi hemen olurken meydana gelir.

```
Private Sub Class_Initialize()

  ifadeler

End Sub
```

ifadeler kısmı sınıf başlatıldığında çalıştırmak için sıfır veya daha çok kod ifadeleri içerir.

Aşağıdaki örnek Initialize olayının kullanımını gösterir.

```
<%
...

Class sınıfTesti

  Private Sub Class_Initialize   ' Initialize olayı.

    MsgBox("Sınıf testi başladı")

  End Sub

  Private Sub Class_Terminate   ' Terminate olayı.

    MsgBox("Sınıf testi durdu")

  End Sub

End Class
```

```
Set T = New sınıfTesti   ' sınıfTesti için tutamaç oluşturuluyor.

Set T = Nothing   ' Tutamaç yok ediliyor.
...

%>
```

Terminate

Sadece VBScript sınıfı yok edilirken meydana gelir. Siz bunu nesne başvuruları ve değişkenleri yok ederek temizlemek için kullanın.

```
Private Sub Class_Terminate()

  ifadeler

End Sub
```

ifadeler kısmı sınıf başlatıldığında çalıştırmak için sıfır veya daha çok kod ifadeleri içerir.

Aşağıdaki örnek Terminate olayının kullanımını gösterir.

```
<%
...
Class sınıfTesti

  Private Sub Class_Initialize   ' Initialize olayı.

    MsgBox("Sınıf testi başladı")

  End Sub

  Private Sub Class_Terminate   ' Terminate olayı.

    MsgBox("Sınıf testi durdu")

  End Sub

End Class

Set T = New sınıfTesti   ' sınıfTesti için tutamaç oluşturuluyor.

Set T = Nothing   ' Tutamaç yok ediliyor.
...
%>
```

Dictionary Nesnesi

Dictionary nenesi ad/değer çiftini bir dizi içinde tutar. Anahtar maddeye benzeyen tek dize bir tanıtıcıdır ve aynı nesne içinde herhangi diğer bir tanıtıcı için kullanılamaz.

Bir Dictionary nesnesi bir verinin herhangi bir şekli olabilir, bu veri dizi içinde saklanır. Herbir madde (item) bir tek anahtar ile ilişkilendirilir. Anahtar bireysel bir maddeyi getirir ve genellikle bir tamsayı veya bir string olur, fakat bir dizi dışında herşey olabilir.

Aşağıdaki kod "sehir" adlı bir Dictionary nesnesi oluşturur, birkaç anahtar/madde parçaları ekler, madde özelliği kullanılarak 'c' anahtarı için madde değerini getirir ve çıkışlar tarayıcıya string olarak döner.

```
<%

Dim sehir

Set sehir = CreateObject("Scripting.Dictionary")

sehir.Add "a", "İstanbul"

sehir.Add "b", "Ankara"

sehir.Add "c", "Bursa"

sehir.Add "d", "Erzurum"

sehir.Add "e", "Sivas"

Response.Write "'c'anahtarı için değer" & sehir.Item("c")

%>
```

Çıktısı "'c'anahtarı için değer Bursa" olur.

Dictionary Nesne Özellikleri

CompareMode Özelliği

Bu özellik arama veya tarama yapılırken anahtarların nasıl uyum sağlayacağını belirleyen anahtar string karşılaştırma modunu döndürmek veya ayarlamak için kullanılır.

```
nesne.CompareMode[ = karsilastirma_modu]
```

Kaşılaştırma modu için verilen tablodaki sabitleri veya değerleri kullanabilirsiniz.

Sabit	Değer	Tanımı
vbBinaryCompare	0	İkili karşılaştırma.
vbTextCompare	1	Text karşılaştırma.

Count Özellliği

Bu özellik Dictionary nesnesi içinde anahtar/madde çiftinin sayısını belirlemek için kullanılır.

nesne.Count

Önceki örnekte tanımlanan sehir dictionary nesnesi için eklenen maddelerin sayısını aşağıdaki kod ile elde edebiliriz.

Document.Write sehir.Count

Aşağıdaki kod **Count** özelliğinin kullanımını gösteriyor.

```
' Birkaç değişken oluşturuyoruz

Dim madde, sozluk, i

Set sozluk = CreateObject("Scripting.Dictionary")

' Birkaç anahtar ve madde ekliyoruz

sozluk.Add "a", "HP"

sozluk.Add "b", "Cisco"

sozluk.Add "c", "Exper"


' Birkaç anahtar getir

madde = sozluk.Keys

' Dizi üzerinde ardışıl işlemler yapıyoruz

For i = 0 To sozluk.Count -1

        Print madde(i) 'Print key
Next
...
```

Item Özelliği

Bu özellik belirtilen anahtar argümanı ile gösterilen koleksiyondaki bir maddenin değerini alıp getirmemize ve maddedegeri ile değeri ayarlamamıza izin verir

```
nesne.Item(anahtar) [ = maddedegeri]
```

Eğer bir madde değiştirme esnasında anahtar bulunamaz ise, o zaman belirtilen yeni madde değeri ile yeni bir anahtar oluşturulur. Eğer mevcut bir madde döndürmek istenildiğinde anahtar bulunamaz ise, o zaman yeni bir anahtar oluşturulur ve ona uygun madde boş bırakılır.

Key Özelliği

Bu özellik var olan bir anahtar/madde çiftinin anahtar değerini değiştirmemize izin verir.

```
nesne.Key(anahtardegeri) = yeniAnahtardegeri
```

Eğer bir anahtarı değiştirme esnasında anahtar bulunamaz ise, o zaman bir hata meydana gelecektir.

Dictionary Nesne Metodları

Add Metodu

Bu metod bir Dictionary nesnesine bir yeni anahtar/madde çiftini eklemek için kullanılır.

```
nesne.Add anahtardegeri, maddedegeri
```

Aşağıdaki örnek bir Dictionary nesnesinin nasıl oluşturulduğunu gösteriyor. Örnekte 3 tane anahtar değer ve o anahtar değerler için maddeler verilmiştir.

```
' Bir değişken tanımı

Dim sozluk

Set sozluk = CreateObject("Scripting.Dictionary")

sozluk.Add "a", "Ankara"

sozluk.Add "b", "Bandırma"

sozluk.Add "c", "Çanakkale"
...
```

Exists Metodu

Bu metod belirli bir Dictionary nesnesinde zaten var olan bir anahtar olup olmadığını belirlemek için kullanılır. Eğer anahtar var ise True yoksa False döndürür.

```
nesne.Exists(anahtardegeri)
```

Items Metodu

Bu metod özel bir Dictionary nesnesindeki maddelerin hepsini getirmek için kullanılır ve onları bir dizi içinde saklamak için kullanılır.

```
[diziadi = ]nesne.Items
```

Aşağıdaki kod **Items** metodunun kullanımını gösteriyor.

```
' Birkaç değişken oluşturuyoruz

Dim madde, sozluk, i

Set sozluk = CreateObject("Scripting.Dictionary")

' Birkaç anahtar ve madde ekliyoruz

sozluk.Add "a", "HP"

sozluk.Add "b", "Cisco"

sozluk.Add "c", "Exper"

' Birkaç madde getir

madde = sozluk.Items

' Dizi üzerinde ardışıl işlemler yapıyoruz

For i = 0 To sozluk.Count -1

        Response.Write madde(i)

Next
...
```

Keys Metodu

Bu metod özel bir Dictionary nesnesindeki anahtarların hepsini getirmek için kullanılır ve onları bir dizi içinde saklamak için kullanılır.

```
[diziadi = ]nesne.Keys
```

Aşağıdaki kod **Keys** metodunun kullanımını gösteriyor.

```
' Birkaç değişken oluşturuyoruz
Dim madde, sozluk, i
Set sozluk = CreateObject("Scripting.Dictionary")
' Birkaç anahtar ve madde ekliyoruz
sozluk.Add "a", "HP"
sozluk.Add "b", "Cisco"
sozluk.Add "c", "Exper"
' Birkaç anahtar getir
madde = sozluk.Keys
For i = 0 To sozluk.Count -1
        Response.Write madde(i)
Next
...
```

Remove Metodu

Bu metod belirli bir Dictionary nesnesinden tekbir anahtar/madde çiftini tamamen almak için kullanılır.

```
nesne. Remove(anahtardegeri)
```

Aşağıdaki kod **Remove** metodunun kullanımını gösteriyor.

```
' Birkaç değişken oluşturuyoruz
Dim madde, sozluk, i
Set sozluk = CreateObject("Scripting.Dictionary")
' Birkaç anahtar ve madde ekliyoruz
sozluk.Add "a", "HP"
sozluk.Add "b", "Cisco"
```

```
sozluk.Add "c", "Exper"

' Birkaç anahtar getir

madde = sozluk.Items
. . .
' Bir anahtar ve maddeden oluşan çifti geri alıyoruz

madde = sozluk.Remove("c")
. . .
```

RemoveAll Metodu

Bu metod belirli bir Dictionary nesnesinden tüm anahtar/madde çiftlerini tamamen almak için kullanılır.

```
nesne.RemoveAll
```

Drive Nesnesi

Drive nesnesi yerel veya uzak disk sürücüsünün değişik özelliklerine erişimi sağlar.

Aşağıdaki kod FileSystemObject nesnesinin GetDrive metodunu "c" sürücüsü için Drive nesnesini getirmek için kullanıyor.

```
<%

Set dosyasistemi = CreateObject("Scripting.FileSystemObject")

Set surucu = dosyasistemi.GetDrive("c")

%>
```

Aşağıdaki örnek bir text dosyanın nasıl oluşturulacağını gösterir.

```
<SCRIPT LANGUAGE="VBScript">

Dim nesneFSO , dosya

Set nesneFSO = CreateObject("Scripting.FileSystemObject")

Set dosya = nesneFSO.CreateTextFile("d:\deneme.txt" , true)

dosya.WriteLine("Örnek text bilgiler")

dosya.Close

</SCRIPT>
```

Bu kod diskiniz üzerinde deneme.txt adlı bir dosya oluşturur (eğer siz d adlı bir sürücünüz varsa). O dosyaya bir satır text yazacaktır.

Aşağıdaki kod Drive nesnesinin sürücü özelliklerine erişmek için kullanımını gösterir.

```
Sub BosDiskAlani(surucuyolu)
Dim fs, d, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set d = fs.GetDrive(fs.GetDriveName(surucuyolu))
s = "Drive " & UCase(surucuyolu) & " - "
s = s & d.VolumeName & vbCrLf
s = s & "Boş disk alanı: " & FormatNumber(d.FreeSpace/1024, 0)
s = s & " Kbytes"
Response.Write s
End Sub
```

Drive Nesne Özellikleri

AvailableSpace Özelliği

Belirtilen yerel veya uzak disk sürücüsü üzerinde mevcut boş alanı getirir.

```
nesne.AvailableSpace
```

Aşağıdaki örnek C diskindeki mevcut boş alan miktarını verir.

```
<SCRIPT LANGUAGE="VBScript">
Set dosyaSis = CreateObject("Scripting.FileSystemObject")
Set surucu = dosyaSis.GetDrive("c")
Document.Write surucu.AvailableSpace
</SCRIPT>
```

DriveLetter Özelliği

Belirtilen yerel veya uzak disk sürücüsünün sürücü harfini getirir. Yanlızca okunabilirdir.

nesne.DriveLetter

Aşağıdaki örnek ilgili diske ait sürücü harfini yazar.

```
<SCRIPT LANGUAGE="VBScript">

Set dosyaSis = CreateObject("Scripting.FileSystemObject")

Set surucu = dosyaSis.GetDrive("c")

Document.Write surucu.DriveLetter

</SCRIPT>
```

DriveType Özelliği

Sürücünün tipini gösteren bir tamsayı döndürür.

nesne.DriveType

Bu örnekte seçilen sürücünün tipini gösterir, örneğin, sabit disk, taşınabilir disk, ram-disk gibi.

```
<SCRIPT LANGUAGE="VBScript">

<!- -

'Değişkenleri oluştur

Dim nesneFSO , surucu , strText

'Nesne tutamacı oluştur

Set nesneFSO = CreateObject("Scripting.FileSystemObject")

Set surucu = nesneFSO.GetDrive("c:\")

'Sürücü tipini araştır

Select Case surucu.DriveType

Case 0:
```

```
strText = "Bilinmiyor"
Case 1:
strText = "Taşınabilir"
Case 2:
strText = "Sabit"
Case 3:
strText = "Ağ"
Case 4:
strText = "CD-Rom"
Case 5:
strText = "RAM Disk"
End Select
'Sonuçlar
Document.Write (surucu.DriveLetter & " sürücüsünün tipi"  & strText & "dir.")
// - ->
</SCRIPT>
```

FileSystem Özelliği

Bu özellik belirtilen sürücü üzerindeki kullanımda olan dosya sistemini döndürür.

```
nesne.FileSystem
```

FAT, NTFS, ve CDFS gibi dosya sistemleri mevcuttur.

Aşağıdaki kod **FileSystem** özelliğinin kullanımını gösterir.

```
Sub DosyaSistemiNedir
Dim dosyasistemi,dosya, sistem
Set dosyasistemi = CreateObject("Scripting.FileSystemObject")
Set surucu = dosyasistemi.GetDrive("d:")
```

```
sistem = surucu.FileSystem

Response.Write sistem

End Sub
```

FreeSpace Özelliği

Belirtilen yerel veya uzak sürücü üzerindeki bir kullanıcıya mevcut boş alanı miktarını döndürür.

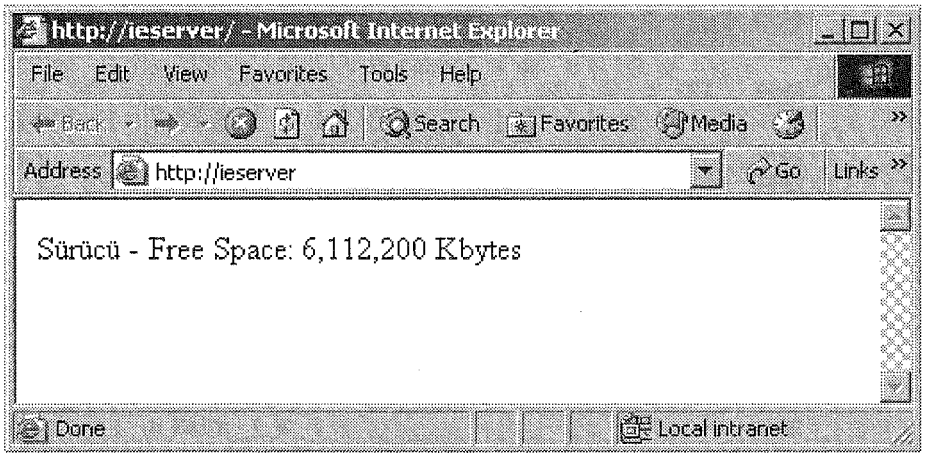
```
nesne.FreeSpace
```

FreeSpace özelliği tarafından döndürülen değer tipik olarak VBScript Drive Object AvailableSpace özelliği tarafından döndürülen değer ile aynıdır. Aralarındaki fark tırnak işaretlerini destekleyen iki bilgisayar sistemleri için olabilir.

Aşağıdaki kod FreeSpace özelliğinin kullanımını gösterir.

```
Sub BosALan(surucuYolu)

Dim fs, d, s

Set fs = CreateObject("Scripting.FileSystemObject")

Set d = fs.GetDrive(fs.GetDriveName(surucuYolu))

s = "Sürücü " & UCase(surucuYolu) & " - "

s = s & d.VolumeName & vbCrLf

s = s & "Free Space: " & FormatNumber(d.FreeSpace/1024, 0)

s = s & " Kbytes"

Response.Write s

End Sub
```

Ekran çıktısı



IsReady Özelliği

Bu özellik belirtilen sürücü kullanım için mevcut ise bir Boolean değeri True yoksa False olur.

nesne.IsReady

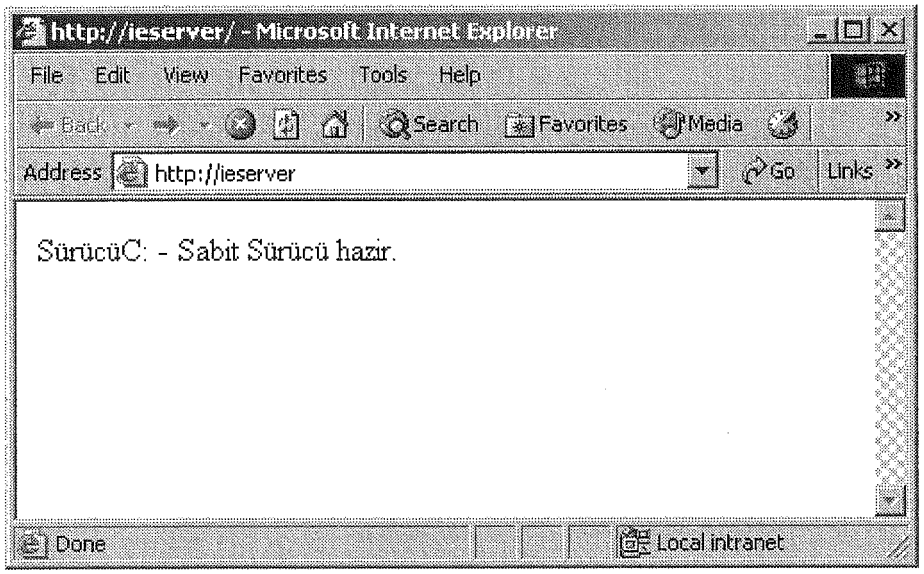
Taşınabilir-ortam sürücüleri ve CD-ROM sürücüler için, IsReady eğer erşim için uygun ortam yerleştirilmiş ise o zaman True döndürür. UNIX sistemlerinde IsReady özelliği daima True olarak döner.

Aşağıdaki kod **IsReady** özelliğinin kullanımını gösterir.

```
Sub SurucuBilgisi(surucuYolu)

Dim fs, d, s, t

Set fs = CreateObject("Scripting.FileSystemObject")

Set d = fs.GetDrive(surucuYolu)

Select Case d.DriveType

Case 0: t = "Bilinmiyor"

Case 1: t = "Taşınabilir"

Case 2: t = "Sabit"

Case 3: t = "Network"
```

```
Case 4: t = "CD-ROM"

Case 5: t = "RAM Disk"

End Select

s = "Sürücü" & d.DriveLetter & ": - " & t

If d.IsReady Then

s = s & vbCrLf & "Sürücü hazır."

Else

s = s & vbCrLf & "Sürücü hazır değil."

End If

Response.Write s

End Sub
```

Ekran çıktısı



Path Özelliği

Belirtilen bir dosya, dizin veya sürücü için yol tanımını döndürür.

nesne.Path

Sürücü harfleri için, asıl sürücü (root) dahil değildir. Örneğin, C sürücüsü için yol C: dir , C:\ şeklinde değildir.

RootFolder Özelliği

Belirtilen sürücünün root dizinini gösteren bir Folder nesnesini döndürür.

nesne.RootFolder

Sürücü üzerindeki tüm dosyalar ve folderlar Folder nesnesi döndürülerek erişilebilir.

SerialNumber Özelliği

Belirtilen disk için onlu seri numarasını getirir. Bu numara bir disk için tek olan tanımlayıcı olarak kullanılır.

nesne.SerialNumber

Aşağıdaki kod SerialNumber özelliğinin kullanımını gösterir.

```
Sub SurucuBilgisi(surucuYolu)

Dim fs, d, s, t

Set fs = CreateObject("Scripting.FileSystemObject")

Set d = fs.GetDrive(fs.GetDriveName(fs.GetAbsolutepathname(surucuYolu)))

Select Case d.DriveType

Case 0: t = "Bilinmiyor"

Case 1: t = "Taşınabilir"

Case 2: t = "Sabit"

Case 3: t = "Network"

Case 4: t = "CD-ROM"

Case 5: t = "RAM Disk"

End Select

s = "Sürücü " & d.DriveLetter & ": - " & t

s = s & vbCrLf & "SN: " & d.SerialNumber
```

```
Response.Write s

End Sub
```

ShareName Özelliği

Uzak disk sürücü için evrensel isimlendirme anlaşmasına (UNC) göre ağ adını getirir. Uzak bir sürücü (DriverType özelliği 3 olmalı) ile çalışma yapıldığında kullanılır.

```
nesne.ShareName
```

Eğer nesne bir ağ sürücüsü değilse, ShareName özelliği bir sıfır uzunluklu bir string ("") döndürür.

Aşağıdaki kod ShareName özelliğinin kullanımını gösterir.

```
Sub SurucuBilgisi(surucuYolu)

Dim fs, d, s

Set fs = CreateObject("Scripting.FileSystemObject")

Set d = fs.GetDrive(fs.GetDriveName(fs.GetAbsolutepathname(surucuYolu)))

s = "Sürücü " & d.DriveLetter & ": - " & d.ShareName

Response.Write s

End Sub
```

TotalSize Özelliği

Belirtilen sürücünün toplam alanını byte olarak getirir.

```
nesne.TotalSize
```

Aşağıdaki kod TotalSize özelliğinin kullanımını gösterir.

```
Sub DiskBosAlani(surucuYolu)

Dim fs, d, s

Set fs = CreateObject("Scripting.FileSystemObject")

Set d = fs.GetDrive(fs.GetDriveName(fs.GetAbsolutepathname(surucuYolu)))

s = "Sürücü " & d.DriveLetter & ":"
```

```
s = s & vbCrLf
s = s & "Toplam : " & FormatNumber(d.TotalSize/1024, 0) & " Kbytes"
s = s & vbCrLf
s = s & "Mevcut : " & FormatNumber(d.AvailableSpace/1024, 0) & " Kbytes"
Response.Write s
End Sub
```

VolumeName Özelliği

Belirtilen sürücünün disk etiketini ayarlar veya döndürür.

```
nesne. VolumeName [ = yeniadı]
```

Aşağıdaki kod Drive nesnesinin kullanılarak sürücü özelliklerine nasıl erişildiğini gösterir.

```
Sub BosDiskAlani(surucuYolu)
Dim fs, d, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set d = fs.GetDrive(fs.GetDriveName(surucuYolu))
s = "Drive " & UCase(surucuYolu) & " - "
s = s & d.VolumeName  & vbCrLf
s = s & "Boş alan miktarı: " &
FormatNumber(d.FreeSpace/1024, 0)
s = s & " Kbytes"
MsgBox s
End Sub
```

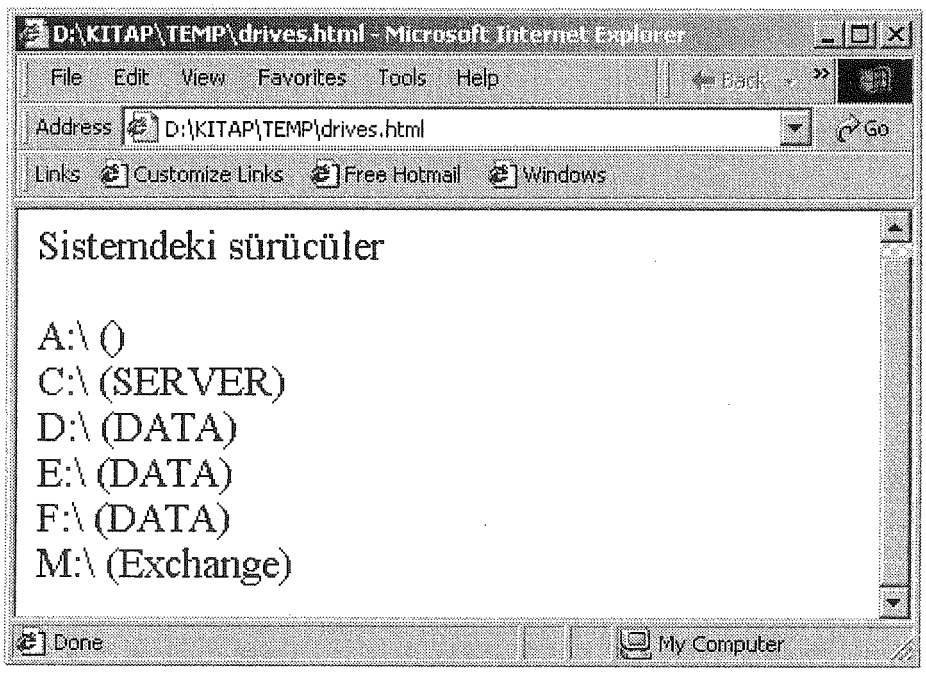
Drives Collection Nesnesi

Drives Collection nesnesi sistem üzerindeki mevcut tüm disk sürücülerinin koleksiyonunu döndürür. Bu koleksiyon FileSystemObject nesnesinin Drives özelliğinin içinden alınır. Tüm mevcut sürücülerin koleksiyonu sadece okunabilir.

Aşağıdaki kod Drives koleksiyonunu alır, sistemimiz üzerinde yer alan mevcut sürücülerin (Disk, CD-ROM gibi) bir listesini oluşturur ve tarayıcıda sonuçları gösterir.

```
<SCRIPT LANGUAGE="VBScript">

Dim dosyasis, surucu, surkoll, surlist, etiket

Set dosyasis = CreateObject("Scripting.FileSystemObject")

Set surkoll = dosyasis.Drives

Document.Write "Sistemdeki sürücüler " & "<BR><BR>"

For Each surucu In surkoll

If surcu.IsReady Then

etiket = surucu.VolumeName

End If

Document.Write surucu.DriveLetter & ":\  ("  & etiket & ") " & "<BR>"

Next
</SCRIPT>
```

Örnek script kodumuzun ürettiği tarayıcı görüntüsü aşağıdaki şekildedir.



Drives Collection Nesne Özellikleri

Count Özelliği

Koleksiyonda (mevcut yerel ve uzak sürücülerin sayısı) var olan Drive nesnelerinin kaç tane olduğunu bize veren bir tamsayı döndürür.

```
nesne.Count
```

Item Özelliği

Item özelliği belirli anahtar argüman ile gösterilen koleksiyondaki bir maddenin değerini alıp getirmemize veya o değeri maddedegeri kullanılarak ayarlamamıza izin verir.

```
nesne.Item(anahtar) [ = maddedegeri]
```

Aşağıdaki kod Drives collection nesnesinin nasıl getirileceğini ve For Each...Next ifadesinin kullanılarak koleksiyon içinde nasıl ardışık işlem yapılacağını gösterir.

```
Sub surucuListesi

Dim fs, d, dc, s, n

Set fs = CreateObject("Scripting.FileSystemObject")

Set dc = fs.Drives

For Each d in dc

s = s & d.DriveLetter & " - "

If d.DriveType = Remote Then

n = d.ShareName

Else

n = d.VolumeName

End If

s = s & n & vbCrLf

Next

MsgBox s

End Sub
```

Err Nesnesi

Çalışma-zamanı hataları hakkında bilgi içerir. Çalışma-zamanı hatalarını temizleme ve üretme için Clear ve Raise metodlarını kabul eder.

Err nesnesi global kapsamı ile bir ana nesnedir, bu yüzden ayrıca kodunuz içinde bir tutamaç oluşturmanıza gerek yoktur. Err nesnesinin özellikleri bir hatanın üretimiyle ayarlanır.

Err nesnesinin varsayılan özelliği Number dır. Err.Number bir tamsayı içerir. Bir çalışma-zamanı hatası meydana geldiğinde, Err nesnesinin özellikleri hataları kotarmak için kullanılabilen bilgi ve hataların tek tanımlayıcılarıyla ilgili bilgiler ile doldurulur.

Err nesnesinin özellikleri On Error Resume Next yapısından sonra sıfıra veya sıfır-uzunluklu stringler (" ") değerlerine ayarlanır. Clear metodu ayrı bir şekilde Err sıfırlamak için kullanılabilir.

Aşağıdaki örnek Err nesnesinin kullanımını gösterir.

```
<%
On Error Resume Next
Err.Raise 6       'Bir taşma hatası meydana geliyor.
MsgBox ("Error# " & CStr(Err.Number) & " " & Err.Description)
Err.Clear
' Hata temizleniyor.
%>
```

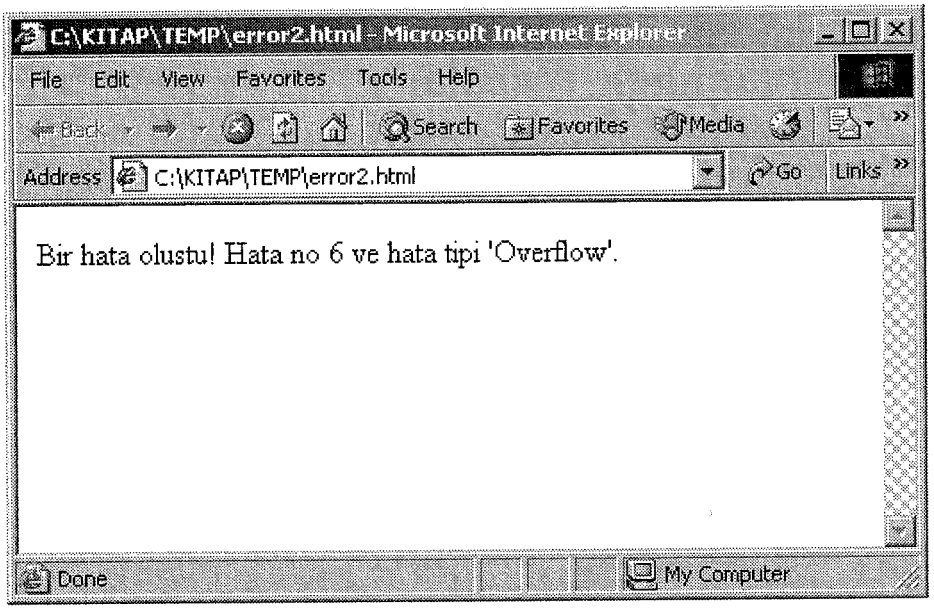
Aşağıdaki örnek Number özelliğinin değerini sınar ve, eğer sıfırdan başka bir değer içeriyorsa, tarayıcıda detayını gösterir.

```
<%
Dim sayı, hata
On Error Resume Next
Err.Raise 6
sayı = Err.number
hata = Err.description
If sayı <> 0 Then
```

```
Document.Write "Bir hata oluştu! Hata no" & sayı & " ve hata tipi'" & hata & "'."
End If
%>
```

Çıktısı:

" Bir hata oluştu! Hata no 6 ve hata tipi 'Overflow'."



Err Nesne Özellikleri

Description Özelliği

Bu özellik bir hatanın tanımını kısaca text olarak içeren bir string döndürür veya ayarlar.

```
nesne.Description [ = string]
```

Description özelliği hataların kısa bir tanımını içerir. Bu özelliği sizin bir hatayı kotarmak istemediğinizde veya kotaramadığınızda kullanıcıya bir uyarı mesajı vermek için kullanabilirsiniz. Kullanıcı tanımlı bir hata oluşturduğunuzda, bu özelliğe sizin hatanızın kısa bir tanımını atayabilirsiniz. Eğer Description içeriği boş ise, ve VBScript Err Nesnesi Number özelliğinin değeri bir VBScript çalışma zamanı hatasına karşılık gelir, hata ile ilişkilendirilmiş tanıtıcı bilgi döndürülür.

HelpContext Özelliği

Bu özellik HelpFile özelliği ile belirtilen bir yardım konusu için bir bağlam ID döndürmek veya ayarlamak için kullanılır.

nesne.HelpContext [= baglamID]

Eğer bir yardım dosyası VBScript Err nesnesi HelpFile özelliği içinde belirtilmiş ise, HelpContext özelliği tanımlanan yardım konusunun otomatik olarak görüntülenmesi için kullanılır. Eğer HelpFile ve HelpContext boş ise, VBScript Err nesnesi Number özelliğinin değeri kontrol edilir. Eğer o bir VBScript çalışma zamanı hatası değerine karşılık gelirse, o zaman hata için VBScript Help context ID kullanılır. Eğer Number özelliği bir VBScript hatasına karşılık gelmez ise, Vbscript yardım dosyası için içerik gösterilir.

HelpFile Özelliği

Bu özellik bir yardım dosyası yol tanımını getirmek veya bunu tanımlamak için kullanılır.

nesne.HelpFile [= baglamID]

HelpFile değişken tipi text değişkendir (string) , HelpContext değişken tipi ise uzun tamsayıdır (long Integer).

Err sözcüğü üretilen herhangi bir hataya işaret eder. Örneğin , yardım dosyamızın adı Yardim.Hlp olsun ve C:\WINDOWS dizini altında olduğunu varsayalım:

```
Err.HelpFile = "C:\Windows\Yardim.Hlp"

Err.Helpcontext = 12440
```

Eğer yardım dosyasının (.HLP) yolu (path) belirtilmezse , VBScript yardım dosyası görüntülenecektir.

Eğer Internet Explorer yardım dosyasını herhangi bir durumda görüntülemek için aşağıdaki kod satırını yazarız.

```
Err.HelpFile = "C:\WINDOWS\Iexplore.hlp"
```

Number Özelliği

Bu özellik belirli bir çalışma zamanı hatasına bağlı olan değeri ayarlamak veya alıp getirmek için kullanılır.

nesne.Number [= hatanumarasi]

Number özelliği Err nesnesinin varsayılan özelliğidir. Bu yüzden, Number sözcüğünü

kullanmanıza gerek yoktur. Err nesnesine bir hata kodu atadığınızda Number özelliğine atanmış demektir.

```
Err.Number = 134
```

veya

```
Err = 134
```

Örneğin,

```
<HTML>
<HEAD>
<TITLE>Err.Number</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE = "VBScript">
Err.Number = 1444
Msgbox Err
</SCRIPT>
</BODY>
</HTML>
```

Source Özelliği

Bu özellik bir hataya neden olan uygulama veya nesneyi belirlememize izin verir ve sınıf adını içeren bir string döndürür.

```
nesne.Source [ = stringifade]
```

stringifade hata üretilen uygulamayı gösteren bir string ifade.

Err Nesne Metodları

Clear Metodu

Bu metod hata nesnesinin ayarlanmış değerlerini temizlemek için kullanılır.

nesne.Clear

Raise Metodu

Bu metod bir VBScript çalışma zamanı hatası oluşturmak için kullanılır.

nesne. Raise (number[, source, description, helpfile, helpcontext])

File Nesnesi

File nesnesi bir dosyanın değişik özelliklerine erişmek ve onları düzenlememize izin verir. Bu nesne bir dosyanın tüm özelliklerine ulaşmanıza izin verir.

Aşağıdaki kod FileSystemObject nesnesinin GetFile metodunu bir File nesnesi oluşturmak ve onun özelliklerini göstermek için kullanır.

```
Sub dosyaBilgisi(dosyaBelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyaBelirtimi)
s = f.DateCreated
MsgBox s
End Sub
```

File Nesne Özellikleri

Attributes Özelliği

Bu özellik bir dosyanın değişik özelliklerini alıp getirmemize veya değiştirmemize izin verir.

nesne.Atributes [= yeniözellikler]

yeniözellikler argümanı aşağıda verilen değerlerden herhangi birini veya bunların mantıksal bir kombinasyonunu alır.

Sabit	Değer	Tanımı
Normal	0	Normal dosya. Özellikler ayarlanmamış.
ReadOnly	1	Read-only dosya. Özellik read/write.
Hidden	2	Gizli dosya. Özellik read/write.
System	4	Sistem dosyası. Özellik read/write.
Volume	8	Disk sürücü etiketi. Özellik read-only.
Directory	16	Folder veya dizin. Özellik read-only.
Archive	32	En son yedekten sonra dosya değişti. Özellik read/write.
Alias	64	Bağ veya kısayol. Özellik read-only.
Compressed	128	Sıkıştırılmış dosya. Özellik read-only.

Aşağıdaki kod bir dosya ile Attributes özelliğinin kullanımını gösterir.

```
Sub ArchiveBitiniTemizle(dosyabelirtimi)

Dim fs, f, r

Set fs = CreateObject("Scripting.FileSystemObject")

Set f = fs.GetFile(fs.GetFileName(dosyabelirtimi))

If f.attributes and 32 Then

r = MsgBox("Archive bit ayarlı, temizlemek istermisiniz?", vbYesNo,
"Ayarla/Temizle Archive Bit")

If r = vbYes Then

f.attributes = f.attributes - 32

MsgBox "Archive bit temizlendi."

Else

MsgBox "Archive bit ayarlı kaldı."

End If

Else

r = MsgBox("Archive bit ayarlı değil. Ayarlamak istermisiniz?", vbYesNo,
"Ayarla/Temizle Archive Bit")

If r = vbYes Then

f.attributes = f.attributes + 32
```

```
MsgBox "Archive bit ayarlı."
Else
MsgBox "Archive bit temiz ."
End If
End If
End Sub
```

DateCreated Özelliği

Bu özellik dosyanın oluşturulduğu tarih ve zamanını getirir.

```
nesne.DateCreated
```

Aşağıdaki kod bir dosya ile DateCreated özelliğinin kullanımını gösteriyor.

```
Sub DosyaBilgisi(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
s = "Dosya " & f.DateCreated & "tarihinde oluşturuldu."
Response.Write s
End Sub
```

DateLastAccessed Özelliği

Dosyaya erişilen en son tarih ve zamanı getirir.

```
nesne.DateLastAccessed
```

Aşağıdaki kod bir dosya ile DateLastAccessed özeliğinin kullanımını gösteriyor.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
```

```
s = UCase(dosyabelirtimi) & vbCrLf
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Dosya erişim bilgisi"
End Sub
```

DateLastModified Özelliği

Bu özellik dosyayı en son değiştirme tarihi ve zamanını döndürür.

```
Nesne.DateLastModified
```

Aşağıdaki kod bir dosya ile DateLastModified özelliğinin kullanımını gösterir.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
s = UCase(dosyabelirtimi) & vbCrLf
s = s & "Oluşturulma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Dosya erişim bilgisi"
End Sub
```

Drive Özelliği

Dosyanın yerleştirildiği yerdeki sürücünün sürücü harfini getirir.

```
nesne.Drive
```

Aşağıdaki kod Drive özelliğinin kullanımını gösterir.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
```

```
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
s = f.Name & " on Drive " & UCase(f.Drive) & vbCrLf
s = s & " Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & " En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & " En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Dosya erişim bilgisi"
End Sub
```

Name Özelliği

Belirtilen dosyanın adını değiştirmemize veya getirmemize izin verir.

```
nesne.Name [ = yeniadi]
```

Aşağıdaki kod Name özelliğinin kullanımını gösteriyor.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
s = f.Name & " Sürücü " & UCase(f.Drive) & vbCrLf
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Dosya erişim bilgisi"
End Sub
```

ParentFolder Özelliği

Bu özellik belirtilen dosyaya bağlı ana dizin için Folder nesnesini getirir.

nesne.ParentFolder

Aşağıdaki kod bir dosya ile ParentFolder özelliğinin kullanımını gösteriyor.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
s = UCase(f.Name) & " in " & UCase(f.ParentFolder) & vbCrLf
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Dosya erişim bilgisi"
End Sub
```

Path Özelliği

Bu özellik bir dosyanın yolunu (path) döndürür.

nesne.Path

Aşağıdaki kod bir File nesnesi ile Path özelliğinin kullanımını gösteriyor.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, d, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
s = UCase(f.Path) & vbCrLf
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Dosya erişim bilgisi"
End Sub
```

ShortName Özelliği

Bir dosyanın kısa versiyonunu (Employees.html şeklini alır Employ~1.htm) döndürür.

nesne.ShortName

Aşağıdaki kod bir File nesnesi ile ShortName özelliğinin kullanımını gösteriyor.

```
Sub KisaAd(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
s = "Kısa ad " & "" & UCase(f.Name)
s = s & "" & vbCrLf
s = s & "" & f.ShortName & " dır."
Response.Write s, 0, "Kısa ad bilgisi"
End Sub
```

ShortPath Özelliği

Dosya yolunun (bu yukarıdaki gibi kesilen dosya ve dizi adları ile olan yoldur)kısa versiyonunu döndürür.

nesne.ShortPath

Aşağıdaki kod bir File nesnesi ile ShortPath özelliğinin kullanımını gösteriyor.

```
Sub KisaYol(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(dosyabelirtimi)
s = "Kısa yol " & "" & UCase(f.Name)
s = s & "" & vbCrLf
s = s & "" & f.ShortPath & " dır."
Response.Write s, 0, "Kısa yol bilgisi"
End Sub
```

Size Özelliği

Bir dosyanın büyüklüğünü byte olarak döndürür.

```
nesne.Size
```

Burada nesne bir File nesnesidir.

Type Özelliği

Dosya tip tanımını içeren bir string döndürür. sonu .TXT ile biten dosyalar için "Text Document" olarak döner.

```
nesne.Type
```

File Nesnesi Metodları

Copy Metodu

Bu metod seçilen bir dosyayı belirtilen hedefe kopye eder.

```
nesne.Copy hedef[, overwrite]
```

Delete Metodu

Metod belirtilen File nesnesine bağlı dosyayı silmek için kullanılır.

```
nesne.Delete [force]
```

Move Metodu

Bu metod yeni bir hedefe belirtilen File nesnesine bağlı dosyayı taşımak için kullanılır.

```
nesne.Move hedef
```

OpenAsTextStream Metodu

Bu metod belirli bir dosyayı açar ve üzerinde düzenleme yapılabilen (okuma, yazma ve açma gibi) bir TextStream nesnesi tutamacı döndürür.

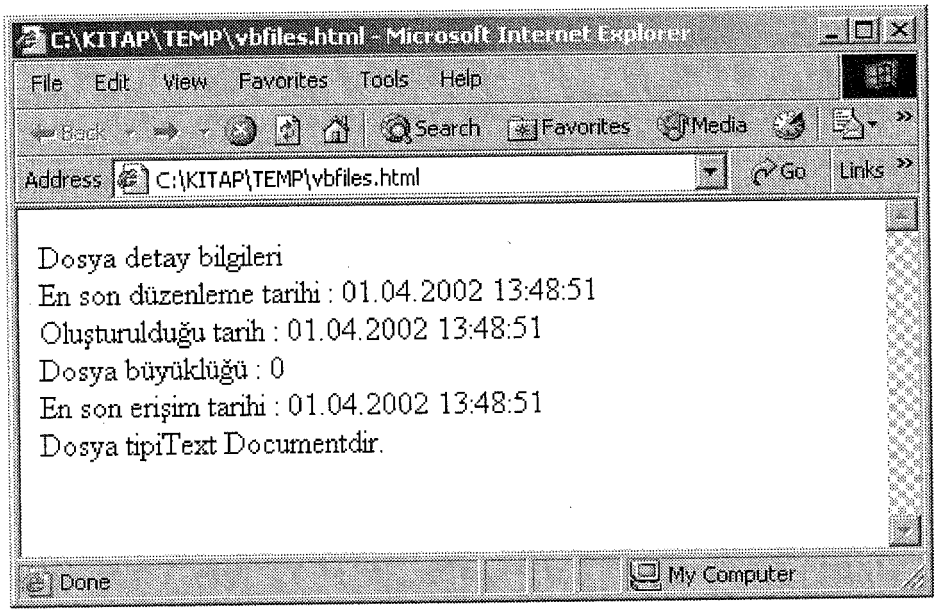
```
nesne.OpenAsTextStream([iomode [, format]])
```

Bu örnekte size bir dosyanın değişik özelliklerinin nasıl öğrenileceğini gösterecektir.

```
<SCRIPT LANGUAGE="VBScript">
```

```
<!- -

Dim nesneFSO , dosya

Set nesneFSO = CreateObject("Scripting.FileSystemObject")

Set dosya = nesneFSO.GetFile("d:\cihazlar.txt")

Document.write ("Dosya detay bilgileri")

Document.write ("<br>")

Document.write ("En son düzenleme tarihi : " & dosya.DateLastModified)

Document.write ("<br>")

Document.write ("Oluşturulduğu tarih : " & dosya.dateCreated)

Document.write ("<br>")

Document.write ("Dosya büyüklüğü : " & dosya.Size)

Document.write ("<br>")

Document.write ("En son erişim tarihi : " & dosya.DateLastAccessed)

Document.write ("<br>")

Document.write ("Dosya tipi " & dosya.Type & "dir. ")

//-->

</SCRIPT>
```

yazdığımız örnekte d sürücüsü üzerindeki cihazlar.txt adlı dosyayı açıyoruz ve sonra onun hakkında biraz bilgi alıyoruz. Buradaki özellikler File nesnesinin tüm özelliklerini içermez.



Files Collection Nesnesi

Bu nesne File nesnelerinin bir kümesini içerir ve genellikle bir Folder nesnesi gibi, başka bir nesnenin bir özelliği olarak saklanır.

Aşağıdaki kod bir Files koleksiyonu nesnesinin nasıl getirileceğini ve For Each...Next ifadesinin kullanılarak koleksiyon üzerinde nasıl ardışık işlem yapılacağını gösterir.

```
Sub folderListesi(folderBelirtimi)

Dim fs, f, f1, fc, s

Set fs = CreateObject("Scripting.FileSystemObject")

Set f = fs.GetFolder(folderBelirtimi)

Set fc = f.Files

For Each f1 in fc

s = s & f1.name

s = s & vbCrLf

Next

MsgBox s

End Sub
```

Files Collection Nesnesi Özellikleri

Count Özelliği

Koleksiyonda kaç tane File nesnesi olduğunu bize söyleyen bir tamsayı döndürür.

nesne.Count

nesne bir koleksiyon adıdır.

Aşağıdaki kod Count özelliğinin kullanımını gösteriyor.

```
Dim a, d, i

Set d = CreateObject("Scripting.Dictionary")

d.Add "a", "Opel"

d.Add "b", "WV"

d.Add "c", "BMW"

a = d.Keys

For i = 0 To d.Count -1

Print a(i)

Next
...
```

Item Özelliği

Bu özellik belirli bir anahtar argüman ile gösterilen koleksiyonda bir maddenin değerini alıp getirmemize ve maddedegeri kullanımıyla değeri ayarlamamızada izin verir.

nesne.Item(anahtar) [= maddedegeri]

FileSystemObject Nesnesi

Bu nesne bir bilgisayarın dosya sistemine erişim kazanmak için kullanılır. O yeni dosyalar oluşturabilir ve var olanlarada erişebilir.

Aşağıdaki kod okunabilen veya yazılabilen bir TextStream nesnesinin FileSystemObject nesnesini kullanarak nasıl getirileceğini gösterir.

```
<SCRIPT LANGUAGE="VBScript">

Set fs = CreateObject("Scripting.FileSystemObject")
```

```
Set a = fs.CreateTextFile("c:\deneme.txt", True)

a.WriteLine("Bu bir deneme.")

a.Close

</SCRIPT>
```

Yukarıda verilen kod içinde, CreateObject fonksiyonu FileSystemObject (fs) nesnesini döndürür. CreateTextFile metodu o zaman bir TextStream nesnesi (a) olarak dosyayı oluşturur ve WriteLine metodu oluşturulan text dosyaya bir satır yazar.

Başka bir örnek

Aşağıdaki örnek kod FileSystemObject nesnesinin CreateTextFile metodunu bir text dosya oluşturmak (c:\dosya.txt) ve sonra text bilgi yazmak için kullanıyor.

```
<%

Dim dosyasistemi, textdosya, adgetir, yol

Set dosyasistemi = CreateObject("Scripting.FileSystemObject")

Set textdosya = dosyasistemi.CreateTextFile("c:\dosya.txt", True)

yol = dosyasistemi.GetAbsolutePathName("c:\dosya.txt")

adgetir = dosyasistemi.GetFileName(yol)

textdosya.WriteLine("Text bilginiz burada olur.")

textdosya.Close

If dosyasistemi.FileExists(yol) Then

  Response.Write ("Dosyanız, '" & adgetir & "', oluşturuldu.")

End If

%>
```

Çıkış :

```
"Dosyanız, 'dosya.txt', oluşturuldu."
```

FileSystemObject Nesne Özellikleri

Drives özelliği

Bir bilgisayar üzerindeki tüm Drive nesnelerini içeren bir Drives koleksiyonu döndürür.

[surucukoleksiyonu =] nesne.Drives

For Each...Next yapısını kullanarak Drives üzerinde ardışıl işlemler yapabilirsiniz.

```
Sub SurucuListesi
Dim fs, d, dc, s, n
Set fs = CreateObject("Scripting.FileSystemObject")
Set dc = fs.Drives
For Each d in dc
s = s & d.DriveLetter & " - "
If d.DriveType = 3 Then
n = d.ShareName
Else
n = d.VolumeName
End If
s = s & n & vbCrLf
Next
Response.Write s
End Sub
```

FileSystemObject Nesnesi Metodları

BuildPath Metodu

Bu metod var olan bir yol üzerine bir isim açmak için kullanılır.

[yenitamyol =]nesne.BuildPath(yol, isim)

nesne gerekli olup bir FileSystemObject nesnesidir.

CopyFile Metodu

Bu metod bir yerden (kaynak) diğerine (hedef) bir veya daha çok dosyayı kopye etmemize izin verir.

```
nesne.CopyFile kaynak, hedef [, overwrite]
```

CopyFolder Metodu

Bu metod bir yerden (kaynak) diğer bir yere (hedef) bir veya daha çok dizinleri ve tüm içeriklerini, dosyalar ve alt dizinlerde dahil, kopye eder.

```
nesne.CopyFolder kaynak, hedef, [, overwrite]
```

CreateFolder Metodu

Bu metod bir dizin oluşturmamıza izin verir.

```
nesne.CreateFolder dizinadi
```

CreateTextFile Metodu

Bir text dosyası oluşturur ve dosyadan okumak veya yazmak için kullanılan bir TextStreamObject nesnesi döndürür.

```
nesne.CreateTextFile dosyadi [, overwrite[, unicode]]
```

Aşağıdaki kod CreateTextFile metodu ile bir text dosyanın nasıl oluşturulacağını ve açılacağını gösteriyor.

```
Sub DosyaOlustur

Set fs = CreateObject("Scripting.FileSystemObject")

Set a = fs.CreateTextFile("c:\dosya.txt", True)

a.WriteLine("Bu bir deneme.")

a.Close

End Sub
```

DeleteFile Metodu

Bu metod belirtilen bir dosyayı veya dosyaları(wilcard kullanılarak) siler.

```
nesne.DeleteFile dosya [, force]
```

DeleteFolder Metod

Bu metod belirtilen bir dizini siler, buna tüm dosyalar ve alt dizinler de dahildir.

```
nesne.DeleteFolder dizin [, force]
```

DriveExists Metodu

Bu metod belirtilen bir sürücünün mevcut olup olmadığını sınar. O eğer mevcut ise True yoksa False döndürür.

nesne.DriveExists(sürücü)

FileExists Metodu

Belirtilen dosyanın var olup olmadığını sınar. Eğer mevcut ise True yoksa False döndürür.

nesne.FileExists(dosya)

FolderExists Metodu

Belirtilen dizinin var olup olmadığını sınar. Eğer mevcut ise True yoksa False döndürür.

nesne.FolderExists(dizin)

GetAbsolutePathName Metodu

Bu metod belirtilen yol stringi için sürücünün root dizininden tam yolu getirir.

nesne.GetAbsolutePathName(yol)

GetBaseName Metodu

Bu metod belirtilen bir yoldaki dosya veya dizinin taban adını getirir.

nesne.GetBaseName(yol)

GetDrive Metodu

Bu metod desteklenilen bir yoldaki sürücüye benzeyen bir Drive nesnesi döndürür.

nesne.GetDrive(sürücü)

GetDriveName Metodu

Bu metod desteklenen bir yoldaki sürücünün adını içeren bir string getirir.

nesne.GetDriveName(yol)

GetExtensionName Metodu

Desteklenilen bir yoldaki en son bileşenin uzantı adını içeren bir string döndürmek için kullanılır.

nesne.GetExtensionName(yol)

GetFile Metodu

Belirtilen dosya adı için File nesnesini döndürür.

nesne.GetFile(dosyadı)

GetFileName Metodu

Bu metod desteklenen yolun en son dosyasının veya dizininin adını döndürmek için kullanılır.

nesne.GetFileName(yol)

GetFolder Metodu

Bu metod dizin parametresindeki belirtilen dizin için bir Folder nesnesini döndürür.

nesne.GetFolder(dizi)

GetParentFolderName Metodu

Belirtilen bir yoldaki en son dosya veya dizinin ana dizininin adını içeren bir string döndürür.

nesne.GetParentFolderName(yol)

GetSpecialFolder Metodu

Özel dizinlerden (\Windows, \System, veya \TMP) birine yol döndürür.

nesne.GetSpecialFolder(dizin)

GetTempName Metodu

Bu metod geçici bir dosya için rasgele bir dosyaadı üretmek için kullanılır.

nesne.GetTempName

nesne seçimlilik argüman olup daima bir FileSystemObject nesnesi adıdır. GetTempName metodu bir dosya oluşturmaz . O sadece CreateTextFile metodu ile bir dosya oluşturmak için kullanılabilen geçici bir dosya adı sağlar.

MoveFile Metodu

Bir veya daha çok dosyayı bir yerden başka bir yere taşır.

nesne.MoveFile kaynak, hedef

nesne gerekli olup bir FileSystemObject nesnesi adıdır. Kaynak taşınacak olan dosya veya dosyaların yol tanımlarını gösterir. hedef ise taşınacak dosya veya dosyaların nereye kopye edileceğini gösterir yol.

MoveFolder Metodu

Bir veya daha çok dizini bir yerden başka bir yere taşır.

nesne.MoveFolder kaynak, hedef

nesne gerekli olup bir FileSystemObject nesnesi adıdır. kaynak taşınacak olan folder veya folderların yol tanımıdır. hedef taşınacak olan folder veya folderların nereye taşınacağı yolunu gösterir.

OpenTextFile Metodu

Dosyaadı parametresinde belirtilen dosyayı açar ve o dosya için TextStreamObject nesnesinin bir tutamacını döndürür.

nesne.OpenTextFile(dosyaadı [, iomode[, create[, format]]])

nesne gerekli olup bir FileSystemObject nesnesi adıdır. dosyaadi gerekli olup açılacak olan dosyayı tanıtan bir string ifadedir. iomode giriş/çıkış modunu gösterir. format TriState değerlerinden birini alabilir, bu açılacak olan dosyanın formatını gösterir. Eğer bu değer girilmez ise o zaman dosya ASCII olarak açılır.

iomode argümanı aşağıdaki ayarlardan birini alabilir.

Sabit	Değer	Tanımı
ForReading	1	Dosyayı okuma modunda açar. Dosyaya yazamazsınız.
ForAppending	8	Dosyayı açar ve en dosya sonuna konumlanır.

Format argümanı aşağıdaki ayarlardan herhangi birini alabilir.

Sabit	Değer	Tanımı
TristateUseDefault	-2	Bilgisayar ayarlarını kullanarak dosyayı açar.
TristateTrue	-1	Bir dosyayı Unicode olarak açar.
TristateFalse	0	Bir dosyayı ASCII olarak açar.

Aşağıdaki kod yazma modunda bir dosyayı açmak için OpenTextFile metodunun kullanımını gösterir.

```
Sub TextDosyaAc

Const ForReading = 1, ForWriting = 2, ForAppending = 3

Dim fs, f

Set fs = CreateObject("Scripting.FileSystemObject")

Set f = fs.OpenTextFile("c:\dosya.txt", ForAppending,TristateFalse)

f.Write "Merhaba!"

f.Close

End Sub
```

Aşağıdaki örnek bir dosyanın kopye edilmesini yapar.

```
<SCRIPT LANGUAGE="VBScript">

Dim nesneFSO

Set nesneFSO = CreateObject("Scripting.FileSystemObject")

nesneFSO.CopyFile "d:\dosya.txt" , "d:\dosya.bak"

Set nesneFSO = Nothing

</SCRIPT>
```

Örnekte d sürücüsü üzerindeki dosya.txt dosyasını yine d sürücüsü üzerindeki dosya.bak dosyasına kopye ediyoruz, bu bir anlamda dosyamızın bir yedeğini almak demektir. Bu bilinmesi gereken faydalı bir tekniktir.

CopyFile metodunun yazım şekli

```
CopyFile kaynakdosya, hedefdosya [, overwrite]
```

overwrite seçimlilik olup ilk değeri True

Folder Nesnesi

Folder nesnesi bir dizinin değişik özelliklerine ulaşmanıza ve düzenlemenize izin verir.

Aşağıdaki kod FileSystemObject nesnesinin GetFolder metodunu bir Folder nesnesi getirmek ve onun özelliklerini göstermek için kullanılır.

```
Sub folderBilgisi(folderbelirtimi)

Dim fs, f, s,

Set fs = CreateObject("Scripting.FileSystemObject")

Set f = fs.GetFolder(folderbelirtimi)

s = f.DateCreated

MsgBox s

End Sub
```

Folder Nesnesi Özellikleri

Attributes Özelliği

Bu özellik bir dosyanın değişik özelliklerini getirip değiştirmenize izin verir.

```
nesne.Atributes [ = yeniözellikler]
```

nesne gerekli olup bir Folder nesnesi adıdır. Yeniözellikler seçimlilik olup belirtilen nesnenin özellikleri için yeni değerdir.

Yeniözellikler argümanı aşağıdaki değerlerden herhangi birini alabilir veya aşağıdaki değerlerin herhangi bir mantıksal kombinasyonunu alabilirler.

Sabit	Değer	Tanımı
Normal	0	Normal dosya. özellikler ayarlanmamış.
ReadOnly	1	Read-only dosya. özellik read/write.
Hidden	2	Gizli dosya. özellik read/write.
System	4	Sistem dosyası. özellik read/write.

Volume	8	Disk sürücü etiketi. özellik read-only.
Directory	16	Folder veya dizin. özellik read-only.
Archive	32	Dosya en son yedeklemeden sonra değişti. özellik read/write.
Alias	64	Bağ veya kısayol. özellik read-only.
Compressed	128	Sıkıştırılmış dosya. özellik read-only.

DateCreated Özelliği

Bu özellik dizinin oluşturulduğu tarih ve zamanı getirir.

```
nesne.DateCreated
```

Aşağıdaki kod bir folder ile DateCreated özelliğinin kullanımını gösterir.

```
Sub DosyaBilgisi(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(dosyabelirtimi)
s = "Oluşturma tarihi : " & f.DateCreated
Response.Write s
End Sub
```

DateLastAccessed Özelliği

Dizine erişilen en son tarih ve zamanı döndürür.

```
nesne.DateLastAccessed
```

Aşağıdaki kod bir folder ile DateLastAccessed özelliğinin kullanımını gösterir.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(dosyabelirtimi)
s = UCase(dosyabelirtimi) & vbCrLf
```

```
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Folder erişim bilgisi"
End Sub
```

DateLastModified özelliği

Bu özellik dizinin en son değiştirildiği tarih ve zamanı döndürür.

```
nesne.DateLastModified
```

Aşağıdaki kod bir folder ile DateLastModified özelliğinin kullanımını gösterir.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(dosyabelirtimi)
s = UCase(dosyabelirtimi) & vbCrLf
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi: " & f.DateLastModified
Response.Write s, 0, "Folder erişim bilgisi"
End Sub
```

Drive Özelliği

Bir dizinin yer aldığı yerdeki sürücünün sürücü harfini döndürür.

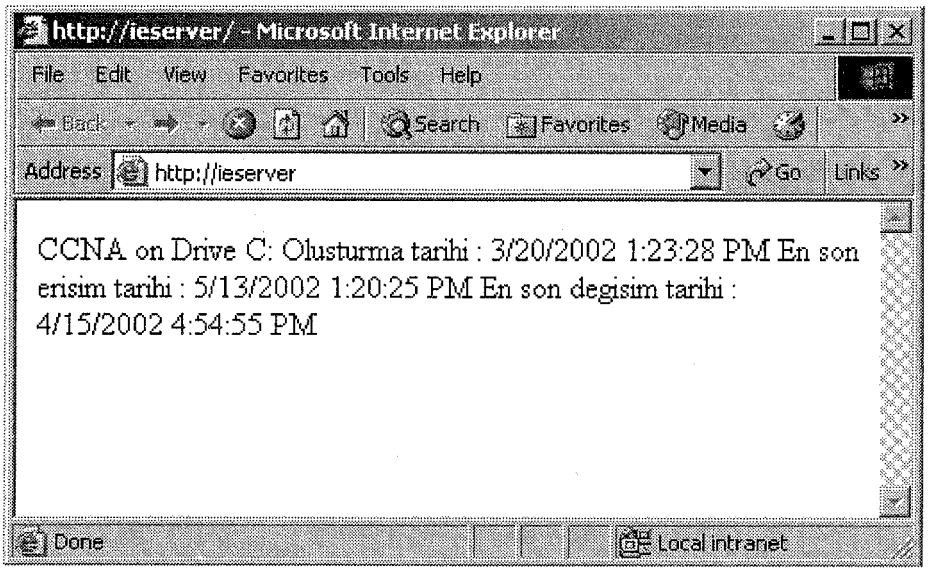
```
nesne.Drive
```

Aşağıdaki kod Drive özelliğinin kullanımını gösterir.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, f, s
```

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(dosyabelirtimi)
s = f.Name & " on Drive " & UCase(f.Drive) & vbCrLf
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Folder erişim bilgisi"
End Sub
```

Ekran çıktısı



Files Özelliği

Belirli dizin içindeki tüm File nesnelerini içeren bir Files koleksiyonu döndürür.

```
nesne.Files
```

Aşağıdaki kod Files özelliğinin kullanımını gösterir.

```
Sub DosyaListesi(folderbelirtimi)
Dim fs, f, f1, fc, s
```

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(folderbelirtimi)
Set fc = f.Files
For Each f1 in fc
s = s & f1.name
s = s & vbCrLf
Next
Response.Write s
End Sub
```

IsRootFolder Özelliği

Bir boolean değer döndürür: eğer dizin bir root dizin ise True, ve aksi taktirde False döndürür.

```
nesne.IsRootFolder
```

Aşağıdaki kod IsRootFolder özelliğinin kullanımını gösterir.

```
Dim fs
Set fs = CreateObject("Scripting.FileSystemObject")

Sub SeviyeDerinligi(yolbelirtimi)
Dim f, n
Set f = fs.GetFolder(yolbelirtimi)
If f.IsRootFolder Then
MsgBox "Belirtilen folder root folder."
Else
Do Until f.IsRootFolder
Set f = f.ParentFolder
n = n + 1
```

```
Loop

MsgBox "Belirtilen folder " & n & "  nci seviye derinlikte."

End If

End Sub
```

Name Özelliği

Bu özellik belirli dizinin adını getirip değiştirmenize izin verir.

nesne.Name [= yeniadi]

nesne gerekli olup bir Folder nesnesinin adıdır. yeniadi ise belirtilen nesneye ait olan yeni adı gösterir seçimlilik bir değerdir.

Aşağıdaki kod Name özelliğinin kullanımını gösterir.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)

Dim fs, f, s

Set fs = CreateObject("Scripting.FileSystemObject")

Set f = fs.GetFolder(dosyabelirtimi)

s = f.Name & " Sürücüde " & UCase(f.Drive) & vbCrLf

s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf

s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf

s = s & "En son değişim tarihi : " & f.DateLastModified

Response.Write s, 0, "Folder erişim bilgisi "

End Sub
```

ParentFolder Özelliği

Bu özellik belirli dizine bağlı olan ana Folder nesnesini getirir.

nesne.ParentFolder

Aşağıdaki kod bir folder ile ParentFolder özelliğinin kullanımını gösterir.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
```

```
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(dosyabelirtimi)
s = UCase(f.Name) & " içinde " & UCase(f.ParentFolder) & vbCrLf
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Folder erişim bilgisi "
End Sub
```

Path Özelliği

Bu özellik bir dizinin yolunu döndürür.

```
nesne.Path
```

Aşağıdaki kod bir Folder nesnesi ile Path özelliğinin kullanımını gösterir.

```
Sub DosyaErisimBilgisi(dosyabelirtimi)
Dim fs, d, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(dosyabelirtimi)
s = UCase(f.Path) & vbCrLf
s = s & "Oluşturma tarihi : " & f.DateCreated & vbCrLf
s = s & "En son erişim tarihi : " & f.DateLastAccessed & vbCrLf
s = s & "En son değişim tarihi : " & f.DateLastModified
Response.Write s, 0, "Folder erişim bilgisi "
End Sub
```

ShortName Özelliği

Bir dizin adının kısa versiyonunu getirir (8.3 formatı kullanarak). Örneğin, dizin adı "kasabaninsirri" olsun, bu ismin bir kısmı kesip atılır ve sonuç olarak "kasaba~1" döner.

nesne.ShortName

Aşağıdaki kod bir Folder nesnesi ile ShortName özelliğinin kullanımını gösterir.

```
Sub KisaAd(dosyabelirtimi)

Dim fs, f, s

Set fs = CreateObject("Scripting.FileSystemObject")

Set f = fs.GetFfolder(dosyabelirtimi)

s = "Kısa ad " & "" & UCase(f.Name) & "dır."

s = s & "" & vbCrLf

s = s & "is: " & "" & f.ShortName & ""

Response.Write s, 0, "Kısa ad bilgisi "

End Sub
```

ShortPath Özelliği

Dizi yolunun (bu yukarıdaki gibi kesilen herhangi bir dizi veya dosyanın adlarının parçasıdır) kısa versiyonunu döndürür.

nesne.ShortPath

Aşağıdaki kod bir Folder nesnesi ile ShortPath özelliğinin kullanımını gösterir.

```
Sub KisaYol(dosyabelirtimi)

Dim fs, f, s

Set fs = CreateObject("Scripting.FileSystemObject")

Set f = fs.GetFolder(dosyabelirtimi)

s = "Kısa yol " & "" & UCase(f.Name) & " dır."

s = s & "" & vbCrLf

s = s & "is: " & "" & f.ShortPath & ""

Response.Write s, 0, "Kısa yol bilgisi "

End Sub
```

Size Özelliği

Belirli dizinin büyüklüğünü ve içeriğini byte olarak döndürür.

nesne.Size

Aşağıdaki kod bir Folder nesnesi ile Size özelliğinin kullanımını gösterir.

```
Sub FolderBuyuklugu(dosyabelirtimi)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(dosyabelirtimi)
s = UCase(f.Name) &  f.size & " byte kullanır."
Response.Write s, 0, "Folder büyüklük bilgisi"
End Sub
```

SubFolders Özelliği

Bu özellik belirli dizindeki tüm dizinleri içeren bir Folders koleksiyonunu döndürür.

nesne.SubFolders

Aşağıdaki kod SubFolders özelliğinin kullanımını gösterir.

```
Sub FolderListesi(folderbelirtimi)
Dim fs, f, f1, s, sf
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(folderbelirtimi)
Set sf = f.SubFolders
For Each f1 in sf
s = s & f1.name
s = s & vbCrLf
Next
```

```
Response.Write s

End Sub
```

Type Özelliği

Dizin tip tanımını içeren bir string döndürür.

```
nesne.Type
```

Folder Nesnesi Metodları

Copy Metodu

Belirli dizini bir yerden diğer bir yere kopye eder.

```
nesne.Copy hedef[, overwrite]
```

nesne gerekli olup bir Folder nesnesi adıdır. hedef folder nereye kopye edilecekse orayı gösterir. overwrite değeri True ise mevcut dosya üzerine yazılır eğer False ise yazılmaz.

CreateTextFile Metodu

Bu metod bir text dosya oluşturmak için kullanılır ve dosyadan okumak ve yazmak için kullanılacak bir TextStreamObject nesnesi döndürür.

```
nesne.CreateTextFile(dosyaadi,[, overwrite[, unicode]]))
```

nesne gerekli olup bir Folder nesnesi adıdır. dosyaadi yeni oluşturulacak olan dosyayı tanıtan bir string ifadedir. overwrite eğer mevcut bir dosyanın üzerine yazılacaksa bunu gösteren bir boolean değerdir. Eğer değer True ise, dosya üzerine yazılabilir; False ise yazılamaz. unicode oluşturulacak dosyanın bir Unicode veya ASCII dosya olup olmadığını gösteren bir boolean değerdir.

Aşağıdaki kod bir dosya oluşturmak ve bir text dosyası açmak için CreateTextFile metodunun nasıl kullanılacağını gösterir.

```
Sub DosyaOlustur

Set fd = CreateObject("Scripting.Folder")

Set a = fd.CreateTextFile("c:\dosya.txt", True)

a.WriteLine("Bu bir denemedir.")
```

```
a.Close

End Sub
```

Delete Metodu

Belirli dizini siler

```
nesne.Delete [kuvvet]
```

nesne gerekli olup bir Folder nesnesi adıdır. Silinecek olan folder özelliği read-only ise o zaman kuvvet değeri True bir boolean.

Move Metodu

Belirli dizini bir yerden başka bir yere taşır.

```
nesne.Move hedef
```

nesne gerekli olup daima bir Folder nesnesi adıdır. Hedef gerekli olup folder un nereye taşınacağını gösterir. Burada willcard kullanamazsınız.

Folders Collection Nesnesi

Folder nesnesinin bir tutamacı kullanıldığı zaman, onun SubFolder özelliği o Folder içindeki tüm subfolder (Folder nesneleri) ların hepsini içeren bir Folders koleksiyonu döndürür.

Aşağıdaki kod bir Folders koleksiyonunu nasıl getireceğinizi ve onun içeriğini tarayıcıda nasıl göstereceğinizi göstermektedir.

```
<%

Dim dosyasis, demofolder, subfol, folcoll, folist

Set dosyasis = CreateObject("Scripting.FileSystemObject")

Set demofolder = dosyasis.GetFolder(folderbelirtimi)

Set folcoll = demofolder.SubFolders

For Each subfol in folcoll

   folist = folist & subfol.Name

   folist = folist & "<BR>"
```

```
Next

Response.Write folist

%>
```

Folders Collection Nesnesi Özellikleri

Count Özelliği

Koleksiyon içinde kaçtane Folder nesnesi olduğunu gösteren bir tamsayı döndürür.

```
nesne.Count
```

Item Özelliği

Item özelliği bir anahtar argüman belirtilerek işaret edilen koleksiyonda bir maddenin değerini alıp getirmek ve maddeDegeri kullanılarak değer ayarlaması yapmamızada izin verir.

```
nesne.Item(anahtar) [ = maddeDegeri]
```

Folders Collection Nesnesi Metodları

Add Metodu

Bu metod bir Folders Koleksiyonuna yeni bir Folder eklemek için kullanılır.

```
nesne.Add("folderadi")
```

Match Nesnesi

Düzenli bir ifade (Regular Expression) uyumunun yanlızca okunabilir bir özelliğine erişim sağlar. Bir Match nesnesi Match nesnelerinin bir koleksiyonunu güncel olarak döndüren RegExp nesnesinin Execute metodu kullanılarak yanlızca oluşturulabilir. Tüm Match nesnesi özellikleri yanlızca okunabilirdir.

Bir düzenli ifade yürütüldüğü zaman, sıfır veya daha çok Match nesnesi sonuçlanabilir. Herbir Match nesnesi düzenli ifade tarafından bulunan stringe erişimi sağlar, string uzunluğu, ve uyumun nerede bulunduğunu gösteren bir indis değeri.

Aşağıdaki kod Match nesnesinin kullanımını gösteriyor:

```
Function RegExpTest(desen, str1)

Dim regEx, Match, Matches
```

```
' Değişken oluşturma.
Set regEx = New RegExp
' Düzenli İfade oluşturma.
regEx.Pattern = desen
' Desen ayarlama.
regEx.IgnoreCase = True
' Büyük-küçük harf duyarlılığı yok.
regEx.Global = True
' Glogal erişime izin ver.
Set Matches = regEx.Execute(str1)
' Aramayı yürüt.
For Each Match in Matches
  RetStr = RetStr & "Uyum " & I & " pozisyonunda bulundu"
  RetStr = RetStr & Match.FirstIndex & ". Uyum değeri'"
  RetStr = RetStr & Match.Value & "'." & vbCRLF
Next
RegExpTest = RetStr
End Function
MsgBox(RegExpTest("is.", "IS1 is2 IS3 is4"))
```

Aşağıdaki kod örnek Microsoft tarafında yayınlanmış çalışan bir programın basitleştirilmiş halidir.

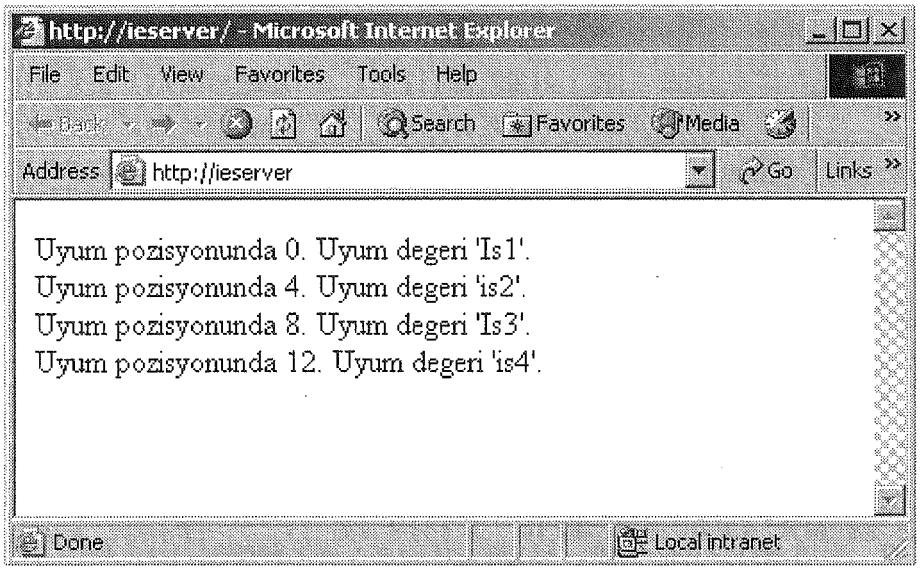
```
<%
' Bu sub prosedür uyumlar bulur
Sub RegExpTest(strMatchPattern, strPhrase)
   ' değişkenler oluştur
   Dim objRegEx, Match, Matches, StrReturnStr
```

```
'RegExp nesnesinin tutamacını oluştur
Set objRegEx = New RegExp
' tüm uyumları bul
objRegEx.Global = True
'büyük küçük harfe duyarlı
objRegEx.IgnoreCase = True
' pattern değerini ayarla
objRegEx.Pattern = strMatchPattern
' uyumlar koleksiyonunu oluştur
Set Matches = objRegEx.Execute(strPhrase)
' tüm uyumları yazdır
For Each Match in Matches
   strReturnStr = "Uyum pozisyonunda  "
   strReturnStr = strReturnStr & Match.FirstIndex & ". Uyum değeri '"
   strReturnStr = strReturnStr & Match.value & "'." & "<BR>" & VBCrLf
   ' çıktıyı yaz
   Response.Write(strReturnStr)
: Next
End Sub
' alt rutini çağır
RegExpTest "is.", "Is1 is2 Is3 is4"
%>
```

Çıkış:

```
Uyum 0. pozisyonda. Uyum değeri 'Is1'.
Uyum 4. pozisyonda. Uyum değeri 'is2'.
Uyum 8. pozisyonda. Uyum değeri 'Is3'.
Uyum 12. pozisyonda Uyum değeri 'is4'.
```

Ekran çıktısı



Match nesnesi bir arama işlemi sonuçları ile iliştirilen üç yanlızca okunabilir özelliklere erişmek için kullanılır ve bir düzenli ifade kullanan uyum işlemi.

Basit bir şekilde,bir düzenli ifade sizin bütünüyle karşılaştırma yapabileceğiniz bir string desenidir veya başka bir stringin bir kısmıdır. Bunun yanında, söyleyebilirimki düzenli ifadeler çok karmaşık olabilir.

RegExp nesnesi başka bir string içindeki uyum string desenini aramak için kullanılabilir. Bir Match nesnesi RegExp nesnesi bir uyum bulduğu her bir zamanda oluşturulur. Sıfır veya daha çok uyum yapılabileceği için, RegExp nesnesi gerçekte bir Matches koleksiyonu olarak işaret edilen Match nesnesinin bir koleksiyonunu döndürür.

Match Nesnesi Özellikleri

FirstIndex Özelliği

Bu özellik sıfır olarak numaralandırılan ilk pozisyon ile soldan sayılan pozisyonu döndürür.

```
Match.FirstIndex
```

Length Özelliği

Bu özellik bir arama stringi içinde bulunan uyumlu text uzunluğunu döndürür.

Match.Length

Value Özelliği

Bu özellik arama esnasında uyumlu olan o anki text döndürür.

Match.Value

Matches Collection Nesnesi

Düzenli ifade Match nesnelerinin bir koleksiyonudur. Bir Matches koleksiyonu bireysel Match nesneleri içerir, ve yanlızca RegExp nesnesinin Execute metodu kullanılarak oluşturulabilir. Matches koleksiyonunun bir özelliği yanlızca okunabilir, bu bireysel Match nesne özelliklerinin de özelliğidir.

Matches koleksiyonu bir arama sonuçlarını içeren nesnelerin bir koleksiyonudur ve uyum işlemi olarak bir düzenli ifade kullanır.

Basit bir şekilde,bir düzenli ifade sizin bütünüyle karşılaştırma yapabileceğiniz bir string desenidir veya başka bir stringin bir kısmıdır. Bunun yanında, söyleyebilirimki düzenli ifadeler çok karmaşık olabilir.

RegExp nesnesi başka bir string içindeki uyum string desenini aramak için kullanılabilir. Bir Match nesnesi RegExp nesnesi bir uyum bulduğu her bir zamanda oluşturulur. Sıfır veya daha çok uyum yapılabileceği için, RegExp nesnesi gerçekte bir Matches koleksiyonu olarak işaret edilen Match nesnesinin bir koleksiyonunu döndürür.

Aşağıdaki kod örnek Microsoft tarafında yayınlanmış çalışan bir programın basitleştirilmiş halidir.

```
<%

' Bu sub prosedür uyumlar bulur

Sub RegExpTest(strMatchPattern, strPhrase)

   ' değişkenler oluştur

   Dim objRegEx, Match, Matches, StrReturnStr

   'RegExp nesnesinin tutamacını oluştur

   Set objRegEx = New RegExp

   ' tüm uyumları bul
```

```
    objRegEx.Global = True
    ' büyük küçük harf duyarlı
    objRegEx.IgnoreCase = True
    ' pattern değerini ayarla
    objRegEx.Pattern = strMatchPattern
    ' uyumlar koleksiyonunu oluştur
    Set Matches = objRegEx.Execute(strPhrase)
    ' tüm uyumları yazdır
    For Each Match in Matches
       strReturnStr = "Uyum "
       strReturnStr = strReturnStr & Match.FirstIndex & ". Uyum değeri '"
       strReturnStr = strReturnStr & Match.value & "'." & "<BR>" & VBCrLf
       ' çıktıyı yaz
       Response.Write(strReturnStr)
  : Next
End Sub
' alt rutini çağır
RegExpTest "is.", "Is1 is2 Is3 is4"
%>
```

Çıkış:

```
Uyum pozisyon 0. Uyum değeri 'Is1'.
Uyum pozisyon 4. Uyum değeri 'is2'.
Uyum pozisyon 8. Uyum değeri 'Is3'.
Uyum pozisyon 12.Uyum değeri 'is4'.
```

Matches Collection Nesnesi Özellikleri

Count Özelliği

Matches koleksiyonunda var olan Match nesnelerinin kaç tane olduğunu bize bir tamsayı olarak döndürür.

```
nesne.Count
```

Item Özelliği

Item özelliği belirli anahtar argümanı ile gösterilen koleksiyonda bir maddenin değerini getirmemize izin verir ve maddedegeri kullanılarak da değeri ayarlanabilir.

```
nesne.Item(anahtar) [ = maddedegeri]
```

RexExp Nesnesi

Basit düzenli ifade desteği sağlar. Düzenli bir ifade (Regular Expression) uyumunun yanlızca okunabilir bir özelliğine erişim sağlar. Bir Match nesnesi Match nesnelerinin bir koleksiyonunu güncel olarak döndüren RegExp nesnesinin Execute metodu kullanılarak yanlızca oluşturulabilir. Tüm Match nesnesi özellikleri yanlızca okunabilirdir.

Bir düzenli ifade yürütüldüğü zaman, sıfır veya daha çok Match nesnesi sonuçlanabilir. Herbir Match nesnesi düzenli ifade tarafından bulunan stringe erişimi sağlar, string uzunluğu, ve uyumun nerede bulunduğunu gösteren bir indis değeri.

Aşağıdaki kod Match nesnesinin kullanımını gösteriyor:

```
Function RegExpTest(desen, str1)

Dim regEx, Match, Matches

' Değişken oluşturma.

Set regEx = New RegExp

' Düzenli İfade oluşturma.

regEx.Pattern = desen

' Desen ayarlama.

regEx.IgnoreCase = True
```

```
' Büyük-küçük harf duyarlılığı yok.
regEx.Global = True
' Glogal erişime izin ver.
Set Matches = regEx.Execute(str1)
' Aramayı yürüt.
For Each Match in Matches
  RetStr = RetStr & "Uyum " & I & " pozisyonunda bulundu"
  RetStr = RetStr & Match.FirstIndex & ". Uyum değeri'"
  RetStr = RetStr & Match.Value & "'." & vbCRLF
Next
RegExpTest = RetStr
End Function
MsgBox(RegExpTest("is.", "IS1 is2 IS3 is4"))
```

RexExp nesnesi bir hedef string içinde bir arama string deseninin tüm oluşumlarına uymak ve aramak için kullanılır.

RegExp nesnesi arama esnasında bir uyum bulduğu zaman, bir Match nesnesi oluşturulur ve bir Matches koleksiyonuna eklenir.

Arama deseni Pattern özelliği kullanılarak bildirilir. O ,örneğin, basit bir string olabilir, "fiyat analizi" gibi. Bunun yanında, arama deseni bir düzenli ifade de olabilir. Düzenli ifadeler çok basit olandan çok karmaşık olan bir bölgede olabilir.Pattern özelliği sayfası düzenli ifadeler ile kullanılabilen özel karakterlerin bir tanıtıcı listesini taşır.

RegExp Nesnesi Özellikleri

Global Özelliği

Bu özellik yanlızca RegExp nesne değişkeni ile kullanılır ve bir boolean değer döndürür. False bir aramanın yanlızca ilk uyum oluşumunu bulması anlamına gelir. True bir arama bir uyumun tüm oluşumlarını bulması anlamına gelir.

```
nesne.Global = [ True | False ]
```

IgnoreCase Özelliği

Bu özellik yanlızca RexExp nesne değişkeni ile kullanılır ve bir boolean değer döndürür. False bir arama küçük-büyük harf ayırımı dikkate almalı anlamına gelir. True bir aramanın bir uyumda harfin büyük veya küçük olmasını dikkate almaması gerektiği anlamına gelir.

```
nesne.IgnoreCase = [ True | False ]
```

Pattern Özelliği

Bu özellik bir düzenli ifadeyi veya arama esnasında uyum sağlayan arama desen stringini tanımlar.

```
nesne.Property = [ AramaDesenStringi ]
```

RegExp Nesnesi Metodları

Execute Metodu

Bu metod aramayı yürütmek için ve arama deseni stringinin (veya düzenli ifade) uyumlarını aramak için kullanılır. Her bir uyum olduğunda, bir Match nesnesi oluşturulur ve bir Matches koleksiyon olarak adlandırılan bir koleksiyona eklenir.

```
nesne.Execute
```

Replace Metodu

Bu metod düzenli bir ifade aramasında bulunan texti yerdeğiştirmek için kullanılır. Bu metodu Replace fonksiyonu ile karıştırmayın.

```
nesne.Replace (String1, String2)
```

Test Metodu

Bu metod belirli bir string içinde arama deseninin olup olmadığını belirlemek için kullanılır ve aramanın sonuçlarını göstermek için bir boolean değer döndürür. True eğer arama deseni bulunduysa döner. Diğer türlü False döndürülür.

```
nesne.Test
```

TextStream Nesnesi

TextStream nesnesi text olarak okunabilen bir şekil içeren herhangi bir dosyanın içeriklerine sıralı olarak erişim sağlar. Siz CreateTextFile veya FileSystemObject

nesnesinin OpenTextFile metodunu, veya başka bir alternatif olarak File nesnesinin OpenAsTextStream metodunu kullanarak TextStream nesnesine bir tutamaç oluşturabiliriz.

Aşağıdaki kod içinde, FileSystemObject üzerindeki CreateTextFile metodunu kullanarak döndürülen TextStream nesnesidir.

```
Set fs = CreateObject("Scripting.FileSystemObject")

Set a = fs.CreateTextFile("c:\deneme.txt", True)

a.WriteLine("Bu bir denemedir.")

a.Close
```

WriteLine ve Close TextStream nesnesinin iki metodudur.

Her bir TextStream nesnesinin tutamacı dosyadaki belirli bir yere işaret eden bir iç işaretçiye sahiptir. İlk akım (stream) açıldığı zaman, bu işaretçi dosyadaki ilk karakterde pozisyonlanır. Siz TextStream nesnesini kendisine ait metod ve özelliklerini kullanarak düzenleme yaparken, İşaretçi o zaman civarda hareket edecektir.

TextStream Nesnesi Özellikleri

AtEndOfLine Özelliği

Bir boolean değer döndürür. Eğer dosya işaretçisi dosyanın son satır işaretleyicisinden önce hemen pozisyonlanırsa, değer True olur, ve diğer türlü False olur.

```
nesne.AtEndOfLine
```

AtEndOfLine özelliği TextStream dosyalarının okuma modunda açıldığında kullanılır, yoksa bir hata meydana gelir.

Aşağıdaki kod AtEndOfLine özelliğinin kullanımını gösterir.

```
Dim fs

Dim a

Dim strgetir

Set fs = CreateObject("Scripting.FileSystemObject")

Set a = fs.OpenTextFile("c:\dosya.txt", ForReading, False)

Do While a.AtEndOfLine <> True
```

```
strgetir = a.Read(1)
...

Loop

a.Close
```

AtEndOfStream Özelliği

Bir boolean değer döner. Eğer dosya işaretçisi bir dosyanın sonunda pozisyonlandıysa, değer True, diğer türlü False olur.

```
nesne.AtEndOfStream
```

AatEndOfStream özelliği yanlıca TextStream dosyalarını okuma modu için açıldığında kullanılır, yoksa bir hata meydana gelir.

Aşağıdaki kod AtEndOfStream özelliğinin kullanımını gösterir.

```
Dim fs

Dim a

Dim strgetir

Set fs = CreateObject("Scripting.FileSystemObject")

Set a = fs.OpenTextFile("c:\dosya.txt", ForReading, False)

Do While a.AtEndOfStream <> True

strgetir = a.ReadLine
...

Loop

a.Close
```

Column Özelliği

O anki satır içindeki dosya işaretçisinin güncel pozisyonunu verir. 1. kolon herbir satırdaki ilk karakterdir.

```
nesne.Column
```

yeni bir satır karakter yazıldıktan sonra, fakat herhangi diğer karakter yazılmadan önce Column 1 değerine eşit olur.

Line Özelliği

Bu özellik bir text dosya içindeki o anki güncel satır numarasını döndürür.

```
nesne.Line
```

bir dosya ilk olarak açıldığında ve herhangi bir şey yazılmadan önce, Line özelliği 1 değerine eşittir.

TextStream Nesnesi Metodları

Close Metodu

O an açık olan bir TextStream dosyasını kapatır.

```
nesne.Close
```

Read Metodu

Bu metod sizin belirttiğiniz bir TextStream dosyasından birçok karakter okur ve onları bir string olarak döndürür.

```
nesne.Read(karekterler)
```

nesne gerekli olup bir TextStream nesnesidir. Karakterler dosyadan okumak istediğiniz karakter sayısını gösterir, bu parametre gereklidir.

ReadAll Metodu

Bu metod bir text dosyanın tüm içeriğini okur ve bir string olarak döndürür.

```
nesne.ReadAll
```

büyük dosyalar için, ReadAll metodu bellek kaynaklarını tüketir. Bunun için başka dosya okuma teknikleri kullanılmalı, örneğin, bir dosyayı satır satır okumak gibi.

ReadLine Metodu

Bir TextStream dosyasından tek bir satır (newline karakteri hariç) okur ve içeriğini bir string olarak döndürür.

```
nesne.ReadLine
```

Skip Metodu

Bir TextStream dosyası okunduğu zaman, bir dosya işaretçisinin belirli sayıda karakteri atlamasına neden olur. Negatif veya pozitif olabilir.

nesne.Skip(karakterler)

nesne gerekli olup bir TextStream nesnesi adıdır. Karakterler ise bir dosyayı okuma esnasında atlanılacak karakterlerin sayısıdır, bu değeri girmek zorundayız.

SkipLine Metodu

Bir sonraki satırın başlangıcına onun o anki pozisyonundan bir dosya işaretçisini taşır.

nesne.SkipLine

bir satır atlamak demek bir satırdaki tüm karakterleri okumak ve gözardı etmektir ve sonraki yeni bir satır karakterini dahil etmek.

Eğer okuma için bir dosya açılmassa o zaman bir hata meydana gelir.

Write Metodu

Bu metod açık bir TextStream dosyasına desteklenen bir string yazar.

nesne.Write(string)

nesne gerekli olup bir TextStream nesnesidir. string dosyaya yazmak istediğiniz text bilgidir.

WriteLine Metodu

Bir TextStream dosyasına bir destekli string yazar, ve onu bir yeni satır karekteri takip eder.

nesne.WriteLine([string])

nesne gerekli olup bir TextStream nesnesidir. string dosyaya yazmak istediğiniz text bilgidir. Eğer bunu yazmazsanız, dosyaya yeni bir satır karakteri yazılır.

WriteBlankLines Metodu

Bir TextStream dosyasına ardışık yeni satır karekterlerinin bir sayısını yazmak için kullanılır.

nesne.WriteBlankLines(satırlar)

nesne bir TextStream nesnesinin adı, satırlar ise dosyaya yazmak istediğiniz newline (yenisatır) karakterlerinin sayısıdır ve bu gereklidir.

Windows NT sistemleri için, WriteBlankLines yeni bir satır için <CR><LF> kullanır. UNIX sistemlerinde ise, WriteBlankLines sadece <LF> kullanır.

Aşağıdaki örnek bir dosyadan okuma işlemi yapar.

```
<SCRIPT LANGUAGE="VBScript">

Dim nesneFSO

Dim strDosya

Dim nesneTextStream

Set nesneFSO = CreateObject("Scripting.FileSystemObject")

strDosya = "d:\dosya.txt"

If nesneFSO.FileExists(strDosya) Then

        Set nesneTextStream = nesneFSO.OpenTextFile(strDosya)

        Document.Write nesneTextStream.ReadAll

        Set nesneTextSTream = Nothing

Else

        Document.Write "Dosya mevcut değil"

End If

Set nesneFSO = Nothing

</SCRIPT>
```

Bu örnek basit olarak d sürücü üzerindeki dosya.txt olarak adlandırılan dosyayı okur ve onu gösterir, burada dosya önceden oluşturulması gerekir yoksa dosyanın olmadığını belirten bir mesaj alırsınız.

Nesneler ve For Each...Next Çevrimi

Bir For Each...Next çevrimi bir For...Next çevrimine benzer. Belirli bir sayıda yapıların tekrarlanması yerine, bir For Each...Next çevrimi bir nesnelerin toplamındaki her bir maddeyi veya bir dizinin herbir elemanı için bir grup yapıyı tekrar eder. Bu özellikle eğer bir koleksiyonda kaçtane eleman olduğunu bilmiyorsanız faydalı olur.

Aşağıdaki HTML kod örneğinde, Dictionary nesnesinin içerikleri birkaç text kutusuna bilgilerin yerleştirilmesi için kullanılıyor.

```
<HTML>
<HEAD>
<TITLE>Formlar ve Elemanlar</TITLE>
</HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub degistir_OnClick
  Dim m
  Set m = CreateObject("Scripting.Dictionary")
  m.Add "0", "Marsilya"  'Birkaç anahtar ve madde ekliyoruz
  m.Add "1", "Nice"
  m.Add "2", "Cannes"
  m.Add "3", "Frankfurth Main"
  m.Add "4", "Miami"
  m.Add "5", "Paris"
  m.Add "6", "Milano"
  m.Add "7", "Ludwigshafen"
  m.Add "8", "Orlando"
  m.Add "9", "Newyork"
  m.Add "10","Monaco"
  m.Add "11","Milano"
  m.Add "12","Cenova"
  m.Add "13","San Marino"
  m.Add "14","Metz"
```

```
  m.Add "15","Reims"
  m.Add "15","New Jersey"
  For Each i in m
    Document.form1.Elements(i).Value = m.Item(i)
  Next
End Sub
-->
</SCRIPT>
<BODY>
<CENTER>
<FORM NAME="form1"
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
```

```
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Text"><p>
<Input Type ="Button" NAME="degistir" VALUE="Buraya Tıkla"><p>
</FORM>
</CENTER>
</BODY>
</HTML>
```

BÖLÜM
ONBEŞ

15

VBScript Nesne Özellikleri

- Description
- FirstIndex
- Global
- HelpContext
- HelpFile
- IgnoreCase
- Number
- Pattern
- Source
- Value

VBScript Nesne Özellikleri

Burada VBScript içinde tanımlanmış olan nesnelerin özellikleri tek tek ele alınarak incelenecektir.

VBScript nesne özellikleri

- Description
- FirstIndex
- Global
- HelpContext
- HelpFile
- IgnoreCase
- Number
- Pattern
- Source
- Value

Description

Err nesnesi içinde saklanılan bir hatanın tanımını döndürür veya ayarlar.

nesne.Description [= stringifade]

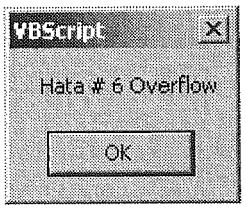
Argümanları *nesne* daima bir Err nesnesidir, *stringifade* ise hatanın bir tanımını içeren bir string ifade.

Description özelliği hatanın kısa bir tanımını içerir. Bu özelliği kotarmak istemediğiniz veya kotaramadığınız bir hatayı kullanıcıya göstermek için kullanın. Bir kullanıcı tanımlı bir hata üretiminde, bu özelliğe sizin hatanızın kısa bir tanımını atarsınız. Eğer Description içi dolu değilse, ve Number değeri bir VBScript çalışma zamanı hatasına benziyorsa, hata ile ilişkilendirilen tanıtıcı string döndürülür.

```
<SCRIPT LANGUAGE="VBScript">

On Error Resume Next

Err.Raise 6

' Bir taşma hatası meydana gelir.

MsgBox ("Hata # " & CStr(Err.Number) & " " & Err.Description)
```

```
'Hata temizleniyor.

Err.Clear

</SCRIPT>
```

Örnek program kodunun tarayıcı üzerindeki görünümü



FirstIndex

Düzenli bir ifade aramasından döndürülen bir uyum nesnesinin ilk karakterinin karakter ofsetini döndürür. Bu arama stringi hedef string içinde bulunmuş demektir, ve o FirstIndex özelliği tarafından işaret edilen indeks pozisyonunda görünür. Çoğu diğer VB koleksiyonları ve Instr fonksiyonu gibi değil, aranılan string içinde ilk karekterin karakter ofseti 0 dır, 1 değildir.

Bir arama stringi içinde bir uyumun meydana geldiği yerdeki pozisyonu getirir.

```
nesne.FirstIndex
```

nesne daima bir Match nesnesidir.

FirstIndex özelliği arama stringinin başlangıcından bir sıfır-tabanlı ofset kullanır. Diğer bir deyişle, string içindeki ilk karakter karakter sıfır (0) olarak tanımlıdır.

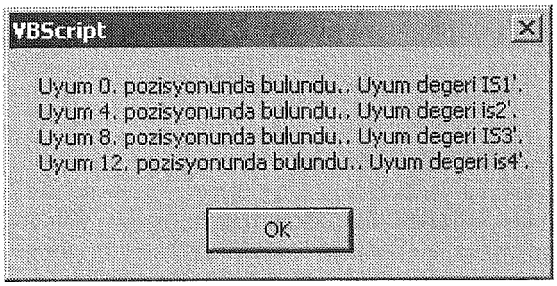
Aşağıdaki örnek FirstIndex özelliğinin kullanımını gösterir

```
<HTML>

<HEAD>

<TITLE>Düzenli İfadeler</TITLE>

<META http-equiv="Content-Type" content="text/html; charset=">

<SCRIPT LANGUAGE="VBScript">
```

```
Function RegExpTest(desen, str1)
Dim regEx, Match, Matches
' Degisken olusturma.
Set regEx = New RegExp
' Düzenli ifade oluşturma.
regEx.Pattern = desen
' Desen ayarlama.
regEx.IgnoreCase = True
' Büyük-küçük harf duyarlılığı yok.
regEx.Global = True
' Glogal erişime izin ver.
Set Matches = regEx.Execute(str1)
' Aramayi yürüt.
For Each Match in Matches
  RetStr = RetStr & "Uyum " & Match.FirstIndex & "." & " pozisyonunda
bulundu."
  RetStr = RetStr & ". Uyum degeri '"
  RetStr = RetStr & Match.Value & "'." & vbCRLF
Next
RegExpTest = RetStr
End Function
</SCRIPT>
</HEAD>
<BODY bgcolor="#FFFFFF" text="#000000">
<SCRIPT>
MsgBox(RegExpTest("is.", "IS1 is2 IS3 is4"))
```

```
</SCRIPT>

</BODY>

</HTML>
```

Örnek program kodunun tarayıcı üzerindeki görünümü



Global

Düzenli bir ifade aramasının tüm oluşumlar veya yanlızca ilk oluşumla uyumlu olması gerekip gerekmediğini tanımlar.

Bir desen bir string içindeki tüm oluşumlar veya sadece ilk olanı uyumlu olması gerektiğini gösterir bir Boolean değer döndürür veya ayarlar.

```
nesne.Global [= True | False ]
```

nesne argümanı daima RegExp nesnesidir. Eğer arama tüm stringde olursa, global özelliğin değeri True dur. Eğer değilse varsayılan olarak False dur.

FirstIndex özelliği için vermiş olduğumuz örnek VBScript kod içinde Global özelliği aşağıda örnek olarak verilmiştir.

```
. . .

' tüm uyumları bul, true değeri aramanın global anlamda olacağını gösterir

  objRegEx.Global = True

' büyük-küçük harf duyarlılığını kaldır

  objRegEx.IgnoreCase = True
```

```
' deseni ( pattern ) ayarla

  objRegEx.Pattern = strMatchPattern
...
```

HelpContext

Bir yardım dosyasındaki bir konu başlığının ID değerini gösteren bir HelpContextID değerini ayarlar veya döndürür.

```
nesne.HelpContext [= baglamID]
```

Argüman olarak *nesne* gerekli ve daima bir Err nesnesi olmak zorunda, ve seçimlilik olan *baglamID* ise yardım dosyası içindeki bir yardım konusu için geçerli bir tanıtıcıdır. Eğer bir yardım dosyası HelpFile içinde belirtilmiş ise, HelpContext özelliği tanımlı yardım konusunu otomatik olarak göstermesi için kullanılır. Eğer her iki HelpFile ve HelpContext boş ise, Number özelliğinin değeri sınanır. Eğer o VBScript çalışma zamanı hata değerine benziyorsa, o zaman hata için VBScript yardım bağlam ID si kullanılır. Eğer Number özelliği bir VBScript hatasına benzemiyorsa, VBScript yardım dosyası için içerik ekranı gösterilir.

Aşağıdaki örnek HelpContext özelliğinin kullanımını gösterir.

```
<%
...

On Error Resume Next

Dim Msg

Err.Clear

Err.Raise 6   ' Bir taşma hatası oldu.

Err.Helpfile = "yardim.hlp"

Err.HelpContext = baglamID

If Err.Number <> 0 Then

  Msg = "F1 tuşuna bas veya Yardımı görmek için " &

      Err.Helpfile & " konusu için" &   " izleyen

      HelpContext: " & Err.HelpContext

  MsgBox Msg, ,"hata:" & Err.Description, Err.Helpfile, Err.HelpContext
```

```
End If
...
%>
```

HelpFile

Bir nesne ile ilişkilendirilmiş bir yardım dosyasının adını ayarlar veya döndürür.

```
nesne.HelpFile [= baglamID]
```

Argümanlar *nesne* gerekli olup daima bir Err nesnesidir, ve *baglamID* tamamen yardım dosyasının yolunu belirtir.

Eğer yardım dosyası HelpFile içinde belirtilmiş ise, kullanıcı hata mesaj diyalog kutusundaki düğmeye (veya F1 tuşuna basar ise) tıklarsa otomatik olarak çağrılır. Eğer HelpContext özelliği belirtilen dosya için geçerli bir bağlam ID si taşıyorsa, o konu otomatik olarak gösterilir. Eğer HelpFile belirtilmemiş ise, VBScript yardım dosyası gösterilir.

Aşağıdaki örnek HelpFile özelliğinin kullanımını gösterir.

```
<%
...

On Error Resume Next

Dim Msg

Err.Clear

Err.Raise 6   ' Bir taşma hatası oldu.

Err.HelpFile = "yardim.hlp"

Err.HelpContext = baglamID

If Err.Number <> 0 Then

  Msg = "F1 tuşuna bas veya Yardımı görmek için " &

      Err.Helpfile & " konusu için" &   " izleyen

      HelpContext: " & Err.HelpContext

  MsgBox Msg, ,"hata:" & Err.Description, Err.Helpfile, Err.HelpContext

End If
...
%>
```

IgnoreCase

Bir düzenli ifade aramasında büyük-küçük harf duyarlılığının olup olmayacağını belirler.

```
nesne.IgnoreCase [= True | False ]
```

Argüman olarak *nesne* daima bir RegExp nesnesidir. Eğer arama yaparken büyük – küçük harf ayırımı yapılacaksa, IgnoreCase özelliğinin değeri False, True ise tam tersidir. Varsayılan ise False değeridir.

Aşağıdaki örnek IgnoreCase özelliğinin kullanımını gösterir.

```
<%
...

Function RegExpTest(desen, str1)

Dim regEx, Match, Matches
...

regEx.IgnoreCase = True

' büyük-küçük harf duyarlılığı yok.

' eğer False atanırsa büyük-küçük harf duyarlılığı etkin olur.
...

End Function
...

%>
```

Number

Err nesnesinin varsayılan özelliği Number dır. Err.Number bir tamsayı içerir. Bir çalışma-zamanı hatası meydana geldiğinde, Err nesnesinin özellikleri hataları kotarmak için kullanılabilen bilgi ve hataların tek tanımlayıcılarıyla ilgili bilgiler ile doldurulur.

```
nesne.Number
```

Aşağıdaki örnek Number özelliğinin kullanımını gösterir.

```
<%
...

On Error Resume Next
```

```
Err.Raise 6

'Bir taşma hatası meydana geliyor.

MsgBox("Hata# " & CStr(Err.Number) & " " & Err.Description)

Err.Clear

' Hata temizleniyor.
...

%>
```

Pattern

Düzenli bir ifade araması için desen stringin ayarlanmasını veya döndürülmesini sağlamak için kullanılır.

```
nesne.Pattern [= "aramastringi"]
```

Argüman olarak *nesne* gerekli olup daima bir RegExp nesnesi değişkenidir, aramastringi seçimlilik olup aranılacak olan düzenli string ifadesi . ayarlamalar kısmında tablo içinde tanımlı düzenli ifade karakterleri dahil olabilir.

Pattern özelliği bir string arama deseni (bir düzenli ifade olarak ta bilinir) tanımlamak için bir RegExp nesne değişkeni ile kullanılır.

Aşağıdaki örnekler Pattern özelliğinin kullanımını gösterir.

```
<%

OrnekRegExpNesne.Pattern = "Kontrol ve Bilgisayar Mühendisliği"

%>
```

Source

Bir hatanın meydana geldiği yerdeki kaynağı döndürür. Sunucu tarafındaki VBScript içinde, kaynak daima hatanın meydana geldiği sayfadaki sayfa adını içerir.

Başarılı bir düzenli ifade aramasından sonuçlanan bir Match nesnesinin text bilgisini döndürür.

```
nesne.Source [= stringifadesi]
```

Argüman olarak *nesne* daima bir Err nesnesidir, ve stringifadesi hatayı üreten uygulamayı gösteren bir string ifadesidir.

```
<%
On Error Resume Next
Err.Raise 8    ' kullanıcı tanımlı bir hata oluşturma
Err.Description = "örnek hata"
Err.Source = "UygulamaAdi"
Response.Write "Hata tipi " & Err.Description & "' hataya neden olan_
                program '" & Err.Source & "'."
%>
```

Çıkış:

"Hata tipi ' örnek hata' hataya neden olan program 'UygulamaAdi'."

Value

Bir arama stringi içinde bulunan bir uyumun text bilgisi veya değerini döndürür.

nesne.Value

nesne argümanı daima Match nesnesidir.

Aşağıdaki örnek program parçası Value özelliğinin kullanımını gösterir.

```
. . .
For Each Match in Matches
  RetStr = RetStr & "Uyum " & Match.FirstIndex & "." & " pozisyonunda
bulundu."
  RetStr = RetStr & ". Uyum degeri "'
  RetStr = RetStr & Match.Value & "'." & vbCRLF
Next
. . .
```

VBScript Anahtar Sözcükleri

Visual Basic Script dili alışılmış programlama görevlerini yapmak için iç fonksiyonlar ve metodlar sağlar. Visual Basic Script 5 inci versiyona kendi sınıf ve nesnelerinizi oluşturmak için destek ve güçlü tarama yetenekleri de eklenmiştir.

Script yazımı anahtar sözcüklerin, iç fonksiyonların, kullanıcılar için olan rutinlere çağrı yapma , ve nesne metodlarının bir kombinasyonundan oluşur. Anahtar sözcükler dilin bir parçası olup VBScript tarafından bunlar bilinirler. Siz bir anahtar sözcük ile aynı ada sahip bir değişken tanımı yapamazsınız, bundan dolayı onları ön plana çıkarmak iyi olur, ve siz çıkabilecek problemlerden bu şekilde kaçınabilirsiniz.

Aşağıdaki tabloda alfabetik sıraya göre VBScript anahtar sözcükleri ve tip tanımları verilmiştir.

Anahtar sözcük	Tipi
Abs	Fonksiyon
Toplama (+)	Operatör
And	Operatör
Array	Fonksiyon
Asc	Fonksiyon
Atama (=)	Operatör
Atn	Fonksiyon
Call	İfade
Cbool	Fonksiyon
Cbyte	Fonksiyon
Ccur	Fonksiyon
Cdate	Fonksiyon
CDbl	Fonksiyon
Chr	Fonksiyon
Cint	Fonksiyon
Clear	Fonksiyon
Clng	Fonksiyon
Class	Nesne
Class	İfade
String birleştirme (&)	Operatör
Const	İfade
Cos	Fonksiyon
CreateObject	Fonksiyon
CSng	Fonksiyon

CStr	Fonksiyon
Date	Fonksiyon
DateAdd	Fonksiyon
DateDiff	Fonksiyon
DatePart	Fonksiyon
DateSerial	Fonksiyon
DateValue	Fonksiyon
Day	Fonksiyon
Description	Özellik
Dictionary	Nesne
Dim	İfade
Bölme (/)	Operatör
Do...Loop	İfade
Empty	Değer
Eqv	Operatör
Erase	İfade
Err	Nesne
Eval	Fonksiyon
Execute	Metod
Execute	İfade
ExecuteGlobal	İfade
Exit	İfade
Exp	Fonksiyon
Üs (^)	Operatör
False	Değer
FileSystemObject	Nesne
Filter	Fonksiyon
FirstIndex	Özellik
Fix	Fonksiyon
For...Next	İfade
For Each...Next	İfade
FormatCurrency	Fonksiyon
FormatDateTime	Fonksiyon
FormatNumber	Fonksiyon
FormatPercent	Fonksiyon
Function	Fonksiyon
GetLocale	Fonksiyon
GetObject	Fonksiyon
GetRef	Fonksiyon
Global	Özellik
Hex	Fonksiyon
HelpContext	Özellik
HelpFile	Özellik
Hour	Fonksiyon
If...Then...Else	İfade

IgnoreCase	Özellik
Imp	Operatör
Initialize	Olay
InputBox	Fonksiyon
InStr	Fonksiyon
InStrRev	Fonksiyon
Int	Fonksiyon
Tamsayı Bölme (\ )	Operatör
Is	Operatör
IsArray	Fonksiyon
IsDate	Fonksiyon
IsEmpty	Fonksiyon
IsNull	Fonksiyon
IsNumeric	Fonksiyon
IsObject	Fonksiyon
Join	Fonksiyon
Lbound	Fonksiyon
Lcase	Fonksiyon
Left	Fonksiyon
Len	Fonksiyon
LoadPicture	Fonksiyon
Log	Fonksiyon
Ltrim	Fonksiyon
Match	Nesne
Matches	Koleksiyon
Mid	Fonksiyon
Minute	Fonksiyon
Mod	Operatör
Month	Fonksiyon
MonthName	Fonksiyon
MsgBox	Fonksiyon
Çarpma (*)	Operatör
Eksileme ve Çıkarma (-)	Operatör
Not	Operatör
Now	Fonksiyon
Nothing	Değer
Null	Değer
Number	Özellik
Oct	Fonksiyon
On Error	İfade
Option Explicit	İfade
Or	Operatör
Pattern	Özellik
Private	Özellik
PropertyGet	Özellik
PropertyLet	Özellik

PropertySet	Özellik
Public	Özellik
Raise	Metod
Randomize	İfade
ReDim	İfade
RexExp	Nesne
Rem	İfade
Replace	Fonksiyon
Replace	Metod
RGB	Fonksiyon
Right	Fonksiyon
Rnd	Fonksiyon
Round	Fonksiyon
Rtrim	Fonksiyon
ScriptEngine	Fonksiyon
ScriptEngineBuildVersion	Fonksiyon
ScriptEngineMajorVersion	Fonksiyon
ScriptEngineMinorVersion	Fonksiyon
Second	Fonksiyon
Select Case	İfade
Set	İfade
SetLocale	Fonksiyon
Sgn	Fonksiyon
Sin	Fonksiyon
Source	Özellik
Space	Fonksiyon
Split	Fonksiyon
Sqr	Fonksiyon
StrComp	Fonksiyon
String	Fonksiyon
StrReverse	Fonksiyon
Sub	İfade
Çıkarma (-) ve Eksileme	Operatör
Tan	Fonksiyon
Terminate	Olay
Test	Metod
Time	Fonksiyon
Timer	Fonksiyon
TimeSerial	Fonksiyon
TimeValue	Fonksiyon
Trim	Fonksiyon
True	Değer
TypeName	Fonksiyon
Ubound	Fonksiyon
Ucase	Fonksiyon

Value	Özellik
VarType	Fonksiyon
WeekDay	Fonksiyon
WeekDayName	Fonksiyon
While...Wend	İfade
With...End With	İfade
Xor	Operatör
Year	Fonksiyon

Value	Özellik
VarType	Fonksiyon
WeekDay	Fonksiyon
WeekDayName	Fonksiyon
While...Wend	İfade
With...End With	İfade
Xor	Operatör
Year	Fonksiyon

VBScript Versiyon Bilgileri

Tabloda sunucu uygulamaları tarafından yürürlülüğe konan Microsoft Visual Basic Scripting yayınının versiyon listesi verilmiştir. Bu tabloya bakarak kullanacağınız VBScript komutun kullanacağınız tarayıcı üzerinde çalışıp çalışmayacağına karar verebilirsiniz. Bu şekilde versiyon farkından dolayı karşılaşacağınız problemleri önceden tespit etmiş ve önlem almış olursunuz.

Sunucu Uygulamaları	1.0	2.0	3.0	4.0	5.0	5.5	5.6
Microsoft Internet Explorer 3.0	x						
Microsoft Internet Information Server 3.0		x					
Microsoft Internet Explorer 4.0			x				
Microsoft Internet Information Server 4.0			x				
Microsoft Windows Scripting Host 1.0			x				
Microsoft Outlook 98			x				
Microsoft Visual Studio 6.0				x			
Microsoft Internet Explorer 5.0					x		
Microsoft Internet Information Services 5.0					x		
Microsoft Internet Explorer 5.5						x	
Microsoft Visual Studio .NET							x

Tabloda VBScript dilinin özellikleri ve ilk çıkış zamanı versiyon bilgisi verilmiştir.

Dil Elemanları	1.0	2.0	3.0	4.0	5.0	5.5	5.6
Abs Fonksiyonu	x						
Toplama Operatörü (+)	x						
And Operatörü	x						
Array Fonksiyonu		x					
Asc Fonksiyonu	x						
Atama Operatörü (=)	x						
Atn Fonksiyonu	x						
Call Yapısı	x						
CBool Fonksiyonu	x						
CByte Fonksiyonu	x						
CCur Fonksiyonu	x						
CDate Fonksiyonu	x						
CDbl Fonksiyonu	x						
Chr Fonksiyonu	x						
CInt Fonksiyonu	x						
Class Nesnesi					x		
Class İfadesi					x		

Clear Metodu	x						
CLng Fonksiyonu	x						
Color Sabitleri		x					
Karşılaştırma Sabitleri		x					
Birleştirme Operatörü (&)	x						
Const Yapısı		x					
Cos Fonksiyonu	x						
CreateObject Fonksiyonu		x					
CSng Fonksiyonu	x						
CStr Fonksiyonu	x						
Date ve Time Sabitleri		x					
Date Format Sabitleri		x					
Date Fonksiyonu	x						
DateAdd Fonksiyonu		x					
DateDiff Fonksiyonu		x					
DatePart Fonksiyonu		x					
DateSerial Fonksiyonu	x						
DateValue Fonksiyonu	x						
Day Fonksiyonu	x						
Description Özelliği	x						
Dim İfadesi	x						
Bölme Operatörü (/)	x						
Do...Loop İfadesi	x						
Empty	x						
Eqv Operatörü	x						
Erase İfadesi	x						
Err Nesnesi	x						
Eval Fonksiyonu					x		
Execute Metodu					x		
Execute İfadesi					x		
ExecuteGlobal İfadesi					x		
Exit İfadesi	x						
Exp Fonksiyonu	x						
Üs Operatörü (^)	x						
False	x						
Filter Fonksiyonu		x					
FirstIndex Özelliği					x		
Fix Fonksiyonu	x						
For...Next İfadesi	x						
For Each...Next İfadesi		x					
FormatCurrency Fonksiyonu		x					
FormatDateTime Fonksiyonu		x					
FormatNumber Fonksiyonu		x					
FormatPercent Fonksiyonu		x					
Function İfadesi	x						

GetLocale Fonksiyonu					x		
GetObject Fonksiyonu		x					
GetRef Fonksiyonu					x		
Global Özelliği					x		
Hex Fonksiyonu	x						
HelpContext Özelliği		x					
HelpFile Özelliği		x					
Hour Fonksiyonu	x						
If...Then...Else İfadesi	x						
IgnoreCase Özelliği					x		
Imp Operatörü	x						
Initialize Olayı					x		
InputBox Fonksiyonu	x						
InStr Fonksiyonu	x						
InStrRev Fonksiyonu		x					
Int Fonksiyonu	x						
Tamsayı Bölme Operatörü (\)	x						
Is Operatörü	x						
IsArray Fonksiyonu	x						
IsDate Fonksiyonu	x						
IsEmpty Fonksiyonu	x						
IsNull Fonksiyonu	x						
IsNumeric Fonksiyonu	x						
IsObject Fonksiyonu	x						
Join Fonksiyonu		x					
LBound Fonksiyonu	x						
LCase Fonksiyonu	x						
Left Fonksiyonu	x						
Len Fonksiyonu	x						
Length Özelliği					x		
LoadPicture Fonksiyonu		x					
Log Fonksiyonu	x						
LTrim Fonksiyonu	x						
Match Nesnesi					x		
Matches Koleksiyonu					x		
Mid Fonksiyonu	x						
Minute Fonksiyonu	x						
Miscellaneous Sabitleri		x					
Mod Operatörü	x						
Month Fonksiyonu	x						
MonthName Fonksiyonu		x					
MsgBox Sabitleri		x					
MsgBox Fonksiyonu	x						
Çarpma Operatörü (*)	x						
Eksileme Operatörü (-)	x						

Not Operatörü	X						
Now Fonksiyonu	X						
Nothing	X						
Null	X						
Number Özelliği	X						
Oct Fonksiyonu	X						
On Error İfadesi	X						
Option Explicit İfadesi	X						
Or Operatörü	X						
Pattern Özelliği					X		
Private İfadesi		X					
PropertyGet İfadesi					X		
PropertyLet İfadesi					X		
PropertySet İfadesi					X		
Public İfadesi		X					
Raise Metodu	X						
Randomize İfadesi	X						
ReDim İfadesi	X						
RegExp Nenesi					X		
Rem İfadesi	X						
Replace Fonksiyonu		X					
Replace Metodu					X		
RGB Fonksiyonu		X					
Right Fonksiyonu	X						
Rnd Fonksiyonu	X						
Round Fonksiyonu		X					
RTrim Fonksiyonu	X						
ScriptEngine Fonksiyonu		X					
ScriptEngineBuildVersion Fonksiyonu		X					
ScriptEngineMajorVersion Fonksiyonu		X					
ScriptEngineMinorVersion Fonksiyonu		X					
Second Fonksiyonu	X						
Select Case İfadesi	X						
Set İfadesi	X						
SetLocale Fonksiyonu					X		
Sgn Fonksiyonu	X						
Sin Fonksiyonu	X						
Source Özelliği	X						
Space Fonksiyonu	X						
Split Fonksiyonu		X					
Sqr Fonksiyonu	X						
StrComp Fonksiyonu	X						
String Sabitleri		X					
String Fonksiyonu	X						
StrReverse Fonksiyonu		X					

Sub İfadesi	x						
Çıkarma Operatörü (-)	x						
Tan Fonksiyonu	x						
Terminate Olayı					x		
Test Metodu					x		
Time Fonksiyonu	x						
Timer Fonksiyonu					x		
TimeSerial Fonksiyonu	x						
TimeValue Fonksiyonu	x						
Trim Fonksiyonu	x						
Tristate Sabitleri		x					
True	x						
TypeName Fonksiyonu		x					
UBound Fonksiyonu	x						
UCase Fonksiyonu	x						
Value Özelliği					x		
VarType Sabitleri		x					
VarType Fonksiyonu	x						
VBScript Sabitleri		x					
Weekday Fonksiyonu	x						
WeekdayName Fonksiyonu		x					
While...Wend İfadesi	x						
With İfadesi					x		
Xor Operatörü	x						
Year Fonksiyonu	x						