Guida a VisualBasic Script

(a cura di www.html.it)

Primi passi

1. Introduzione
I linguaggi di scripting lato client e loro utilizzo all'interno delle pagine HTML
2. Aggiunta di codice VBScript in una pagina
Come aggiungere il codice VBscript in una pagina HTML e quale editor utilizzare per gestire codice VBscript

Strumenti di base

3. Variabili
Come utilizzare le variabili all'interno di VBscript. Elenco dei sottotipi delle variabili
4. Operatori
L'elenco di tutti gli operatori previsti da VBscript
5. Strutture condizionali
Le principali strutture condizionali di VBscript: IF ... THEN ... ELSE e la struttura SELECT
6. Strutture per cicli
I Cicli in VBscript: FOR - NEXT, FOR - EACH, DO LOOP.

Le Funzioni

7. Funzioni Data e Ora
Le Funzioni di VBscript per la gestione della data e dell'ora: sintassi ed utilizzo
8. Funzioni sulle stringhe
Le Funzioni di VBscript per la gestione delle stringhe: sintassi ed utilizzo
9. Funzioni Matematiche
Le Funzioni matematiche di VBscript per la gestione delle operazioni complesse: sintassi ed utilizzo
10. Funzioni di Conversione
Le Funzioni di Conversione di VBscript per la gestione delle operazioni complesse: sintassi ed utilizzo
11. Creare funzioni e procedure
Funzioni e procedure: i metodi per semplificare la scrittura del codice VBscript

VBScript e gli oggetti

12. Gli oggetti
Programmazione ad oggetti ed eventi in VBscript: gli oggetti estensione di Javascript
13. Gli oggetti di VBScript
Programmazione ad oggetti ed eventi in VBscript: gli oggetti propri di VBScript
14. L'Oggetto Drive
L'oggetto drive: recuperare informazioni sugli hard disk del computer
15. L'Oggetto Cartella
Come agire direttamente sulle directory: l'oggetto per cancellare, spostare e copiare una cartella.
16. L'Oggetto File
L'oggetto File consente di operare direttamente sui file di un sistema operativo Windows
17. L'Oggetto TextStream
Grazie a questo oggetto è possibile creare, leggere e scrivere in un file di testo
18. Utilizzo di VBScript con gli oggetti
Come gestire i controlli ActiveX e gli oggetti Java con VBScript

Tutorial

19. VBscript: Introduzione
Introduzione: scopriamo passo passo come creare una pagina HTML con dentro del codice VBScript
20. Formattare la data
L'aspetto della data è un elemento client. Vedremo in questo capitolo come imporre un formato indipendente
21. Confronti fra date
Eseguire un confronto fra date diverse: trucchi e precauzioni
22. Un piccolo calendario
Costruiamo un piccolo calendario da pubblicare sul Web
23. Il gioco dell'impiccato
Seguendo il detto "giocando s'impara", vediamo come implementare questo semplice gioco
24. Il FileSystem Object
Se VBScript è utilizzato per realizzare virus informatici, ci sarà un perchè. Ecco alcuni esempi di cosa può fare VBScript sul tuo computer
25. Sfogliando l'HardDisk
Lo scopo di questa applicazione è quello di visualizzare il contenuto dell'HardDisk, come fa esplora risorse
26. Gestione dei file
Lavorando con i file, spesso capita di doverli spostare, copiare, cancellare
27. Gestione delle cartelle
Creazione, copia, e cancellazione delle cartelle nel file system
28. Oggetto TextStream
Impariamo ora a leggere e scrivere da un file di testo

Strumenti di interattività

29. Comunicare con l'utente
Come rendere più interattive le nostre pagine con VBScript
30. Comunicare con l'utente per mezzo di finestre
Usare delle alert box di windows per comunicare con l'utente
31. Le MsgBox
Utilizzare le finestre di messaggio di Windows
32. Impedire la scrittura in una textbox
Come impedire la scrittura in un'area di un form o modulo

Primi passi

1. Introduzione

Per l'utente che ha imparato a creare pagine in HTML, ad un certo punto nasce l'esigenza di pretendere di più dal proprio lavoro. Occorre allora imparare un linguaggio di **scripting**. Vedremo che esistono

- **script lato client**, che sono eseguiti direttamente dal browser senza il bisogno di ricaricare la pagina,
- **script lato server**, che sono eseguiti su server e che ad ogni richiesta generano una nuova pagina da inviare al browser.

Con VbScript si possono realizzare sia script lato server (con ASP) sia script lato client (in genere supportati solo da IE).
I linguaggi di scripting si inseriscono dentro il codice HTML per rendere una pagina dinamica e interattiva con l'utente; ad esempio, uno di questi linguaggi è ASP che in particolare è un linguaggio per script lato server.
Con ASP in realtà si può usare anche Javascript, ma è uso comune programmare le pagine ASP con VbScript.
Per comodità intediamo per VbScript il codice lato client e per ASP il codice lato server.
Il seguente codice ASP è uno script lato server che scrive "Linguaggio di Scripting" in una tabella:

```
<table border="1">
<tr>
<td>
<% Response.Write "Linguaggio di Scripting" %>
</td>
</tr>
</table>
```

Risultato:

Linguaggio di Scripting

Il seguente codice è uno script lato client che visualizza "Benvenuti al corso di VBScript"

```
<table border = "1">
<tr>
<td>
<script language="VBScript">
Document.Write "Benvenuti al corso di VBScript."
</script>
</td>
</tr>
</table>
```

Risultato:

Benvenuti al corso di VBScript.

Si noti come dentro il codice HTML sono stati inseriti dei tag che separano i vari linguaggi, ad esempio **<% %>** (separatori per ASP) e **<script> </script>** (separatori per VBScript lato client). Ciò permette di capire al software che va a decodificare le varie istruzioni se usare l'interprete HTML, VBScirpt o un altro ancora.
ASP dunque è un linguaggio lato server cioè il codice viene interpretato sul Server Web e spedito al browser sotto forma di HTML; quindi, nell'esempio sopra, il server web trasmette al browser
<table border="1"> <tr> <td> Linguaggio di Scripting </td> </tr> </table>
Il comando Response.Write, che è la funzione per visualizzare un testo, è già stata interpretata.
VBScript, invece, abbiamo detto che è un linguaggio lato **Client** (ma può essere anche lato Server come si è già detto e si vedrà in seguito), cioè il codice arriva al browser, che lo interpreta generando il risultato a video. Nei linguaggi lato Client si pone il problema del Browser: l'utente può avere vecchie versioni di Internet Explorer e non riuscire ad interpretare il codice VBScript.
Questo può essere un problema trascurabile dati i continui aggiornamenti che si possono trovare per i browser. Il problema specifico più grosso è che non tutti i comandi VBScript sono compatibili con Netscape Navigator, anche nelle ultime versioni. Si vedrà nelle prossime lezioni quali comandi sono incompatibili e come poter risolvere il problema.
Soffermandoci sui linguaggi lato Client, il primo fra tutti è JavaScript, nato per Netscape e adattato ad Explorer, che vede come alternativa il VbScript

2. Aggiunta di codice VBScript in una pagina

In HTML esiste il tag <script> che viene usato per includere il codice di un linguaggio di scripting. Nel caso del VBScript la sintassi è questa:

```
<script language="VBScript">
<!--
.....CODICE VBscript.....
-->
</script>
```

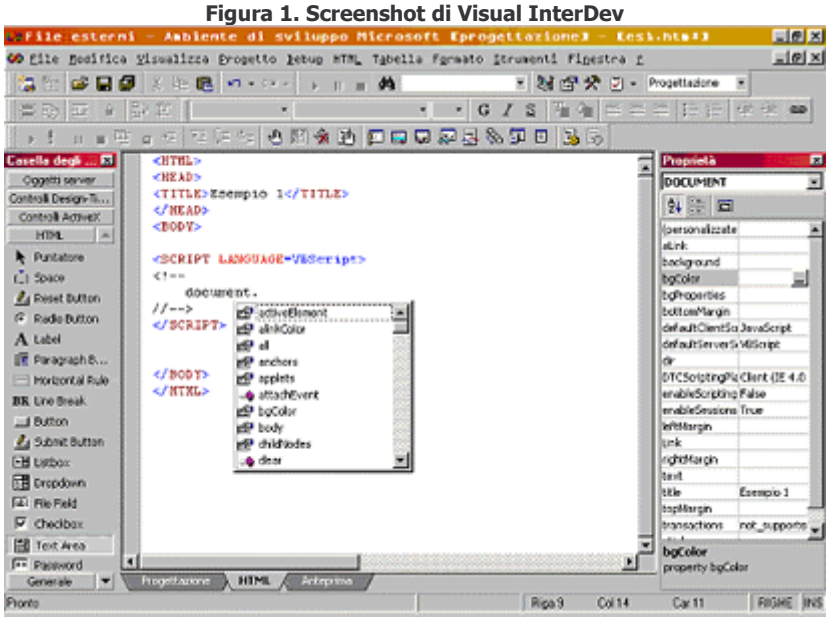
Quando la pagina HTML arriva al browser, questo interpreta il codice compreso nel tag <script>. Attraverso l'attributo Language viene indicato il linguaggio di scripting utilizzato. Il codice VBScript è inserito tra il tag commento <!-- -->, in questo modo il codice non viene visualizzato nei browser che non supportano il tag <script>.
Dove inserire questo codice? Dipende dalla sua funzione. Può essere inserito nel HEAD o nel BODY. Uno script che visualizza un messaggio, sicuramente va inserito nel BODY.
Passiamo ad un esempio pratico. Cominciamo con una pagina che visualizza il classico massaggio "Ciao Mondo". Occorre conoscere il comando che permette di visualizzare un messaggio: **document.write**.

```
<html>
<head>
<title>esempio 1</title>
```

```
</head>
<body>

<script language="VBScript">
<!--
  document.write "Ciao Mondo"
//-->
</script>


</body>
</html>
```

Quale editor usare per scrivere questo codice?
Sicuramente se si parlasse di HTML si potrebbe discutere ore per decidere se sia meglio un editor o un altro, ma per i linguaggi più sofisticati come il VBScript l'editor non dà un grande aiuto, è il programmatore che fa la differenza. Dal Block Notes a Front Page o DreamWeaver, le differenze sono minime per quanto riguarda il codice VBScript. L'unico editor che veramente aiuta a programmare è Visual Interdev. Questo editor, oltre a colorare il codice in funzione del tipo di istruzione, aiuta il programmatore durante la scrittura del codice con menù intelligenti, come in figura.

Figura 1. Screenshot di Visual InterDev



In alcune versioni di Office 2000 viene installata una versione ridotta di questo Editor che si trova nell'Hard Disk su: C:ProgrammiMicrosoft Visual StudioCommonIDEIDE98mse.exe
Qualunque sia l'editor che si usa il risultato del primo esempio sarà la scritta nel browser: «Ciao Mondo». Proviamo ad abbellirla: rendiamo la scritta Rossa, in Arial e in Grassetto. Ecco due modi per farlo:

```
<html>
<head>
<title>Esempio 1A</title>
</head>
<body>
<font face="Arial" color=#FF0000>
<b>

<script language="VBScript">
<!--
  document.write "Ciao Mondo"
//-->
</script>

</b>
</font>
</body>
</html>
```

oppure

```
<html>
<head>
<title>Esempio 1B</title>
</head>
<body>
<script language="VBScript">
<!--
  document.write "<font face=""Arial"" color=#FF0000><b>"
  documnet.write "Ciao Mondo"
```

```
  document.write "</b></font>"
//-->
</script>

</body>
</html>
```

È interessante notare come i due programmi facciano la stessa cosa, ma mentre il primo è un codice VBScript incastrato in un codice HTML, nel secondo si ha un codice HTML dentro il codice VBScript dentro un altro codice HTML. Questo ci fa capire che il browser interpreta prima il codice VBScript e poi l'HTML

Strumenti di base

3. Variabili

Il VBScript è un derivato del Visual Basic, e le variablili si comportano quasi nello stesso modo. Per dichiararle si usa l'istruzione **Dim.** Ad esempio
Dim variabile1
Dim variabile2
ecc.
Oppure
Dim variabile1, variabile2, ecc.
Chi è abituato ad altri linguaggi sa che dichiarando una variabile bisogna anche definire il tipo, cioè se conterrà un numero, una stringa, una data ecc.
In VBScript questa distinzione non c'è: non è necessario dichiarare il tipo delle variabili. La gestione di un numero o di una stringa viene affidata all'interprete VBScript. Questo rende molto più semplice programmare, ma bisogna stare attenti, come nel prossimo esempio:

```
<script language="VBScript">
<!--
  dim a,b,c
  a=inputbox("Primo Numero:")
  b=inputbox("Secondo Numero:")
  c=a+b
  document.write c
//-->
</script>
```

Nell'esempio, assegno i valori alle variabili a e b attraverso le **InputBox**, cioè finestre di dialogo che chiedono all'utente di inserire un valore. Se assegno il valore 3 la prima volta e 5 la seconda, ci si aspetta che la riga "document.write c" stampi il valore 8 ed invece visualizza 35.
Questo perché a e b vengono interpretate come stringhe, e il segno + invece di fare la somma concatena le 2 stringhe. Per risolvere il problema bisogna convertire le variabili da stringa a numerico.
Attenzione: i tipi non vanno dichiarati ma esistono e sono:

- **Boolean**
  Può contenere True o False.
- **Byte**
  Contiene un intero compreso tra 0 e 255.
- **Integer**
  Contiene un intero compreso tra -32.768 e 32.767.
- **Currency**
  Contiene un valore compreso tra -922.337.203.685.477,5808 e 922.337.203.685.477,5807.
- **Long**
  Contiene un intero compreso tra -2.147.483.648 e 2.147.483.647.
- **Single**
  Contiene un numero in virgola mobile e precisione singola compreso tra -3,402823E38 e 3,402823E38
- **Double**
  Contiene un numero in virgola mobile e precisione doppia compreso tra -1,79769313486232E308 e 1,79769313486232E308
- **Date**
  Contiene un numero che rappresenta una data compresa tra l'1 gennaio dell'anno 100 e il 31 dicembre del 9999.
- **String**
  Contiene una stringa di lunghezza variabile composta da un massimo di circa 2 miliardi di caratteri.
- **Object**
  Contiene un oggetto.

La conversione dei tipi viene gestita automaticamente dal VBScript quando è chiara. Nell'esempio precedente, se si facesse la sottrazione il risultato sarebbe corretto:

```
<script language="VBScript">
<!--
  dim a,b,c
  a=inputbox("Primo Numero:")
  b=inputbox("Secondo Numero:")
  c=a-b
  document.write c
//-->
</script>
```

Nel primo esempio, si ha il + che è utilizzato sia per la somma che per la concatenazione. Quindi a e b non vengono convertite in tipo numerico, ma sono lasciate in tipo stringhe. Per forzare questa conversione si utilizza la funzione **Cint:**

```
<script language="VBScript">
<!--
  dim a,b,c
  a=inputbox("Primo Numero:")
  b=inputbox("Secondo Numero:")
```

```
  c=Cint(a)+Cint(b)
  document.write c
//-->
</script>
```

L'istruzione **dim** serve a dichiarare una variabile, ma questa dichiarazione non è obbligatoria: il programma funzionerebbe anche senza la riga "dim a,b,c". Ciò rende la programmazione ancora più veloce perché non bisogna dichiarare le variabili, quando occorrono basta definirle.
Quando si realizza un programma complicato è consigliabile usare il comando "Option Explicit" all'inizio del programma Questo comando obbliga la dichiarazione delle variabili ed è utile se si sbaglia a scrivere il nome di una variabile. Infatti, se sbaglio il nome questa non esiste e si ha un errore.

```
<script language="VBScript">
<!--
  Option Explicit
  dim a,b,c
  a = inputbox("Primo Numero:")
  b = inputbox("Secondo Numero:")
  c = Cint(a)+Cint(b)
  document.write c
//-->
</script>
```

Quando si sceglie il **nome di una variabile** bisogna tener presenti le seguenti regole:

- Deve iniziare con una lettera dell'alfabeto.
- Non può includere punti e spazi.
- Non deve essere composta da più di 255 caratteri.
- Non c'è differenza tra lettere maiuscole e minuscole.

C'è poi la regola non obbligatoria di dare dei **nomi logici** alle variabili e così una variabile che deve contenere la data di nascita si potrà chiamare dtData_Di_Nascita, dove dt indica che è di tipo data. Questo permette di realizzare programmi più leggibili

4. Operatori

Gli operatori in VBScript si possono suddividere in tipi:

Operatori Aritmetici

		a=5 b=2	Risultato
+	Somma	c=a+b	7
-	Sottrazione	c=a-b	3
*	Moltiplicazione	c=a*b	10
/	Divisione	c=a/b	2,5
	Divisione Intera	c=ab	2
Mod	Modulo	c=a Mod b	1
^	Elevamento a potenza	c=a^b	25
&	Concatenamento di strighe	c=a & b	52

Per le formule complesse hanno precedenza addizione e sottrazione da sinistra verso destra.

Operatori di confronto

=	Uguaglianza
>=	Maggiore o uguale a
<=	Minore o uguale a
<>	Diverso
<	Minore
>	Maggiore

Il confronto tra due variabili restituisce Vero o Falso

Operatori Logici

Not	Negazione
And	Congiunzione Logica
Or	Disgiunzione Logica

Permettono le operazioni tra le variabili Booleane. Saranno molto utilizzate nel prossimo capitolo dove si vedranno le istruzioni condizionali.
Il seguente programma permette di verificare gli operatori aritmetici:

```
<script language="VBScript">
<!--
Option Explicit
dim a,b,c
a=5
b=2
document.write "c=a + b ---->" & a + b
```

```
document.write "<br>"
document.write "c=a - b ---->" & a - b
document.write "<br>"
document.write "c=a * b ---->" & a * b
document.write "<br>"
document.write "c=a / b ---->" & a / b
document.write "<br>"
document.write "c=a b ---->" & a b
document.write "<br>"
document.write "c=a Mod b ---->" & a Mod b
document.write "<br>"
document.write "c=a ^ b ---->" & a ^b
document.write "<br>"
document.write "c=a & b ---->" & a & b
//-->
</script>
```

Nel capitolo precedente abbiamo utilizzato l'operatore + per unire due stringhe, mentre qui si è usata la **&**. I due operatori sono equivalenti, ma è consigliabile usare sempre la **&** per distinguere meglio l'operazione di somma

5. Strutture condizionali

Vediamo ora di inserire qualche bivio nei programmi Per eseguire delle operazioni sotto certe **condizioni** il comando principale è IF. La sintassi è la seguente:

```
If <condizione> Then
  <operazioni se la condizione e vera>
Else
  <operazioni se la condizione e falsa>
End If
```

Nel seguente esempio si usa la funzione Month(), che restituisce il mese in formato numerico, e Now(), che restituisce la data e l'ora attuale:

```
<script language="VBScript">
<!--
if month(Now())=6 then
  document.write "È giugno"
end if
//-->
</script>
```

Se questo è il mese di giugno, stamperà "è giugno".
Per rendere più completo il programma:

```
<script language="VBScript">
<!--
if month(Now())=6 then
  document.write "É giugno"
else
  document.write "Non è giugno"
end if
//-->
</script>
```

Si possono anche creare delle strutture dentro altre strutture ossia delle **strutture annidate**:

```
<script language="VBScript">
<!--
if month(Now())<=6 then
  document.write "Siamo nel primo semestre dell'anno <br>"
  if month(Now())<=3 then
    document.write "e nel primo trimestre."
  else
    document.write "e nel secondo trimestre."
  end if
else
  document.write "Siamo nel secondo semestre dell'anno <br>"
  if month(Now())<=9 then
    document.write "e nel terzo trimestre."
  else
    document.write "e nel quarto trimestre."
  end if
end if
//-->
</script>
```

Si noti l'importanza di una formattazione di tipo "indent" (ossia con le rientranze) quando si usano strutture così complesse.

Un altro utilizzo dell'istruzione IF è quello di dare più condizioni possibili:

```
<script language="VBScript">
<!--
if month(Now())<=3 then
  document.write "Siamo nel primo trimestre."
elseif month(Now())<=6 then
  document.write "Siamo nel secondo trimestre."
elseif month(Now())<=9 then
  document.write "Siamo nel terzo trimestre."
else
  document.write "Siamo nel quarto trimestre."
end if
//-->
</script>
```

Si noti che in questo tipo di struttura solo un'operazione viene eseguita. Partendo dall'alto verso il basso, alla prima condizione vera viene eseguita l'operazione corrispondente e si esce dalla struttura. Infatti se fossimo a gennaio sarebbero verificate tutte le condizioni ma verrebbe stampata solo la prima frase.
Nella condizioni si possono usare gli **operatori logici** NOT, AND e OR per unire più condizioni.
Quando si utilizzano strutture con molte scelte dipendenti dal valore di un parametro, si può utilizzare la struttura con SELECT. Ecco lo stesso programma di prima con la struttura SELECT:

```
<script language="VBScript">
<!--
Select case month(Now())
  case 1,2,3
    document.write "Siamo nel primo trimestre."
  case 4,5,6
    document.write "Siamo nel secondo trimestre."
  case 7,8,9
    document.write "Siamo nel terzo trimestre."
  case else
    document.write "Siamo nel quarto trimestre."
end select
//-->
</script>
```

In questa struttura, viene definita la variabile da controllare con l'istruzione "Select case *Variabile*", e vengono poi elencati i valori possibili che può assumere la variabile con l'istruzione "case *valore1*,*valore2*,*valore3*"; seguono a questa istruzione le operazioni da eseguire. Nell'esempio la variabile è numerica, nel caso di variabile stringa i valori vanno messi tra virgolette: case "*valore1*","*valore2*","*valore3*

6. Strutture per cicli

Un ciclo serve a ripetere delle operazioni per un certo numero di volte, oppure finchè una certa condizione non avviene.
Il ciclo **FOR - NEXT** incrementa ad ogni ciclo una variabile. Quando questa variabile sarà arrivata ad una valore stabilito, il ciclo finirà.

```
<script language="VBScript">
<!--
Option Explicit
Dim i
For i=4 to 20
  document.write i & "<br>"
next
//-->
</script>
```

Questo programma stampa i valori da 4 a 20 incolonnati.
Attraverso il parametro STEP, posso anche contare al contrario e con passi diversi:

```
<script language="VBScript">
<!--
Option Explicit
Dim i
For i=20 to 4 step -2
  document.write i & "<br>"
next
//-->
</script>
```

Questo programma visualizzerà in numeri 20, 18, 16, 14, 12, 10, 8, 6, 4
Simile è il ciclo **FOR - EACH**. Questo ciclo si basa su un insieme di oggetti e viene ripetuto per ogni elemento dell'insieme.

In questo esempio viene usato un vettore. Dopo essere stato dichiarato e riempito, attraverso un ciclo For Each viene visualizzato:

```
<script language="VBScript">
<!--
Option Explicit
Dim vettore(5),elemento
vettore(0)="Html"
vettore(1)="Asp"
vettore(2)="Php"
vettore(3)="JavaScript"
vettore(4)="VBScript"
for each elemento in vettore
  document.write elemento & "<br>"
next
//-->
</script>
```

Il ciclo **DO LOOP** viene ripetuto finchè una condizione non diventa falsa. Il codice seguente viene ripetuto finchè non è trascorso un secondo da quando parte. Ad ogni ciclo incrementa la variabile conta. É un modo per apprezzare le qualità del proprio computer!

```
<script language="VBScript">
<!--
Option Explicit
dim TempoStart, TempoEnd, conta
conta=0
TempoStart=now()
TempoEnd=now()+1/(100000)
do While now()<=TempoEnd
  conta=conta+1
loop
document.write "Il ciclo si è ripetuto " & conta & " volte"
document.write " in 1 secondo"
//-->
</script>
```

Se la condizione non è mai verificata, il ciclo non è eseguito neanche una volta. Spostando la condizione al fondo, viene eseguito il test e poi verificata la condizione.
Nel seguente esempio la parola "Test Until" viene visualizzata anche se la condizione è falsa.

```
<script language="VBScript">
<!--
Option Explicit
dim a
a=10
do
  document.write "Test Until"
loop While a < 0
//-->
</script>
```

I cicli possono essre anche annidati. Nel codice seguente visualizziamo il contenuto di una matrice.

```
<script language="VBScript">
<!--
Option Explicit
dim cRighe,cColonne, matrice(3,2)
matrice (1,1)=1
matrice (2,1)=2
matrice (3,1)=3
matrice (1,2)=4
matrice (2,2)=5
matrice (3,2)=6

for cRighe=1 to 3
  for cColonne=1 to 2
    document.write matrice(cRighe,cColonne) & " "
  next
  document.write "<br>"
next
//-->
</script>
```

Le Funzioni

7. Funzioni data e ora

Ecco una carrellata di funzioni sulla data e l'ora:

date	Restituisce la data	**MyDate = Date** Il valore di MyDate è la data di sistema corrente.
time	Restituisce l'ora	**MyTime = Time** Restituisce l'ora di sistema corrente
now	Restituisce la data e l'ora	**MyVar = Now** Il valore di MyVar è la data e l'ora corrente
DateAdd	Aggiunge un intervallo di tempo ad una data	La funzione DateAdd non restituisce una data non valida. Nell'esempio seguente viene aggiunto un mese alla data 31 gennaio: **NewDate = DateAdd("m", 1, "31-Jan-95")** In questo caso la funzione restituisce 28-feb-95, non 31-feb-95. Se data h 31-gen-96, viene restituito 29-feb-96 perché il 1996 h un anno bisestile.
DateDiff	Restituisce il tempo tra due date	**DiffADate = "Giorni mancanti: " & DateDiff("d", Now, "1/1/2010")**
Day	Restituisce il giorno di una data	**MyDay = Day("19/10/1962")** Il valore di MyDay è 19
Month	Restituisce il mese di una data	**MyMonth = Month("19/10/1962")** Il valore di MyMonth è 10
Year	Restituisce l'anno di una data	**MyYear = Month("19/10/1962")** Il valore di MyYear è 1962
MonthName	Restituisce il nome del mese	**MyMonthName = Month("19/10/1962")** Il valore di MyMonthName è Ottobre
WeekDay	Restituisce un numero corrispondente al giorno della settimana	**MyWeekDay = Weekday("19/10/1962")** Il valore di MyWeekDay è 6 perché MyDate corrisponde a un venerdì
WeekDayName	Restituisce il nome del giorno della settimana	**MyWeekDay = WeekdayName("19/10/1962")** Il valore di MyWeekDay è venerdì
Second	Restituisce i secondi	**MySec = Second(Now)** Il valore di MySec è un numero che rappresenta i secondi
Minute	Restituisce i minuti	**MyVar = Minute(Now)** Il valore di MyVar è un numero che rappresenta i minuti
Hour	Restituisce l'ora	**MyVar = Hour(Now)** Il valore di MyVar è un numero che rappresenta l'ora in formato (0-23)
FormatDateTime	Restituisce la data e l'ora secondo un formato	**FormatDateTime(now, 0)** Restituisce 06/05/2007 17.06.50 **FormatDateTime(now, 1)** Restituisce domenica 6 maggio 2007 **FormatDateTime(now, 2)** Restituisce 06/05/2007 **FormatDateTime(now, 3)** Restituisce 17.06.50 **FormatDateTime(now, 4)** Restituisce 17.06

Ecco un piccolo programmino sulle funzioni data

```
<script language="VBScript">
<!--
Option Explicit
dim DataNascita
dim giorni(7)
giorni(2)="Lunedi"
giorni(3)="Martedi"
giorni(4)="Mercoledi"
giorni(5)="Giovedi"
giorni(6)="Venerdi"
```

```
giorni(6)="Sabato"
giorni(1)="Domenica"

DataNascita=InputBox ("Quando sei nato?")
document.write "Sei nato di " & giorni(weekday(DataNascita)) & "<br>"
document.write "Da quando sei nato sono passati " & datediff("d",DataNascita,now) & " giorni<br>"
document.write "Da quando sei nato sono passati " & datediff("m",DataNascita,now) & " mesi<br>"
document.write "Da quando sei nato sono passati " & datediff("ww",DataNascita,now) & " settimane<br>"
//-->
</script>
```

8. Funzioni sulle stringhe

Ecco una carrellata di funzioni sulle stringhe:

Asc	Restituisce il codice ASCII di un carattere	**MyNumber = Asc("A")** Restituisce 65 **MyNumber = Asc("a")** Restituisce 97 **MyNumber = Asc("Albero")** Restituisce 65
Chr	Restituisce un carattere dato il suo codice ASCII	**MyChar = Chr(65)** Restituisce A. **MyChar = Chr(97)** Restituisce a. **MyChar = Chr(62)** Restituisce >. **MyChar = Chr(37)** Restituisce %.
Instr	Restituisce la posizione di una stringa in un'altra	**SearchString ="visulabasicscript"** Stringa in cui eseguire la ricerca. **SearchChar = "i"** Esegue la ricerca della lettera "i". **MyPos = Instr(4, SearchString, SearchChar)** Confronto testuale a partire dalla posizione 4. Restituisce 10
LCase	Converte in minuscolo	**MyString = "VBSCript"** **LCaseString = LCase(MyString)** Il valore di LCaseString è "vbscript".
Left	Restituisce un numero di caratteri dalla sinistra di una stringa	**MyString = "VBSCript"** **LeftString = Left(MyString, 3)** ' Il valore di LeftString è "VBS".
Len	Restituisce il numero di caratteri di una stringa	**MyString = Len("VBSCRIPT")** ' MyString include 8 caratteri
LTrim	Toglie gli spazi a sinistra	**MyVar = LTrim(" vbscript ")** Il valore di MyVar è "vbscript
Mid	Restituisce un certo numero di caratteri da una stringa	**MyVar = Mid("VB Script è divertente!", 4, 6)** Il valore di MyVar è "Script".
Replace	Restituisce una data sottostringa con un'altra	**MyString = Replace("Linguaggio JavaScript ", "Java", "VB")** Restituisce "VBScript".
Rigth	Restituisce un numero di caratteri dalla destra di una stringa	**AnyString = "Salve gente!"** **MyStr = Right(AnyString, 1)** Restituisce "e". **MyStr = Right(AnyString, 6)** Restituisce " gente". **MyStr = Right(AnyString, 20)** Restituisce "Salve gente".
Rtrim	Toglie gli spazi a destra	**MyVar = RTrim(" vbscript ")** Il valore di MyVar è " vbscript".
Space	Crea una Stringa di spazi	**MyString = Space(10)** Restituisce una stringa con 10 spazi
Split	Crea un array di stringhe più piccole da una stringa	**MyString = "VBScriptXèXdivertente!"** **MyArray = Split(MyString, "x")** ' Il valore di MyArray(0) è "VBScript". ' Il valore di MyArray(1) è "è"

		' Il valore di)MyArray(2) è "divertente!".
StrComp	Confronta il valore di due stringhe	**MyStr1 = "Cane"** **MyStr2 = "Gatto"** **MyStr3 = "Gatto"** **MyComp = StrComp(MyStr3, MyStr2)** Restituisce 0. **MyComp = StrComp(MyStr1, MyStr2)** Restituisce -1. **MyComp = StrComp(MyStr2, MyStr1)** Restituisce 1.
String	Crea una stringa di un carattere ripetuto un certo numero di volte	**MyString = String(5, "*")** Restituisce "*****". **MyString = String(5, 42)** Restituisce "*****". **MyString = String(10, "ABC")** Restituisce "AAAAAAAAAA".
StrReverse	Inverte una stringa	**MyStr = StrReverse("VBScript")** Il valore di MyStr è "tpircSBV".
Trim	Taglia gli spazi da entrambi i lati di una stringa	**MyVar = Trim(" vbscript ")** Il valore di MyVar è "vbscript".
Ucase	Converte in maiuscolo	**MyWord = UCase("Salve gente!")** Restituisce "SALVE GENTE!".

Nell'esempio seguente vengono viste alcune delle funzioni qui sopra descritte

```
<script language="VBScript">
<!--
Option Explicit
dim nome,i

nome=inputbox("Inserisci il tuo nome")
document.write "Il tuo nome ha " & len(nome) & " lettere<br>"
document.write "Le prime tre lettere sono: " & left(nome,3) & "<br>"
document.write "Le ultime tre lettere sono: " & right(nome,3) & "<br>"
document.write "Il tuo nome al contrario è : " & strreverse(nome) & "<br>"
document.write "In maiuscolo: " & Ucase(nome) & "<br>"
document.write "In minuscolo: " & Lcase(nome) & "<br>"
document.write "Iniziale maiuscola: " & Ucase(left(nome,1)) & Lcase(mid(nome,2)) & "<br>"
for i =1 to len (nome)
  document.write mid(nome,1,i) & "<br>"
next
//-->
</script>
```

9. Funzioni matematiche

Ecco una carrellata di funzioni sulle stringhe:

Atn	Arco Tangente	**pi = 4 * Atn(1)** Calcola il valore di PiGreco
Cos	Coseno	**MyAngle = 1.3** Definisce l'angolo in radianti. **MySecant = 1 / Cos(MyAngle)** Calcola la secante.
Exp	*e* elevata ad una potenza	**MyAngle = 1.3** Calcola il seno iperbolico. **MyHSin = (Exp(MyAngle) - Exp(-1 * MyAngle)) / 2**
Log	Logaritmo naturale	**Number = 5** **LogNumber = Log(Number)**
Randomize	Inizializza il generatore di numeri casuali	**Randomize** Inizializza il generatore di numeri casuali. **Valore = Int((6 * Rnd) + 1)** Genera un valore casuale compreso tra 1 e 6. **document.write Valore**
Rnd	Restituisce un numero	Per ottenere interi casuali compresi in un determinato intervallo, è

	casuale	necessario utilizzare la seguente formula: **Int((limitesup - limiteinf + 1) * Rnd + limiteinf)**
Sin	Seno	**MyAngle = 1.3** Definisce l'angolo in radianti. **MyCosecant = 1 / Sin(MyAngle)** Calcola la cosecante
Sqr	Radice quadrata	**MySqr = Sqr(4)** Restituisce 2. **MySqr = Sqr(23)** Restituisce 4,79583152331272. **MySqr = Sqr(0)** Restituisce 0. **MySqr = Sqr(-4)** Genera un errore di runtime
Tan	Tangente	**MyAngle = 1.3** Definisce l'angolo in radianti. **MyCotangent = 1 / Tan(MyAngle)** Calcola la cotangente
FormatNumber	Formatta come Number	**MyAngle = 1.3** Definisce l'angolo in radianti. **MySecant = 1 / Cos(MyAngle)** Calcola la secante. **FormatNumberDemo = FormatNumber(MySecant,4)** Applica a MySecant il formato a quattro cifre decimali.
FormatCurrency	Formatta come valuta	**MyCurrency = FormatCurrency(1000)** i l valore di MyCurrency è L.1000,00
FormatPercent	Formatta come percentuale	**MyPercent = FormatPercent(2/32)** Il valore di MyPercent è 6,25%

Funzioni matematiche derivate
Nella tabella seguente sono elencate le funzioni matematiche non intrinseche che è possibile derivare da funzioni matematiche intrinseche.
Funzione Funzione equivalente derivata

Secante Sec(X) = 1 / Cos(X)
Cosecante Cosec(X) = 1 / Sin(X)
Cotangente Cotan(X) = 1 / Tan(X)
Seno inverso Arcsin(X) = Atn(X / Sqr(-X * X + 1))
Coseno inverso Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
Secante inversa Arcsec(X) = Atn(X / Sqr(X * X - 1)) + Sgn((X) -1) * (2 * Atn(1))
Cosecante inversa Arccosec(X) = Atn(X / Sqr(X * X - 1)) + (Sgn(X) - 1) * (2 * Atn(1))
Cotangente inversa Arccotan(X) = Atn(X) + 2 * Atn(1)
Seno iperbolico HSin(X) = (Exp(X) - Exp(-X)) / 2
Coseno iperbolico HCos(X) = (Exp(X) + Exp(-X)) / 2
Tangente iperbolica HTan(X) = (Exp(X) - Exp(-X)) / (Exp(X) + Exp(-X))
Secante iperbolica HSec(X) = 2 / (Exp(X) + Exp(-X))
Cosecante iperbolica Hcosec(X) = 2 / (Exp(X) - Exp(-X))
Cotangente iperbolica HCotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) - Exp(-X))
Seno iperbolico inverso Harcsin(X) = Log(X + Sqr(X * X + 1))
Coseno iperbolico inverso Harccos(X) = Log(X + Sqr(X * X - 1))
Tangente iperbolica inversa HArctan(X) = Log((1 + X) / (1 - X)) / 2
Secante iperbolica inversa Harcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
Cosecante iperbolica inversa HArccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) +1) / X)
Cotangente iperbolica inversa Harccotan(X) = Log((X + 1) / (X - 1)) / 2
Logaritmo in base N LogN(X) = Log(X) / Log(N)

10. Funzioni di conversione

Ecco una carrellata di funzioni di conversione:

Abs	Restituisce il valore assoluto di un numero	**MyNumber = Abs(50.3)** Restituisce 50,3 **MyNumber = Abs(-50.3)** Restituisce 50,3
CByte	Converte in un sottotipo Byte	**MyDouble = 125.5678** MyDouble è un valore di tipo Double. **MyByte = CByte(MyDouble)** Il valore di MyByte è 126.

CDate	Converte in un sottotipo Date	**MyDate = "October 19, 1962"** Definisce la data. **MyShortDate = CDate(MyDate)** Converte nel tipo di dati Date. **MyTime = "4:35:47 PM"** Definisce l'ora. **MyShortTime = CDate(MyTime)** Converte nel tipo di dati Date.
CDbl	Converte in un sottotipo Double	**MyCurr = CCur(234.456784)** MyCurr è un valore di tipo Currency (234.4567). **MyDouble = CDbl(MyCurr * 8.2 * 0.01)** Converte il risultato in un valore di tipo Double (19.2254576).
CInt	Converte in un sottotipo Integer	**MyDouble = 2345.5678** MyDouble è un valore di tipo Double. **MyInt = CInt(MyDouble)** Il valore di MyInt è 2346.
CLng	Converte in un sottotipo Long	**MyVal1 = 25427.45: MyVal2 = 25427.55** Sono valori di tipo Double. **MyLong1 = CLng(MyVal1)** Il valore di MyLong1 è 25427. **MyLong2 = CLng(MyVal2)** Il valore di MyLong2 è 25428.
CSng	Converte in un sottotipo Single	**MyDouble1 = 75.3421115: MyDouble2 = 75.3421555** **MySingle1 = CSng(MyDouble1)** Il valore di MySingle1 è 75,34211. **MySingle2 = CSng(MyDouble2)** Il valore di MySingle2 è 75,34216.
CStr	Converte in un sottotipo String	**MyDouble = 437.324** MyDouble è un valore di tipo Double. **MyString = CStr(MyDouble)** Il valore di MyString è "437,324".
Fix	Restituisce la parte intera di un numero	**MyNumber = Fix(99.2)** Restituisce 99. **MyNumber = Fix(-99.8)** Restituisce -99. **MyNumber = Fix(-99.2)** Restituisce -99.
Hex	Restituisce la stringa che rappresenta il valore esadecimale di un numero	**MyHex = Hex(5)** Restituisce 5. **MyHex = Hex(10)** Restituisce A. **MyHex = Hex(459)** Restituisce 1CB.
Int	Restituisce la parte intera di un numero	MyNumber = Int(99.8) ' Restituisce 99 MyNumber = Int(-99.8) ' Restituisce -100 MyNumber = Int(-99.2) ' Restituisce -100.
Oct	Restituisce una stringa che rappresenta il valore ottale di un numero	MyOct = Oct(4) ' Restituisce 4. MyOct = Oct(8) ' Restituisce 10. MyOct = Oct(459) ' Restituisce 713
Round	Arrotonda ad un numero di decimali specificato	Dim MyVar, pi pi = 3.14159 MyVar = Round(pi, 2) ' Il valore di MyVar è 3,14
Sgn	Restituisce un intero che rappresenta il segno di un numero	MyVar1 = 12: MyVar2 = -2.4: MyVar3 = 0 MySign = Sgn(MyVar1) ' Restituisce 1. MySign = Sgn(MyVar2) ' Restituisce -1. MySign = Sgn(MyVar3) ' Restituisce 0.

11. Creare funzioni e procedura

Spesso capita di scrivere righe di codici uguali in una stessa pagina. Per eliminare queste ripetizioni, si possono usare le **funzioni**. Un semplice esempio può essere una funzione che stampa la data quando richiesta.

```
<script language="vbscript">
function stampadata()
  document.write WeekdayName (weekday(date)) & " " & day(date) & " " & monthname(month(date))
& " " & year(date) end function
```

```
</script>

<script language="vbscript">
  stampadata()
</script>
```

Si noti come la funzione inizia con l'istruzione function e termina con end function.
Nello script più basso abbiamo la chiamata della funzione, semplicemente scrivendo il suo nome. Una funzione può anche ricevere delle variabili; ad esempio, creiamo una funzione che stampi il fattoriale.

```
<script language="vbscript">
function fattoriale(numero)
  dim c,risultato
  risultato=1
  for c=1 to numero
    risultato=risultato*c
  next
  document.write "Il fattoriale di " & numero & " è " & risultato
end function
</script>

<script language="vbscript">
fattoriale(5)
</script>
```

Infine una funzione può anche restituire un valore.

```
<script language="vbscript">
function fattoriale(numero)
  dim c,risultato
  risultato=1
  for c=1 to numero
    risultato=risultato*c
  next
  fattoriale=risultato
end function
</script>

<script language="vbscript">
document.write "Il fattoriale di 5 è " & fattoriale(5)
</script>
```

Per farci restituire il valore, è bastato mettere, prima dell'uscita della funzione, il nome della funzione uguale al valore da restituire.
Un altro modo per creare delle funzioni è attraverso il **comando SUB**. Con il comando SUB si creano procedure. Sia la procedura che le funzioni possono ricevere dei parametri, ma le procedure non restituscono un valore e perciò non si possono usare all'interno di espressioni. Quindi l'ultima istruzione
document.write "Il fattoriale di 5 è " & fattoriale(5)
non può essere generata con una SUB.
Questa procedura stampa una sequenza partendo da valore_iniziale fino a valore_finale con un passo valore_step.

```
<script language="vbscript">
sub sequenza(valore_iniziale,valore_finale,valore_step)
  dim c
  for c=valore_iniziale to valore_finale step valore_step
    document.write c & "<br>"
  next
end sub
</script>

<script language="vbscript">
sequenza 10,50,5
</script>
```

Come si intuisce le procedure sono molto simili alle funzioni sia nella chiamata che nell'implementazione.
Un lettore attento si sarà accorto che la chiamata della procedura sequenza è stata fatta senza tonde, mentre in precedenza si erano sempre usate le tonde. Vediamo di chiarire la differenza in questo esempio:

```
<script language="vbscript">
sub incrementa(valore)
  valore=valore+1
end sub
</script>

<script language="vbscript">
dim x
x=10
```

```
document.write "Il valore di x prima della procedura: " & x & "<br>"
incrementa x
document.write "Il valore di x dopo la procedura: " & x & "<br>"
incrementa (x)
document.write "Il valore di x dopo la procedura (con le parentesi): " & x & "<br>"
</script>
```

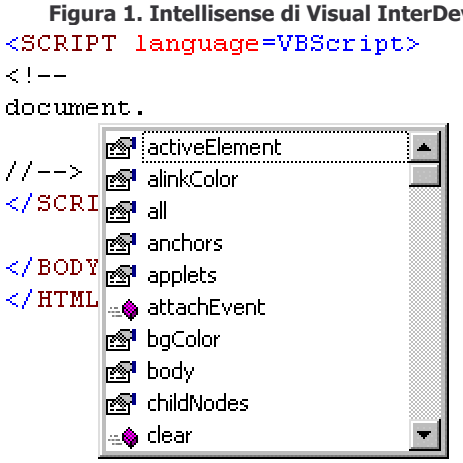
Questo codice produce:
Il valore di x prima della procedura: 10
Il valore di x dopo la procedura: 11
Il valore di x dopo la procedura (con le parentesi): 11
Quando passo x senza tonde il valore di x resta modificato, mentre nel secondo caso x non varia.
Le variabili sono passate alla procedura per riferimento o per valore. Nel passaggio per riferimento passo x, e tutte le operazioni su di essa restano anche dopo che si è conclusa la procedura. Nel passaggio per valore viene passato solo il valore di x, quindi dopo la procedura, la variabile è rimasta intatta. Nella prima chiamata alla procedura, quella senza tonde, si ha un passaggio per riferimento (ByRef), nella seconda, con le tonde, un passaggio per valore (ByVal). Lo stesso principio è valido anche per le funzioni

VBScript e gli oggetti

12. Gli oggetti

Come in Visual Basic, anche in VBScript troviamo il concetto di **oggetto**.
Un oggetto è un'entità che ha delle proprietà, delle azioni (metodi) e genera degli eventi. Ad esempio la pagina HTML, che è l'oggetto Document, ha come proprietà il colore di sfondo, come azioni la scrittura di un messaggio e come evento il caricamento della pagina.
Vedremo in questa parte del corso i principali oggetti utilizzati in VBScript. Alcuni sono estensioni degli oggetti **JavaScript** (Document, ad esempio), altri sono propri del **VbScript**.
Cominciamo analizzando l'oggetto Document. Se avete la fortuna di usare come editor Visual Interdev, provate ad inserire un blocco script e scrivete document.

Figura 1. Intellisense di Visual InterDev



Apparirà un menù come in figura. In questo menù si trovano elencate proprietà e azioni; le proprietà si distinguono per questa icona  , mentre i metodi si riconoscono da questa  .

Ecco le principali proprietà:

document.alinkcolor: restituisce o imposta il colore per i collegamenti attivi.
document.linkcolor: restituisce o imposta il colore per i collegamenti.
document.vlinkcolor: restituisce o imposta il colore per i collegamenti visitati.
document.bgcolor: restituisce o imposta il colore di sfondo della pagina.
document.title: restituisce o imposta il titolo della pagina.
document.url: restituisce l'url della pagina.
Questo script restituisce le proprietà di questa pagina:

```
<script language="VBScript">
<!--
document.write "alinkColor " & document.alinkColor & "<br>"
document.write "linkColor " & document.linkColor & "<br>"
document.write "vlinkColor " & document.vlinkColor & "<br>"
document.write "bgcolor " & document.bgcolor & "<br>"
document.write "title " & document.title & "<br>"
document.write "url " & document.url & "<br>"
//-->
</script>
```

Ecco il risultato della sua esecuzione in questa pagina
alinkColor #0000ff
linkColor #0000ff
vlinkColor #800080
bgcolor #dedede
title Gli oggetti | Guida VBScript | Javascript.HTML.it
url http://javascript.html.it/guide/lezione/919/gli-oggetti/

Analizziamo ora alcuni dei suoi metodi:
document.focus: porta in primo piano la finestra del browser.
document.write: come abbiamo già avuto modo di valutare, questo metodo permette di scrivere un testo sulla pagina HTML
document.writeln: come l'evento write ma in più aggiunge un "a capo" al fondo della stringa. Attenzione, però: non inserisce un
 ma un carriage return, visibile quindi nel codice HTML
Un evento si dichiara con una procedura il cui nome è composto dall'oggetto e dal nome dell'evento. Ad esempio questo è l'evento legato al click sulla pagina.

```
<script language="vbscript">
<!--
```

```
Sub document_onclick
  msgbox "Hai cliccato!"
End Sub
-->
</script>
```

Gli eventi principali per l'oggetto Document sono:
document_onclick
document_onkeypress
document_onkeyup
document_onkeydown
document_onmouseou
document_onmouseover
Chi conosce JavaScript saprà che ai tag di un modulo possono essere associati dei nomi e possono essere utilizzati come oggetti. Ad esempio il tag di una textbox sarà
<input type=text name="nomecampo">
<input type="text" name="nomecampo">
In qualsiasi punto della pagina posso operare sull'oggetto grazia al suo nome. Con il seguente codice inserisco una scritta nella casella di testo

```
<script language="vbscript">
nomecampo.value = "Viva VBScript"
</script>
```

Con VBScript si estende il concetto di oggetto e cosi' qualsiasi tag che ha un nome è un oggetto con le sue proprietà, i suoi metodi e i suoi eventi.
Molto spesso le proprietà di un oggetto sono gli attributi del tag HTML. Nel esempio precedente si è infatti usato l'attributo value.
I metodi derivano dalle azioni che l'utente compie e così abbiamo nomeform.submit() per inviare un form oppure nomeform.reset() per pulirlo.
Gli eventi più utilizzati sono:
onclick: quando si fa un click sull'oggetto
onfocus: quando si attiva l'oggetto (ad esempio cliccando all'interno di una textbox)
onblur: quando si lascia un oggetto
onmouseover: quando si è con il mouse sopra l'oggetto
onmouseout: quando si esce con il mouse dall'oggetto
onchange: quando si apportano modifiche all'oggetto

13. Gli oggetti di VBScript

Gli oggetti visti nel capitolo precedentederivano dagli oggetti JavaScript. Questi che vediamo ora sono gli oggetti del VBScript. Grazie a questi oggetti il VBScript si dimostra un linguaggio più completo del JavaScript anche se quest'ultimo resta quello più compatibile dai browser.
Per utilizzare un oggetto bisogna creare un'istanza cioè dichiararlo ed associarli un nome. Per farlo si usa il comando CreateObject. La sintassi per creare un oggetto è:
Set nomeoggetto = CreateObject("tipo di oggetto")
L'oggetto FileSystemObject permette di lavorare con i file e le cartelle dell'hard disk. Questo oggetto serve a creare altri oggetti che saranno la cartella, il file o il drive. Nel esempio seguente diamo un occhiata al'hard disk:

```
<script language="vbscript">

Function ShowFile(folderspec)
  Dim fso, folder, file, FileItem

  Set fso = CreateObject("Scripting.FileSystemObject")
  Set folder = fso.GetFolder(folderspec)
  Set file = folder.Files

  For each FileItem in file
    document.write FileItem & "<br/>"
  Next
End Function

ShowFile("c:")
</script>
```

L'avvio di questa funzione in una pagina HTML ci avverte che si sta creando una situazione di pericolo. Infatti la seguente funziona visualizza i file ma come vedremo esistono anche comandi per cancellare i file e le cartelle!
Tornando alla funzione ShowFile, per prima cosa è stato creato l'oggetto fso, cioè l'oggetto chepermette di lavorare con il file system. Con il comando successivo "Set folder = fso.GetFolder(folderspec)" ho creato l'oggetto cartella ed infine con "Set file = folder.Files" creo l'oggetto file.
Il resto del programma usa metodi e proprietà di questi oggetti, cheverranno ora decritti
Il FileSystemObject ha solo una proprietà, Drive, che restituisce l'elenco di tutti i drive disponibili sul sistema.
Esistono invece numerosi metodi per lavorare con file e cartelle:

DriveExists(lettera drive) restituisce True se la lettera del dirve specificato esiste.
GetDrive (lettera drive) restituisce un oggetto Drive corrispondente alla lettera specificata
GetDriveName (lettera drive) restituisce il nome del Drive corrispondente alla lettera specificata
BuildPath(percorso, nome) aggiunge file o cartelle specificate da nome al percorso corrente
CopyFolder(origine, destinazione, sovrascrivi) copia una cartella dalla posizione origine a quella destinazione, se esiste già viene sovrascritta in funzione del parametro sovrascrivi
CreateFolder(nome cartella) crea una cartella
DeleteFolder(nome cartella, force) cancella una cartella, con il parametro force=true cancello anche le cartelle in sola lettura
FolderExists(nome cartella) restituisce true o false se la cartella specificata esiste o no
GetFolder(cartella) restituisce un oggetto cartella corrispondente alla cartella specificata.
GetParentFolderName(cartella) restituisce il nome della cartella superiore a quella specificata
MoveFolder(origine, destinazione) sposta una cartella
CopyFile(origine,destinazione, sovrascrivi) copia un file dalla posizione origine a quella destinazione, se esiste già viene sovrascritta in funzione del parametro sovrascrivi
CreateTextFile(persorso, sovrascrivi, unicode) crea un un file di testo nel percorso specificato. Il parametro sovrascrivi, permette di sovrascivere un file già esistente, il parametro unicode permette di salvare in formato ASCII o Unicode
DeleteFile(nome file, force) cancella un file, con il parametro force=true cancello anche i file in sola lettura
FileExists(percorso file) restituisce true o false se il file specificato esiste o no
GetExtensionName(percorso) restituisce l'estensione del file
GetFile(percorso) restituisce un oggetto file corrispondente al file specificato
MoveFile(origine, destinazione) sposta un file
OpenTextFile(nome file, iomode, crea, formato) crea o apre un file di testo utilizzando iomode per leggere, scrivere o aggiungere in formato ASCII o Unicode.
Il seguente codice crea unFileSystemObject ed elenca i drive presenti nella macchina locale:

```
Dim fso
Dim disco

'Creazione dell'oggetto FileSystemObject
Set fso = CreateObject("Scripting.FileSystemObject")

document.write "I dischi su questo computer sono:"
For Each disco in fso.Drives
  document.write "<BR>Drive =" & disco
next
```

14. L'oggetto drive

L'oggetto Drive permette di recuperare le informazioni sui drive disponibili.
Le proprietà dell' oggetto Drive sono:

AvaiableSpace restituisce la quantità di spazio disponibile sul drive
DriveLetter restituisce la lettera del drive
DriveType restituisce il tipo di drive (unknow, removable, fixed, network, CdRom, RamDisk)
FileSystem restituisce il tipo di file system (FAT, NTFS, CDFS)
FreeSpace restituisce lo spazio libero disponibile nel drive
IsReady se è pronto restituisce true
Path restituisce la lettera assegnata al drive
RootFolder restituisce la cartella principale
SerialNumber restituisce il numero di serie che identifica il volume del disco
ShareName restituisce il nome di rete per un drive condiviso
TotalSize restituisce le dimensioni totali del drive
VolumName restituisce o imposta in nome del volume per i drive locali
Applichiamo queste proprietà in un semplice esempio

```
<script language="vbscript">
Dim drive
drive="C:"

Set fso = CreateObject("Scripting.FileSystemObject")

document.write "Informazioni sull'Hard Disk<br/>"
document.write "Lettera del Drive: " & fso.GetDrive(drive).DriveLetter & "<br/>"
document.write "Tipo: " & fso.GetDrive(drive).DriveType & "<br/>"
document.write "File System: " & fso.GetDrive(drive).FileSystem & "<br/>"
document.write "Spazio Disponibile: " & fso.GetDrive(drive).AvailableSpace & "<br/>"
document.write "Spazio Libero: " & fso.GetDrive(drive).FreeSpace & "<br/>"
document.write "É pronto: " & fso.GetDrive(drive).IsReady & "<br/>"
document.write "Path: " & fso.GetDrive(drive).Path & "<br/>"
document.write "Cartella Principale: " & fso.GetDrive(drive).RootFolder & "<br/>"
document.write "Numero Seriale: " & fso.GetDrive(drive).SerialNumber & "<br/>"
document.write "Nome Condiviso: " & fso.GetDrive(drive).ShareName & "<br/>"
document.write "Dimensione Totale: " & fso.GetDrive(drive).TotalSize & "<br/>"
document.write "Nome del Volume: " & fso.GetDrive(drive).VolumeName & "<br/>"
</script>
```

Questo codice restiutirà un risultato simile a:
Informazioni sull'Hard Disk
Lettera del Drive: C
Tipo: 2
File System: FAT32
Spazio Disponibile: 241664000
Spazio Libero: 241664000
É pronto: Vero
Path: C:
Cartella Principale: C:
Numero Seriale: 442505702
Nome Condiviso:
Dimensione Totale: 2147155968
Nome del Volume:
L'esecuzione di codici script dove vengono usati gli oggetti possono generare un messaggio di avvertimento. Infatti come vedremo con l'oggetto FileSystenObject è possibile cancellare file e cartelle

15. L'oggetto cartella

Iniziamo ad utilizzare quelli oggetti "pericolosi" se non usati nel modo opportuno, oppure usati per far danno. Vedremo come cancellare, spostare e copiare una cartella. Pensate inserire questo codice in una email o in una pagina internet!!! Avete fatto una sorta di virus.
L'oggetto cartella ha le seguenti Proprietà:

Attributes restituisce gli attributi di una cartella (normale, sola lettura, nascosta, di sistema, volume, archivio, compresso)
DateCreated restituisce la data e l'ora della creazione della cartella
DateLastAccessed restituisce la data e l'ora dell'ultimo accesso
DateLastModifed restituisce la data e l'ora dell'ultima modifica alla cartella
Drive restituisce la lettera del drive per la cartella
Files restituisce tutti i file presenti nella cartella
IsRootFolder restituisce True se è la cartella principale
Name imposta o restituisce il nome della cartella
ParentFolder restituisce un oggetto che punta alla cartella superiore.
Path restituisce il percorso della cartella
ShortName restituisce il nome Dos della cartela
ShortPath restituisce il nome Dos del percorso della cartella
Size restituisce la dimensione di file e sottocartelle presenti nella cartella corrente
Subfolder restutuisce l'elenco di tutte le sottocartelle presenti nella cartella corrente
Type restituisce una descrizione della cartella corrente come stringa
I metodi dell'oggetto cartella permettono di copiare, cancellare e spostare una cartella. Inoltre c'è un metodo che permette di creare un file di testo.
Copy(destinazione,sovrascrittura) Copia la cartella corrente nella posizione di destinazione, sovrascivendo una cartella esistente se la proprietà sovrascrittura è impostata a True
Delete(force) Cancella la cartella corrente e i suoi contenuti. Il parametro force impostato a true elimina anche le cartelle in sola lettura.
Move(destinazione) sposta la cartella corrente è il suo contenuto in una posizione specificata.
CreateTextFile(percorso,overwrite,unicode) crea un file di testo con il nome specificato nel parametro percorso, overwrite impostato a true cancella un file già esistente, mentre il parametro unicode impostato a true permette di creare un file nel formato Unicode.
Vediamo qualche esempio dove sono applicati i concetti appena incontrati.
Questo esempio, partendo dalla cartella pricipale, elenca tutte le sue sottocartelle.Si noti la crezione dei tre oggetti che permettono di realizzare questo script: ognuno e un sotto oggetto dell'altro.

```
<script language="vbscript">
Dim Drive,folder, subfolder,elemFolder
Drive="C:"

Set fso = CreateObject("Scripting.FileSystemObject")
Set folder = fso.GetDrive (Drive).RootFolder
Set subfolder = folder.Subfolders

Document.write "Le sottocartelle della cartella principale sono:<br>"
For each elemFolder in subfolder
  document.write elemFolder & "<br>"
next
</script>
```

Con il prossima esempio vediamo come copiare una cartella. Per poter eseguire lo script occorre prima creare una cartella chiamata prova in C:

```
<script language="vbscript">
Dim Drive, folder, folderorigine, folderdestinazione
folderorigine="c:prova"
folderdestinazione="c:windows"
```

```
Set fso = CreateObject("Scripting.FileSystemObject")
set folder = fso.GetFolder (folderorigine)

folder.Copy folderdestinazione,true
document.write "La cartella " & folderorigine & " è stata copiata in " & folderdestinazione & "."
</script>
```

Infine vediamo i file contentuti nell'Hard Disk:

```
<script language="vbscript" >
Dim Drive, folder, file, elemfile
Set fso = CreateObject("Scripting.FileSystemObject")
Set folder = fso.GetFolder ("c:")
Set file = folder.Files
document.write "I file nella cartella C: sono <br>"
For Each elemfile in file
  document.write elemfile & "<BR>"
next
</script>
```

16. L'oggetto file

Continuiamo la discesa discesa nel nostro computer e dopo l'oggetto drive e cartella ora tocca all'oggetto file. Utile per copiare, spostare e cancellare un file, ci permette anche di conoscere gli attributi di un file, la dimensione ed altre informazioni che andiamo subito a scoprire
L'oggetto file ha le seguenti Proprietà:

Attributes restituisce gli attributi di un file (normale, sola lettura, nascosto, di sistema, volume, archivio, compresso)
DateCreated restituisce la data e l'ora della creazione del file
DateLastAccessed restituisce la data e l'ora dell'ultimo accesso
DateLastModifed restituisce la data e l'ora dell'ultima modifica al file
Drive restituisce la lettera del drive per il file
Name imposta o restituisce il nome del file
ParentFolder restituisce un oggetto che punta alla cartella contenente il file.
Path restituisce il percorso del file
ShortName restituisce il nome Dos del file
ShortPath restituisce il nome Dos del percorso del file
Size restituisce la dimensione del file corrente
Type restituisce una descrizione del file corrente come stringa
I metodi dell'oggetto file sono simili a quelli dell'oggetto cartella.
Copy(destinazione,sovrascrittura) Copia il file corrente nella posizione di destinazione, sovrascivendo un file esistente se la proprietà sovrascrittura è impostata a True
Delete(force) Cancella il file corrente. Il parametro force impostato a true elimina anche i file in sola lettura.
Move(destinazione) sposta il file corrente in una posizione specificata.
CreateTextFile(percorso,overwrite,unicode) crea un file di testo con il nome specificato nel parametro percorso, overwrite impostato a true cancella un file già esistente, mentre il parametro unicode impostato a true permette di creare un file nel formato Unicode.
OpenAsTextStream(percorso,overwrite,unicode) crea un file di testo con il nome specificato nel parametro percorso, overwrite impostato a true cancella un file già esistente, mentre il parametro unicode impostato a true permette di creare un file nel formato Unicode.
Vediamo qualche esempio dove sono applicati i concetti appena incontrati.
Questo esempio mostra le proprietà di un file.

```
<script language="vbscript" >
Dim FilePath,OggettoFile
FilePath="c:autoexec.bat"

Set fso = CreateObject("Scripting.FileSystemObject")
set OggettoFile = fso.GetFile (FilePath)

document.write "Nome File " & OggettoFile.Name & "<br/>"
document.write "Drive " & OggettoFile.Drive & "<br/>"
document.write "Attributi " & OggettoFile.Attributes & "<br/>"
document.write "Cartella " & OggettoFile.ParentFolder & "<br/>"
document.write "Data Creazione " & OggettoFile.DateCreated & "<br/>"
document.write "Data Ultimo Accesso " & OggettoFile.DateLastAccessed & "<br/>"
document.write "Data Ultima Modifica " & OggettoFile.DateLastModified & "<br/>"
document.write "Dimensione " & OggettoFile.Size & " bytes<br/>"
document.write "Tipo " & OggettoFile.Type & "<br/>"
document.write "Percorso " & OggettoFile.Path & "<br/>"
document.write "Nome Dos " & OggettoFile.ShortName & "<br/>"
document.write "Percorso Dos " & OggettoFile.ShortPath & "<br/>"
</script>
```

Lanciando questo script sul browser, dopo il messaggio di avvertimento, si avrà il seguente Output:

Nome File AUTOEXEC.BAT
Drive c:
Attributi 32
Cartella C:
Data Creazione 16/02/01 23.50.11
Data Ultimo Accesso 08/07/01
Data Ultima Modifica 28/04/01 22.51.06
Dimensione 202 bytes
Tipo File batch MS-DOS
Percorso C:AUTOEXEC.BAT
Nome Dos AUTOEXEC.BAT
Percorso Dos C:AUTOEXEC.BAT
Con il prossima esempio vediamo come copiare un file. Per poter eseguire lo script occorre prima creare una cartella chiamata prova in C:

```
<script language="vbscript">
Dim Drive, OggFile, origine, destinazione

origine="c:autoexec.bat"
destinazione="c:prova"

Set fso = CreateObject("Scripting.FileSystemObject")
set OggFile = fso.GetFile (origine)
OggFile.Copy destinazione,true
document.write "Il file " & origine & " è stato copiato in " & destinazione & "."
</script>
```

Infine vediamo come cancellare il file appena creato:

```
<script language="vbscript" >
Dim OggFile, origine

origine="c:provaautoexec.bat"
Set fso = CreateObject("Scripting.FileSystemObject")
Set OggFile = fso.GetFile (origine)
OggFile.Delete

document.write "Il file " & origine & " è stato cancellato."
</script>
```

17. L'oggetto text-stream

L'ultimo oggetto del file system è TextStream. Grazie a questo oggetto è possibile creare, leggere e scrivere in un file di testo.
L'oggetto TextStream ha le seguenti Proprietà:

AtEndOfLine restituisce True se il puntatore del file è alla fine di una riga di un file
AtEndOfStream restituisce True se il puntatore del file è alla fine di un file
Column restituisce il numero di colonna del carattere corrente
Line restituisce il numero di riga del carattere corrente
I metodi dell'oggetto TextStream permettono di navigare, leggere e scrivere file.
Close chiude il file corrente
Read (numerocaratteri) legge il numero di caratteri specificato da un file.
ReadAll legge tutti i caratteri di un file.
ReadLine legge una riga da un file.
Skip(numerocaratteri) salta il numero di caratteri specificato da un file.
SkipLine salta alla riga successiva.
Write(stringa) scrive una stringa in un file.
WriteLine crea una nuova riga.
WriteBlankLines(numero) crea un numero specificato di righe vuote.

Vediamo qualche esempio dove sono applicati i concetti appena incontrati.
Iniziamo con un esempio che ci permette di scrivere su un file di testo:

```
<script language="vbscript">
Dim OggFile, origine, OggTextStream
origine="c:textstream.txt"
Set fso = CreateObject("Scripting.FileSystemObject")
fso.CreateTextFile (origine)
Set OggFile = fso.GetFile (origine)
Set OggTextStream = OggFile.OpenAsTextStream(2)
OggTextStream.WriteLine "Lezione di VBScript sull'oggetto TextStream"
OggTextStream.WriteLine
OggTextStream.WriteLine "Terza riga dell'esempio."
OggTextStream.close
```

```
document.write "Scrittura eseguita correttamente."
</script>
```

Con il comando CreateTextFile viene creato il file di testo e tramite il comando OpenAsTextStream il file viene aperto. Per aprire un file ci sono 3 modalità:

- Sola lettura 1
- Sola lettura 1
- Scrittura 2
- Scrittura in coda 8

Nell'esempio il file è stato aperto con il parametro 2 cioè in scrittura. Se il file esiste già, viene sovrascritto.
Passiamo ora ad un esempio che ci permette di leggere il file appena creato:

```
<script language="vbscript" >
Dim OggFile, origine, OggTextStream, stringa
origine="c:textstream.txt"
Set fso = CreateObject("Scripting.FileSystemObject")
Set OggFile = fso.GetFile (origine)
Set OggTextStream = OggFile.OpenAsTextStream(1)

Do While Not OggTextStream.AtEndOfStream
  stringa = OggTextStream.ReadLine
  document.write stringa & "<br>"
loop
OggTextStream.close
</script>
```

Attraverso il cilclo do - loop è possibile leggere tutto il file di testo. Si noti come il parametro di OpenAsTextStream sia impostato ad 1, dato che il file è aperto in lettura

18. Utilizzo di VBScript con gli oggetti

Microsoft Visual Basic Scripting Edition e MicrosoftÂ® Internet Explorer gestiscono sia i controlli ActiveX, precedentemente definiti controlli OLE, che gli oggetti Java. Se si utilizza Microsoft Internet Explorer ed è stato installato il controllo Label, è possibile visualizzare la pagina generata dal codice associato.
Per includere un oggetto, è necessario utilizzare il tag <OBJECT>, mentre per impostare i valori iniziali delle proprietà dell'oggetto, è necessario utilizzare il tag <PARAM>. L'utilizzo del tag <PARAM> è equivalente all'impostazione dei valori iniziali delle proprietà per un controllo di un form in Visual Basic. Nell'esempio seguente i tag <OBJECT> e <PARAM> consentono di aggiungere il controllo Label (etichetta) ActiveX a una pagina:

```
<object
 classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
 id=lblActiveLbl
 width=250
 height=250
 align=left
 hspace=20
 vspace=0
>
<param name="Angle" value="90">
<param name="Alignment" value="4">
<param name="BackStyle" value="0">
<param name="Caption" value="Etichetta desiderata">
<param name="FontName" value="Verdana, Arial, Helvetica">
<param name="FontSize" value="20">
<param name="FontBold" value="1">
<param name="FrColor" value="0">
</object>
```

È possibile impostare proprietà e richiamare metodi esattamente come per i controlli nei form. Il codice seguente, ad esempio, include i controlli <FORM> che consentono di richiamare due proprietà del controllo Label:

```
<form name="LabelControls">
<input type="TEXT" name="txtNewText" SIZE=25>
<input type="BUTTON" name="cmdCambiaLo" value="Modifica testo">
<input type="BUTTON" name="cmdRuotalo" value="Ruota etichetta">
</form>
```

Quando il form è stato definito, una routine di eventi del pulsante cmdChangeIt consente di modificare il testo dell'etichetta:

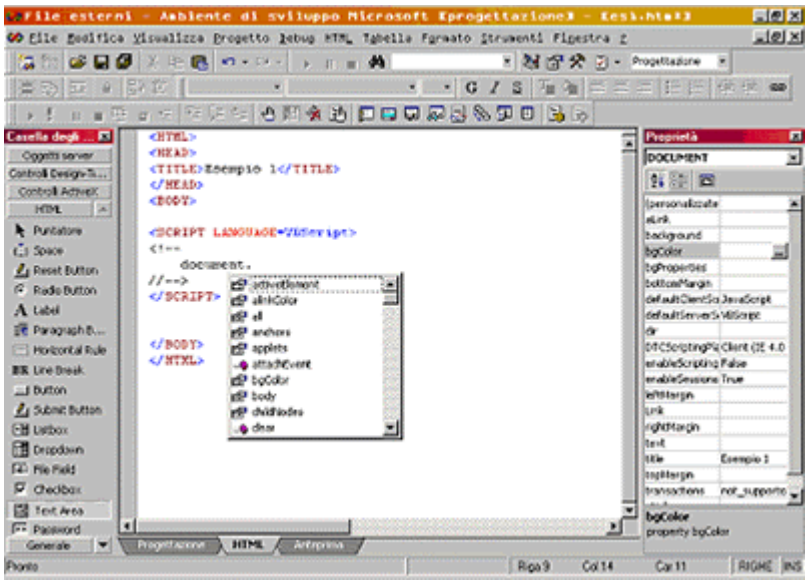
```
<script language="VBScript">
<!--
Sub cmdChangeIt_onClick
  Dim MioForm
  Set MioForm = Document.LabelControls
  lblActiveLbl.Caption = MioForm.txtNewText.Value
End Sub
```

```
-->
</script>
```

Nel codice i riferimenti ai controlli e ai valori inclusi nei form vengono specificati esattamente come nell'esempio di Convalida semplice.
Nel sito Web di Microsoft all'indirizzo (http://www.microsoft.com) sono disponibili diversi controlli ActiveX utilizzabili in Internet Explorer, nonché informazioni dettagliate su proprietà, metodi, eventi e identificatori di classe (CLSID) dei controlli. Per ulteriori informazioni sul tag <object>, vedere la pagina Internet Explorer 4.0 Author's Guide and HTML Reference.
Nelle versioni precedenti di Internet Explorer, gli attributi CLSID dovevano essere racchiusi tra parentesi graffe ({}), diversamente da quanto indicato nella specifica W3C. Nella versione corrente, l'utilizzo delle parentesi graffe genera un messaggio in cui si avvisa che nella pagina viene utilizzata una versione non aggiornata del tag <object>.

19. VBScript: introduzione

Lo scopo di questo Tutorial è quello di offrire una guida pratica a questo linguaggio. Se non l'hai già fatto, sarebbe utile dare un'occhiata alla guida a VBScript, per acquisire le conoscenze di base su questo linguaggio.
Prima di iniziare, diamo uno sguardo agli strumenti che abbiamo a disposizione. Per **scrivere il codice** basta il semplice blocco note, ma consiglio di usare qualche editor più evoluto. Per programmare in VBScript, il migliore è senza dubbio **Visual Interdev**, che fa parte del pacchetto Visual Studio. Questo tool colora il codice, permettendo di distinguere funzioni, stringhe di testo e variabili, inoltre offre menu intelligenti che mostrano le proprietà di un oggetto dopo aver digitato il punto.
Come in figura:



Inoltre, immaginando di dover scrivere la parola "document", Visual Interdev aiuta. Infatti basta inserire "doc" e premere CTRL + barra spaziatrice. In questo modo, si attiva il completamento automatico. Questo è utile specialmente all'inizio, quando non si ricorda la sintassi dei comandi. Chiaramente funziona con qualsiasi comando; in caso di ambiguità viene proposto un menu per la scelta, ad esempio digitando "dat" e control+spazio.
Dopo aver lodato questo prodotto, ricordo però che il Blocco Note di Windows è sufficente per programmare in VBScript; qualsiasi strumento più intelligente non può che essere d'aiuto al programmatore.
Nel caso occorresse, facciamo un rapido ripasso degli **oggetti**: l'automobile è un oggetto; ogni automobile ha dalle proprietà, come il livello di carburante, il colore ecc. L'automobile può compiere delle azioni (o metodi), come accelerare, decelerare ecc.
Tornando a VBScript, document è l'oggetto che si riferisce alla pagina HTML; document ha delle proprietà come title, che indica il titolo della pagina, e delle azioni come write, che permette di scrivere del testo su una pagina.
All'interno della pagina HTML si può inserire del codice VBScript in qualsiasi punto, basta usare la sintassi:

```
<script language=vbscript>
<!--
//-->
</SCRIPT>
```

Il tag "SCRIPT" identifica la porzione di codice VBScript. I tag <!-- //--> permettono ai vecchi browser, che non conoscono il tag "SCRIPT", di saltare il codice senza visualizzarlo.
Dopo una breve introduzione, siamo pronti per la creazione delle nostre pagine con VBScript.

20. Formattare la data

Prima di iniziare a realizzare gli esercizi proposti in queste pagine, è consigliabile rileggere i capitoli sulle funzioni della guida a VBScript. Ti saranno utili le funzioni data/ora e le funzioni sulle stringhe.
Iniziamo con un semplice script che stampi la data odierna:

```
<HTML>
<HEAD>
<TITLE>Esercizio 1</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE=vbscript>
<!--
document.write (date())
//-->
</SCRIPT>
```

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

Dopo aver salvato e aperto questa pagina con il browser, apparirà a video la data di oggi nel formato gg/mm/aaaa. Immaginiamo di volerla far apparire in un formato esteso: ad esempio, mercoledi 5 settembre 2001. Abbiamo a disposizione due modi per farlo: il primo è con la funzione FormatDateTime

```
<SCRIPT LANGUAGE=vbscript>
<!--
document.write (FormatDateTime (date(),1))
//-->
</SCRIPT>
```

La soluzione è perfetta, però può essere **migliorata**. La data che appare è quella del computer dell'utente che visualizza la pagina: se infatti provi a cambiare la data del tuo computer e ricarichi la pagina, la data visualizzata non sarà più quella di oggi; inoltre il formato e la lingua con cui è scritta dipendono sempre dalla impostazioni internazionali del client. Prova nel pannello di controllo a mettere le impostazioni internazionali in inglese e leggerai: Wednesday, September 05, 2001.
Se il contenuto della pagina è in italiano, ha senso visualizzare la data solo in italiano, e sarà appunto questo il nostro obbiettivo.
Per prima cosa si caricano dei vettori con i giorni della settimana e i mesi:

```
<SCRIPT LANGUAGE=vbscript>
<!--
dim giorniset(7)
giorniset(1)="Domenica"
giorniset(2)="Luned&igrave;"
giorniset(3)="Marted&igrave;"
giorniset(4)="Mercoled&igrave;"
giorniset(5)="Gioved&igrave;"
giorniset(6)="Venerd&igrave;"
giorniset(7)="Sabato"

dim mesi(12)
mesi(1)="Gennaio"
mesi(2)="Febbraio"
mesi(3)="Marzo"
mesi(4)="Aprile"
mesi(5)="Maggio"
mesi(6)="Giugno"
mesi(7)="Luglio"
mesi(8)="Agosto"
mesi(9)="Settembre"
mesi(10)="Ottobre"
mesi(11)="Novembre"
mesi(12)="Dicembre"
//-->
</SCRIPT>
```

Attraverso le funzioni WeekDay e Month è possibile avere un numero corrispondente al giorno della settimana e uno al mese corrente. Chiaramente questi numeri corrispondono agli indici dei vettori; in questo modo, se oggi è mercoledì 5 settembre 2001, la funzione WeekDay restituisce 4 e Month restituisce 9.
Ora immaginiamo di volere la data col giorno di 2 cifre e l'anno di 4 cifre.
Se il giorno è minore di 10, si aggiunge uno 0 davanti al numero corrispondente al giorno della data odierna. La funzione che restituisce il giorno dalla data è Day

```
if Day(date()) < 10 then
giorno="0" & Day(date()
else
giorno=Day(date())
end if
```

Per l'anno, invece, andiamo a controllare il numero di cifre che lo compongono. La funzione che restituisce l'anno è Year, mentre con la funzione Len si conoscono il numero di caratteri.

```
if Len(Year(date()) ) = 2 then
anno="20" & Year(date())
elseif Len(Year(date()) ) = 3then
anno="2" & Year(date())
else
anno=Year(date())
end if
```

Non ci resta che combinare queste parti di codice:

```
<html>
<head>
<title>esercizio 1</title>
</head>
<body>
<script language=vbscript>
<!--
```

```
dim giorniset(7)
giorniset(1)="domenica"
giorniset(2)="luned&igrave;"
giorniset(3)="marted&igrave;"
giorniset(4)="mercoled&igrave;"
giorniset(5)="gioved&igrave;"
giorniset(6)="venerd&igrave;"
giorniset(7)="sabato"

dim mesi(12)
mesi(1)="gennaio"
mesi(2)="febbraio"
mesi(3)="marzo"
mesi(4)="aprile"
mesi(5)="maggio"
mesi(6)="giugno"
mesi(7)="luglio"
mesi(8)="agosto"
mesi(9)="settembre"
mesi(10)="ottobre"
mesi(11)="novembre"
mesi(12)="dicembre"

if day(date()) < 10 then
giorno="0" & day(date())
else
giorno=day(date())
end if

if len(year(date()) ) = 2 then
anno="20" & year(date())
elseif len(year(date()) ) = 3then
anno="2" & year(date())
else
anno=year(date())
end if

giornosettimana=giorniset(weekday(date()))
mese=mesi(month(date()))

document.write (giornosettimana & " " & giorno & " " & mese
& " " & anno) //-->
</script>
</body>
</html>
```

Il risultato sarà sempre **Mercoledì 05 Settembre 2001**, indipendentemente dalle impostazioni internazionali del client

21. Confronti fra date

Lavorando con degli scadenziari, può capitare di voler visualizzare le voci di colori diverso in funzione alla data odierna; nell'esempio che adesso andremo a vedere, faremo in modo che le attività scadute siano in rosso, quelle future in verde e quelle odierne in giallo.
Se per fare il confronto tra date si facesse un operazione del tipo

```
<SCRIPT LANGUAGE=vbscript>
if "5/11/2001"<"20/11/2001" then
document.write "Giusto"
else
document.write "Sbagliato"
end if
</SCRIPT>
```

ci si aspetterebbe che il risultato fosse "Giusto", invece è errato. Infatti il confronto non viene effettuato tra date, ma tra strighe: in ordine alfabetico il 5 viene dopo il 2, quindi il risultato è sbagliato.
L'accorgimento da adottare quando si vogliono fare confronti tra date, è quello di specificare che i parametri del confronto sono date. Per farlo si usa CDate

```
<SCRIPT LANGUAGE=vbscript>
if CDate("5/11/2001")<CDate("20/11/2001") then
document.write "Giusto"
else
document.write "Sbagliato"
end if
</SCRIPT>
```

Adesso il risultato è giusto.
Il modo migliore per fare confronti fra date, è tramite la funzione DateDiff: questa funzione restituisce 0 se le date sono uguali, un numero negativo se è maggiore la prima data e un numero positivo se è maggiore la seconda data.

```
<SCRIPT LANGUAGE=vbscript>
if DateDiff("d","5/11/2001","20/11/2001")>0
then
document.write "Giusto"
else
document.write "Sbagliato"
end if
</SCRIPT>
```

Questa funzione restituisce "Giusto".
Tornando al nostro esercizio, useremo DateDiff per confrontare delle date con quella odierna. Il risultato del confronto colorerà in modo diverso lo sfondo di una cella.

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE=vbscript>
if DateDiff("d","5/11/2001","20/11/2001")>0 then
document.write "Giusto"
else
document.write "Sbagliato"
end if
</SCRIPT>
<TABLE width=75% border=1 cellspacing=1 cellpadding=1>
<TR>
<TD id=tab1><input name=data1></TD>
</TR>
<TR>
<TD id=tab2><input name=data2></TD>
</TR>
<TR>
<TD id=tab3><input name=data3></TD>
</TR>
</TABLE>
<input type=button value=Confronta >
<SCRIPT LANGUAGE=vbscript >
<!--
data1.value = DateAdd("d",-5,date())
data2.value = date()
data3.value = DateAdd("d",+6,date())
sub colora()
if DateDiff("d",data1.value,date())>0 then
tab1.bgcolor ="red"
elseif DateDiff("d",data1.value,date())<0 then
tab1.bgcolor ="green"
else
tab1.bgcolor ="yellow"
end if
if DateDiff("d",data2.value,date())>0 then
tab2.bgcolor ="red"
elseif DateDiff("d",data2.value,date())<0 then
tab2.bgcolor ="green"
else
tab2.bgcolor ="yellow"
end if
if DateDiff("d",data3.value,date())>0 then
tab3.bgcolor ="red"
elseif DateDiff("d",data3.value,date())<0 then
tab3.bgcolor ="green"
else
tab3.bgcolor ="yellow"
end if
end sub
//-->
</SCRIPT>
</BODY>
</HTML>
```

22. Un piccolo calendario

Può essere utile inserire in una pagina un calendario che, legato ad altre funzioni VBScript, potrà essere esteso per applicazioni come un'agenda o una bolla cartolina via WEB ecc.
Partiamo con il definire che giorno della settimana è il primo del mese corrente. Per farlo basta usare:
primodelmese="01/" & month(date()) & "/" & year(date())
giornodellaset=WeekDay(primodelmese)
L'obbiettivo è quello di avere una tabella di una riga e 7 colonne: se il primo del mese è di Domenica, il giorno 1 sarà nella prima cella, se è di Lunedì, nella seconda, e così a seguire.
Per farlo, si costruisce un ciclo che genera celle vuote finchè non stampa i giorni in corrispondenza al giorno della settimana cui corrispondono.

```
<HTML>
<HEAD> <TITLE>Esercizio 2</TITLE>
</HEAD>
<BODY>
<table border=1>
<tr>
<SCRIPT LANGUAGE=vbscript>
<!--
primodelmese="01/" & month(date()) & "/" & year(date())
giornodellaset=WeekDay(primodelmese)
for i=1 to giornodellaset-1
document.write ("<td> </td>")
next
giorno=1
for i=giornodellaset to 7
document.write ("<td>" & giorno & "</td>")
giorno=giorno+1
next
//-->
</SCRIPT>
</tr>
</table>
</BODY>
</HTML>
```

Passiamo alle righe successive. La variabile giorno indica il giorno a cui siamo arrivati, non ci resta che compiere lo stesso ciclo in una nuova riga.

```
<HTML>
<HEAD> <TITLE>Esercizio 2</TITLE>
</HEAD>
<BODY>
<table border=1>
<tr>
<SCRIPT LANGUAGE=vbscript>
<!--
primodelmese="01/" & month(date()) & "/" & year(date())
giornodellaset=WeekDay(primodelmese)
for i=1 to giornodellaset-1
document.write ("<td> </td>")
next
giorno=1
for i=giornodellaset to 7
document.write ("<td>" & giorno & "</td>")
giorno=giorno+1
next
//-->
</SCRIPT>
</tr>
</table>
</BODY>
</HTML>
```

Prima di continuare, proviamo a generalizzare il codice, che in fondo non è altro che la ripetizione delle stesse righe per 4 volte:

```
<HTML>
<HEAD>
<TITLE>Esercizio 2</TITLE>
</HEAD>
<BODY>

<SCRIPT LANGUAGE=vbscript>
<!--
document.write ("<table border=1>")
giornodelmese="01/" & month(date()) & "/" & year(date())
giorno=1
for ciclo=1 to 4
document.write ("<tr>")
```

```
giornodellaset=WeekDay(giornodelmese)
for i=1 to giornodellaset-1
document.write ("<td> </td>")
next

for i=giornodellaset to 7
document.write ("<td>" & giorno & "</td>")
giorno=giorno+1
next
document.write ("</tr>")
giornodelmese=(giorno) & "/" & month(date()) & "/" & year(date())
next
document.write ("</table>")
//-->
</SCRIPT>
</BODY>
</HTML>
```

Quante volte deve essere ripetuto il ciclo? Dipende da quanti sono i giorni del mese e da che giorno è il primo del mese. Pensandoci un po', si va dai 4 ai 6 cicli. Per determinare la fine del mese, si può usare questo espediente: dal primo del mese successivo si toglie un giorno

```
primodelmesesuccessivo="01" & "/" & (month(date())+1) & "/" & Year(date())
ultimodelmese=DateAdd("d",-1,primodelmesesuccessivo)
```

Ora non ci resta che continuare il ciclo finchè la variabile giorno è minore dell' ultimo del mese:

```
<HTML>
<HEAD>
<TITLE>Esercizio 2</TITLE>
</HEAD>
<BODY>

<script type="text/vbscript">
<!--
document.write ("<table border=1>")
giornodelmese="01/" & month(date()) & "/" & year(date())
giorno=1
primodelmesesuccessivo="01" & "/" & (month(date())+1) & "/" & Year(date())
ultimodelmese=DateAdd("d",-1,primodelmesesuccessivo)
While (giorno<day(ultimodelmese))
document.write ("<tr>")
giornodellaset=WeekDay(giornodelmese)
for i=1 to giornodellaset-1
document.write ("<td> </td>")
next

for i=giornodellaset to 7
document.write ("<td>" & giorno & "</td>")
giorno=giorno+1
next
document.write ("</tr>")
giornodelmese=(giorno) & "/" & month(date()) & "/" & year(date())
Wend
document.write ("</table>")
//-->
</SCRIPT>
</BODY>
</HTML>

<script type="text/vbscript">
<!--
document.write ("<table class="tabella" border=1>")
giornodelmese="01/" & month(date()) & "/" & year(date())
giorno=1
primodelmesesuccessivo="01" & "/" & (month(date())+1) & "/" & Year(date())
ultimodelmese=DateAdd("d",-1,primodelmesesuccessivo)
While (giorno<day(ultimodelmese))
document.write ("<tr>")
giornodellaset=WeekDay(giornodelmese)
for i=1 to giornodellaset-1
document.write ("<td> </td>")
next

for i=giornodellaset to 7
document.write ("<td>" & giorno & "</td>")
giorno=giorno+1
next
document.write ("</tr>")
```

```
giornodelmese=(giorno) & "/" & month(date()) & "/" & year(date())
Wend
document.write ("</table>")
//-->
</script>
```

Non ci resta che bloccare la scrittura dei giorni alla fine del mese ed abbellire il tutto:

```
<script type="text/vbscript">
<!--
document.write ("<table border=1>")
document.write ("<tr><td align=center colspan=7>" &
Ucase(monthname(month(date()))) & " " & year(date()) & "</td></tr>")
document.write
("<tr><td>D</td><td>L</td><td>M</td><td>M</td><td>G</td><td>V</td><td>S<
;/td></tr>") giornodelmese="01/" & month(date()) & "/" & year(date())
giorno=1
'questo controllo risolve il probelma del mese di dicembre
if (month(date())+1)>12 then
anno=Year(date())+1
mese=1
else
anno=Year(date())
mese=(month(date())+1)
end if
primodelmesesuccessivo="01" & "/" & mese & "/" & anno
ultimodelmese=DateAdd("d",-1,primodelmesesuccessivo)
While (giorno<day(ultimodelmese))
document.write ("<tr>")
giornodellaset=WeekDay(giornodelmese)
for i=1 to giornodellaset-1
document.write ("<td> </td>")
next


for i=giornodellaset to 7
document.write ("<td>" & giorno & "</td>")
giorno=giorno+1
if giorno>day(ultimodelmese) then exit for
next
for i=i+1 to 7
document.write ("<td> </td>")
next
document.write ("</tr>")
giornodelmese=(giorno) & "/" & month(date()) & "/" & year(date())
Wend
document.write ("</table>")
//-->
</SCRIPT>
```

Per chi ama le nuove tecnologie, in **ASP.NET** un calendario simile a quello appena creato si ottiene richiamando un oggetto!

23. Il gioco dell'impiccato

L'interfaccia si compone di un **campo password** in cui inserire la parola da cercare, un campo testo in cui inserire la lettera, un campo testo in cui appare l'andamento del gioco (lettere indovinate e asterischi) e un campo testo che memorizza gli errori.

Parola da indovinare

Lettera Proposta

Soluzione Parziale

Errori

Premendo il pulsante start, tramite l'evento **OnClick** attiviamo la funzione iniziogioco() che pone a zero il campo errori e maschera la soluzione parziale. Per compiere quest'ultima operazione, si utilizzano le funzioni string e leng: string genera un numero di asterischi uguale alla lunghezza (len) della parola da indovinare.

```
<SCRIPT LANGUAGE=vbscript>
<!--
sub iniziogioco()
errori.value = 0
Soluzione.value= string(len(parola.value),"*")
end sub
//-->
</SCRIPT> Parola da indovinare
<input type="password" name="parola">
```

```
<input type="button" name="Start" value="Start" >
<br>
Lettera Proposta
<input type="text" name="lettera" size="1" maxlength="1">
<input type="button" name="prova" value="Prova">
<br>
Soluzione Parziale
<input type="text" name="Soluzione">
<br>
Errori
<input type="text" name="errori" size="3">
```

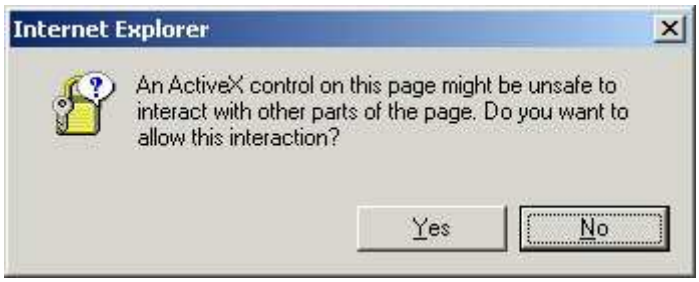
Alla pressione del pulsante prova, si cerca la lettera proposta tra le lettere della parola; nel caso si trovi, essa viene sostituita. Per rendere il gioco completo, viene visualizzato un messaggio nel caso la soluzione non contenga più asterischi.

```
<HTML>
<HEAD>
<TITLE>Esercizio 3</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE=vbscript>
<!--
sub iniziogioco()
errori.value = 0
Soluzione.value= string(len(parola.value),"*")
end sub

sub prova()

sol="
err=1
for x=1 to len(parola.value )
carattere=mid(parola.value,x,1)
if carattere=lettera.value then
sol=sol & lettera.value
err=0
else
sol=sol & mid(soluzione.value,x,1)
end if
next
errori.value =cint(errori.value) + err
Soluzione.value =sol
lettera.value ="
lettera.focus()
if instr(1,Soluzione.value,"*")=0 then
MsgBox "Hai Vinto"
end if
end sub
//-->
</SCRIPT>
Parola da indovinare
<input type="password" name="parola">
<input type="button" name="Start" value="Start" >
<br>
Lettera Proposta
<input type="text" name="lettera" size="1" maxlength="1">
<input type="button" name="prova" value="Prova" >
<br>
Soluzione Parziale
<input type="text" name="Soluzione">
<br>
Errori
<input type="text" name="errori" size="3">
</p>
</BODY>
</HTML>
```

Il tutto si basa sulle funzioni **InStr**, che restituisce la posizione di un'occorrenza all'interno di una parola, e **Mid**, che estrae una sottostringa da una frase.

24. Il FileSystem Object

Grazie all'oggetto FileSystem, attraverso VBScript è possibile conoscere informazioni sui dischi, sulle cartelle e sui file di un utente.
Preoccupante, vero? Per fortuna l'utente, prima di fornire queste informazioni, viene avvertito con un messaggio come questo:



Se provi a cliccare su questo link , ti apparirà un messaggio come quello in figura; **non preoccuparti**, la pagina che vedrai ti darà solo informazioni sul tuo disco rigido!
Come per un qualsiasi oggetto, per istanziare l'oggetto VBScript occorre una sintassi come la seguente:
Set fso = CreateObject("Scripting.FileSystemObject")
Ora l'oggetto fso avrà delle proprietà e dei metodi che gli permetteranno di settare o leggere informazioni riguardanti il File System.
Per ottenere un risultato come quello dell'esempio precedente, si è utilizzato il seguente codice:

```
<SCRIPT LANGUAGE=vbscript>
<!--
Dim drive
drive="C:"
Set fso = CreateObject("Scripting.FileSystemObject")
document.write "Il tuo disco C ha"
& FormatNumber (fso.GetDrive(drive).TotalSize,0)
& " bytes"
//-->
</SCRIPT>
```

Come si vede dall'oggetto FileSystem fso, è stato richiamato l'oggetto drive con la sintassi **fso.GetDrive(drive)**. Da questo oggetto è stata visualizzata la proprietà TotalSize che rappresenta la dimensione del drive.
Dall'oggetto FileSystem è possibile richiamare altri oggetti come l'oggetto Cartella, l'oggetto File e l'Oggetto TextStream.
Vediamo come:
Instanza dell'oggetto FileSystem:
Set fso = CreateObject("Scripting.FileSystemObject")
Instanza dell'oggetto Drive:
Set oggDrive = fso.GetDrive("C:")
Instanza dell'oggetto Cartella:
Set oggCartella = fso.GetFolder("c:winnt")
Instanza dell'oggetto File
Set oggFile = fso.GetFile("c:autoexec.bat")
Instanza dell'oggetto TextStream
Set OggTextStream = oggFile.OpenAsTextStream(1)
Creati gli oggetti, ora diventa facile poterne sfruttare le potenzialità conoscendo metodi e proprietàche vengono descritti nella guida a VBScript
Nei prossimi capitoli vedremo alcuni esempi di utilizzo degli oggetti legati al **FileSystem**

25. Sfogliando l'Hard-disk

Lo scopo di questa applicazione è quello di visualizzare il contenuto dell'HardDisk, come fa esplora risorse. Il primo passo è quello di visualizzare le cartelle e i file contenute nella root:

```
<SCRIPT LANGUAGE=vbscript>
<!--
Dim Drive,folder, subfolder,elemFolder
path="C:"
Set fso = CreateObject("Scripting.FileSystemObject")
Set folder = fso.GetFolder(path)
Set subfolder = folder.Subfolders
document.write path & "<br>"
For each elemFolder in subfolder
document.write " <b>" & replace(lcase(elemFolder),lcase(path),"") &
"</b><br>" next
Set files = folder.files
For each elemFile in files
document.write " " & replace(lcase(elemFile),lcase(path),"") &
"<br>" next
//-->
</SCRIPT>
```

Nella variabile path è inserito il nome della cartella da esaminare, in questo caso abbiamo scelto la root del disco C.
Vengono creati gli **oggetti** folder e subfolder: quest'ultimo conterrà tutte le cartelle contenute nella cartella folder. Per visualizzare le varie sottocartelle si sfrutta un ciclo **For Each**. Il path di ogni sottocartella, attraverso il comando replace, viene liberato della parte contenente il nome della cartella superiore.
Lo stesso principio è adottato per leggere i file contenuti nel path. Per distinguere i file dalle cartelle, queste ultime sono in grassetto.

Risultato dello script (riferito, naturalmente, al mio hard disk):
C:
documents and settings
inetpub
program files
recycler
winnt
autoexec.bat
boot.bak
boot.ini
config.sys
io.sys
msdos.sys
ntdetect.com
ntldr
pagefile.sys

Lo script può essere inserito in una funzione che riceve come parametro il path. Ora non resta che rendere i nomi delle cartelle **cliccabili**. Cliccando sul nome della cartella, il path della cartella viene passato alla funzione la quale ridisegna l'albero con tutte le sottocartelle e i file.
Questo script usa alcuni comandi come **innerHTML** e il tag **span** non propri del VBScript, ma che aumentano le funzionalità di questo linguaggio.

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>
<BODY>

<SCRIPT LANGUAGE=vbscript>
<!--
function naviga(path)
Dim Drive,folder, subfolder,elemFolder
Set fso = CreateObject("Scripting.FileSystemObject")
Set folder = fso.GetFolder(path)
Set subfolder = folder.Subfolders
carfiles.innerHTML = path & "<br>"
For each elemFolder in subfolder
nomefolder=replace(lcase(elemFolder),lcase(path),"")
carfiles.innerHTML = carfiles.innerHTML & " <b><span
style=""cursor:hand"" " & replace(elemFolder,"","") & "')"">" & nomefolder & "</span></b><br>"
next
Set files = folder.files
For each elemFile in files
carfiles.innerHTML = carfiles.innerHTML & " " &
replace(lcase(elemFile),lcase(path),"") & "<br>" next
end function
//-->
</SCRIPT>

<div name=carfiles id=carfiles><span style="cursor:hand"
C:')">C:</span></div>
</BODY>
</HTML>
```

L'unica cosa che manca per rendere l'albero completamente navigabile, è la possibilità di navigare all'indietro. Potrebbe essere interessante provare ad aggiungere questa nuova funzionalità

26. Gestione dei file

Per poter operare con questi esempi si dovrà creare una cartella **prova** nel drive C:
Iniziamo con il copiare il file **autoexec.bat** da C: a C:prova
Per farlo occorre il semplice script:

```
<SCRIPT LANGUAGE=vbscript>
<!--
Dim Drive, OggFile, origine, destinazione
origine="c:autoexec.bat"
destinazione="c:prova"
Set fso = CreateObject("Scripting.FileSystemObject")
set OggFile = fso.GetFile (origine)
OggFile.Copy destinazione,true
document.write "Il file " & origine & " è stato
copiato in " & destinazione & "."
```

```
//-->
</SCRIPT>
```
Associamo questo script ad una funzione richiamata dal click di un pulsante:
```
<HTML>
<HEAD>
<TITLE>Esercizio</TITLE>
</HEAD>
<BODY>

<SCRIPT LANGUAGE=vbscript>
<!--
function copia()
Dim Drive, OggFile, origine, destinazione
origine="c:autoexec.bat"
destinazione="c:prova"
Set fso = CreateObject("Scripting.FileSystemObject")
set OggFile = fso.GetFile (origine)
OggFile.Copy destinazione,true
msgbox "Il file " & origine & " è stato copiato in " & destinazione & "."
end function
//-->
</SCRIPT>
<input type=button value=Copia >
</BODY>
</HTML>
```

Se premiamo più volte il pulsante copia, ogni volta sovrascriviamo il file nella cartella prova.
Il nostro obbiettivo sarà ora quello di verificare se il file **esiste già**: solo nel caso questo manchi, verrà copiato nella cartella prova. Per fare ciò, si usa il metodo **FileExists**. Questo metodo restituisce true se il file esiste, false se non esiste.

```
<HTML>
<HEAD>
<TITLE>Esercizio</TITLE>
</HEAD>
<BODY>

<SCRIPT LANGUAGE=vbscript>
<!--
function copia()
Dim Drive, OggFile, origine, destinazione
origine="c:autoexec.bat"
destinazione="c:prova"
Set fso = CreateObject("Scripting.FileSystemObject")
set OggFile = fso.GetFile (origine)
if fso.FileExists (destinazione & "autoexec.bat") then
MsgBox "Esiste già un file autioexec.bat nella cartella prova."
else
OggFile.Copy destinazione,true
MsgBox "Il file " & origine & " è stato copiato in " & destinazione &
"." end if
end function
//-->
</SCRIPT>
<input type=button value=Copia >
</BODY>
</HTML>
```

Completiamo l'opera con una funzione per cancellare il file dalla cartella prova:

```
function cancella()
Dim OggFile, origine
origine="c:provaautoexec.bat"
Set fso = CreateObject("Scripting.FileSystemObject")
Set OggFile = fso.GetFile (origine)
OggFile.Delete
MsgBox "Il file " & origine & " è stato cancellato."
end function
```
Anche a questa funzione associamo un pulsante. Qui di seguito è indicato il codice completo dell'applicazione:

```
<HTML>
<HEAD>
<TITLE>Esercizio</TITLE>
</HEAD>
<BODY>

<SCRIPT LANGUAGE=vbscript>
<!--
```

```
function copia()
Dim Drive, OggFile, origine, destinazione
origine="c:autoexec.bat"
destinazione="c:prova"
Set fso = CreateObject("Scripting.FileSystemObject")
set OggFile = fso.GetFile (origine)
if fso.FileExists (destinazione & "autoexec.bat") then
MsgBox "Esiste già un file autioexec.bat nella cartella prova."
else
OggFile.Copy destinazione,true
MsgBox "Il file " & origine & " è stato copiato in " & destinazione &
"." end if
end function

function cancella()
Dim OggFile, origine
origine="c:provaautoexec.bat"
Set fso = CreateObject("Scripting.FileSystemObject")
Set OggFile = fso.GetFile (origine)
OggFile.Delete
MsgBox "Il file " & origine & " è stato cancellato."
end function
//-->
</SCRIPT>
<input type=button value=Copia >
<input type=button value=Cancella >
</BODY>
</HTML>
```

Le operazioni più frequenti che si operano sui file sono quelle appena viste: esistenza, copia, cancellazione e spostamento. Lo spostamento di un file non è stato menzionato tra gli esempi, ma è uguale alla copia : solo che al posto del metodo Copy si usa il metodo Move

27. Gestione delle cartelle

Soprattutto nelle applicazioni **server Side**, spesso capita di dover lavorare con cartelle e file. Nella precedente lezione abbiamo visto come usare i file, ora passiamo alle **cartelle**.
Prima di tutto creiamo una cartella con il metodo **CreateFolder**:
Set fso = CreateObject("Scripting.FileSystemObject") fso.CreateFolder ("C:prova_creazione_cartella") MsgBox "La cartella è stata creata."
Prima di creare una cartella, sarebbe utile vedere se esiste già. Per farlo si usa il metodo FolderExists, che restituisce true se la cartella esiste, false se non esiste:
Set fso = CreateObject("Scripting.FileSystemObject") cartella="C:prova_creazione_cartella"
if fso.FolderExists (cartella) then
MsgBox "La cartella " & cartella & " esiste già."
else
fso.CreateFolder (cartella) MsgBox "La cartella è stata creata." end if
Come per l'esercizio precedente, associamo a questo codice una funzione richiamata da un pulsante:

```
<HTML>
<HEAD>
<TITLE>Esercizio</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE=vbscript >
<!--
function crea()
Set fso = CreateObject("Scripting.FileSystemObject")
cartella="C:prova_creazione_cartella"
if fso.FolderExists (cartella) then
MsgBox "La cartella " & cartella & " esiste già."
else
fso.CreateFolder (cartella)
MsgBox "La cartella è stata creata."
end if
end function
//-->
</script>
<input type=button value=Crea >
</BODY>
</HTML>
```

Il passo successivo è cancellare la cartella attraverso il metodo **DeleteFolder**:
Set fso = CreateObject("Scripting.FileSystemObject") cartella="C:prova_creazione_cartella" fso.DeleteFolder(cartella)

Anche in questo caso, prima di cancellare la cartella verifichiamo se esiste. All'operazione di cancellazione, associamo una funzione e un pulsante.

```
<HTML>
<HEAD>
<TITLE>Esercizio</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE=vbscript >
<!--
function crea()
Set fso = CreateObject("Scripting.FileSystemObject")
cartella="C:prova_creazione_cartella"
if fso.FolderExists (cartella) then
MsgBox "La cartella " & cartella & " esiste già."
else
fso.CreateFolder (cartella)
MsgBox "La cartella è stata creata."
end if
end function
function cancella()
Set fso = CreateObject("Scripting.FileSystemObject")
cartella="C:prova_creazione_cartella"
if fso.FolderExists (cartella) then
fso.DeleteFolder(cartella)
MsgBox "La cartella è stata cancellata."
else
MsgBox "La cartella non esiste."
end if
end function
//-->
</script>
<input type=button value=Crea >
<input type=button value=Cancella >
</BODY>
</HTML>
```

Infine, con il metodo **CopyFolder**, copiamo la cartella di prova in se stessa. Anche in questo caso, verifichiamo che la cartella esista prima della copia e creiamo un pulsante apposito per questa operazione.

```
<HTML>
<HEAD>
<TITLE>Esercizio</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE=vbscript >
<!--
function crea()
'Set fso = CreateObject("Scripting.FileSystemObject")
cartella="C:prova_creazione_cartella"
if fso.FolderExists (cartella) then
MsgBox "La cartella " & cartella & " esiste già."
else
fso.CreateFolder (cartella)
MsgBox "La cartella è stata creata."
end if
end function
function cancella()
'Set fso = CreateObject("Scripting.FileSystemObject")
cartella="C:prova_creazione_cartella"
if fso.FolderExists (cartella) then
fso.DeleteFolder(cartella)
MsgBox "La cartella è stata cancellata."
else
MsgBox "La cartella non esiste."
end if
end function
function copia()
Set fso = CreateObject("Scripting.FileSystemObject")
cartella="C:prova_creazione_cartella"
if fso.FolderExists (cartella) then
fso.CopyFolder (cartella,cartella & "copia",1)
MsgBox "La cartella è stata copiata."
else
MsgBox "La cartella non esiste."
end if
end function
```

```
//-->
</script>
<input type=button value=Crea >
<input type=button value=Cancella >
<input type=button value=Copia >
</BODY>
</HTML>
```

Per quanto i comandi appena visti siano quasi inutili in pagine HTML, sono molto utilizzati in applicazioni lato server, in particolare quando si programma in ASP

28. Oggetto TextStream

VBScript ci mette a disposizione l'oggetto **TextStream**, grazie al quale è possibile creare, leggere e scrivere su un file di testo. Le applicazioni legate a questo oggetto sono innumerevoli; ad esempio, si può realizzare un contatore: ad ogni accesso alla pagina si legge il valore nel file di testo, e lo si riscrive incrementato di un'unità. Oppure si può creare un file per le statistiche, in cui memorizzare la data e l'ora di accesso ad una pagina.
Iniziamo con il realizzare un **contatore**:

```
<SCRIPT LANGUAGE=vbscript>
<!--
Set fso = CreateObject("Scripting.FileSystemObject")
contatore="c:contatore.txt"
if not fso.FileExists (contatore)then
fso.CreateTextFile (contatore)
Set OggFile = fso.GetFile (contatore)
Set OggTextStream = OggFile.OpenAsTextStream(2)
OggTextStream.WriteLine "0"
OggTextStream.Close
end if
Set OggFile = fso.GetFile (contatore)

Set OggTextStream1 = OggFile.OpenAsTextStream(1)
valore=OggTextStream1.ReadLine
OggTextStream1.Close

Set OggTextStream = OggFile.OpenAsTextStream(2)
OggTextStream.WriteLine cstr(cint(valore)+1)
OggTextStream.Close
//-->
</SCRIPT>
```

Lo script è composto da una prima parte che si occupa di **verificare** se esiste il file di testo; se non esiste, lo crea e gli inserisce dentro il valore 0.
Il passo successivo è quello di **leggere** il valore contenuto nel file di testo, che poi viene incrementato e scritto nel file di testo. Si noti la differenza tra l'accesso al file in scrittura ed in lettura: in lettura si ha **OpenAsTextStream(1)**, mentre in scrittura **OpenAsTextStream(2)**.
L'altro modo per aprire il file di testo è in accodamento, usando OpenAsTextStream(8).
La scrittura in accodamento è proprio quella che ci occorre per realizzare un file di log.

```
<SCRIPT LANGUAGE=vbscript>
<!--
Set fso = CreateObject("Scripting.FileSystemObject")
logfile="c:logfile.txt"
if not fso.FileExists (logfile)then
fso.CreateTextFile (logfile)
end if
Set OggFile = fso.GetFile (logfile)

Set OggTextStream = OggFile.OpenAsTextStream(8)
OggTextStream.WriteLine now()
OggTextStream.Close

Set OggTextStream1 = OggFile.OpenAsTextStream(1)
Do While Not OggTextStream1.AtEndOfStream
stringa = OggTextStream1.ReadLine
document.write stringa & "<br>"
loop
OggTextStream1.Close
//-->
</SCRIPT>
```

Anche in questa applicazione viene controllata l'esistenza del file di log e, se necessario, viene creato. Il file viene poi aperto in accodamento e gli viene scritta la data e l'ora dell'accesso.
Per dimostrarne il funzionamento, il file viene poi aperto e con un ciclo **Do While-Loop** vengono visualizzati tutti gli accessi.

Come l'oggetto FileSystem, anche l'oggetto TextStream è utilizzato per lo più lato server. Con i due esercizi appena proposti, abbiamo infatti creato un contatore di accessi come se ne vedono tanti nei siti internet, ed un file di statistiche.
Non solo i file di testo possono essere utilizzati come file di configurazione, oppure, se si è capaci, possono essere usati come database per piccole applicazioni. Alcuni forum sfruttano 2 file di testo per funzionare, uno per gli utenti ed uno per i messaggi. Potrebbe essere un ottimo esercizio per verificare le conoscenze di VBScript apprese finora.

Strumenti di interattività

29. Comunicare con l'utente

Comunicando con l'utente scateniamo la sua curiosità e quindi la voglia di navigare all'interno della nostra pagina o sito. A tutti è capitato di collegarsi ad un sito e di fermarsi alla home page annoiati, oppure di restare affascinati e di navigarci a lungo. Questo evidenzia come sia importante la comunicazione con l'utente per chi crea pagine web.
VBScript ci mette a disposizione vari strumenti.
Il primo è la funzione **document.write**, molto semplice come sintassi:
document.write "Sono le " & time()
Risultato:
Sono le 17.29.07
Le varie stringhe, funzioni o variabili sono concatenate dal simbolo &.
Per comunicare con l'utente possiamo anche usare il metodo innerHTML applicato ad un oggetto.
Ad esempio creiamo un area chiamata foglio:
Testo di Default

ora aggiungiamo due pulsanti

Prova a vedere cosa capita premendo i tasti:
Testo di Default
Nota come non si ricarichi la pagina.
Se non bastasse, possiamo intrattenere il nostro utente con una finestra di Input o una finestra di messaggio un po' più complessa del semplice window.alert
Inputbox("Inserire del testo versione 4","Titolo Applicazione","testo di default",0,0)
MsgBox ("Fai una scelta",vbAbortRetryIgnore+vbInformation,"Titolo Applicazione")
Maggiori dettagli sulle funzioni InputBox e MsgBox puoi trovarli sulla guida VBScript
Vedremo applicati nelle prossime lezioni questi strumenti di comunicazione con l'utente.

30. Comunicare per mezzo di finestre

Creiamo una semplice applicazione che richieda 3 numeri all'utente e ne faccia la somma.
Per ricevere in input dei dati dall'utente, usiamo la funzione InputBox. Questa funzione restituisce il valore inserito dall'utente.
Il codice delle tre finestre di input sarà
num1=InputBox("Inserisci il primo numero.","Somma di tre numeri")
num2=InputBox("Inserisci il secondo numero.","Somma di tre numeri")
num3=InputBox("Inserisci il terzo numero.","Somma di tre numeri")
Per sommare i tre valori, attenzione a non commettere l'errore di scrivere:
totale=num1+num2+num3
Infatti il valore restituito dall'InputBox è di sottotipo stringa, quindi nella variabile totale apparirà la concatenazione delle tre variabili e non la somma.
L'istruzione corretta è:
totale=cint(num1)+cint(num2)+cint(num3)
Non resta che mostrare il risultato all'utente:
window.alert(totale)
Il codice completo è:

```
<SCRIPT LANGUAGE=vbscript>
num1=InputBox("Inserisci il primo numero.","Somma di tre numeri")
num2=InputBox("Inserisci il secondo numero.","Somma di tre numeri")
num3=InputBox("Inserisci il terzo numero.","Somma di tre numeri")
totale=cint(num1)+cint(num2)+cint(num3) window.alert (totale)
</SCRIPT>
```

Per una prova clicca sul pulsante

31. Le MsgBox

La finestra MsgBox può essere usata come semplice output per mostrare un risultato oppure può essere usata per ricevere degli input.
Iniziamo dal suo aspetto: può apparire con i pulsanti SI/NO, oppure OK/CANCEL ecc..
Ecco i vari tipi con un esempio:
Possiamo poi abbellire le nostre finestre con vari loghi:
Come si diceva, le MsgBox sono anche finestre di input; infatti la funzione restituisce un valore diverso a seconda del pulsante che si preme:

```
<SCRIPT LANGUAGE=vbscript >
puls=MsgBox("Fai una scelta",vbYesNo+vbQuestion)
if puls=vbYes then
MsgBox "Hai scelto SI",vbInformation
else
MsgBox "Hai scelto NO",vbInformation
```

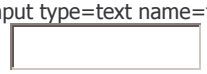
```
end if<
</SCRIPT>
```
Prova il suo funzionamento premendo il pulsante

Dal codice si può notare come per una MsgBox di input si usino le parentesi tonde puls=MsqBox (....), mentre per una finestra di solo output non si usino. Non è una scelta opzionale, metterle creerebbe un errore perchè il valore restituito dalla MsgBox non viene assegnato.
Si può ancora osservare come la scelta tra un tipo di pulsanti e l'icona si ottiene sommando le due richieste vbYesNo+vbQuestion

32. Impedire la scrittura nei TextBox

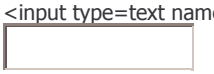
Quando si realizza un form, alcuni campi sono di tipo numerico, come il telefono o il cap.
Creiamo ora un controllo che permetta la scrittura di numeri, ma blocchi l'immissione di altri caratteri.
Per realizzare questa funzione usiamo l'evento onKeyUp, evento che parte alla pressione di un tasto.
Iniziamo con isolare l'ultimo carattere premuto dall'utente:

```
<SCRIPT LANGUAGE=vbscript >
function check()
MsgBox "Hai premuto " & Right (telefono.value,1)
end function
</SCRIPT>
<input type=text name=telefono onkeyup="check()">
```



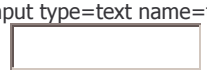
Ora controlliamo se il carattere è un numero:

```
<SCRIPT LANGUAGE=vbscript >
function check()
carattere= Right (telefono.value,1)
if carattere < "0" or carattere > "9" then
MsgBox "Hai premuto un tasto non permesso"
end if
end function
</SCRIPT>
<input type=text name=telefono onkeyup="check()">
```



Infine, se non è un numero, eliminiamo l'ultimo carattere dalla textbox:

```
<SCRIPT LANGUAGE=vbscript >
function check()
carattere= Right (telefono.value,1)
if carattere < "0" or carattere > "9" then
telefono.value =left(telefono.value,len(telefono.value)-1)
end if
end function
</SCRIPT>
<input type=text name=telefono onkeyup="check()">
```



Modificando la condizione nella funzione check, è possibile permettere o no l'inserimento di qualsiasi tipo di carattere.