

ESCOLA PROFISSIONAL DE TONDELA



Prof. Ricardo Jorge Santos

Novembro de 2004

INTRODUÇÃO AO VBSCRIPT

O Que Necessita de Saber à Partida...

Antes de continuar, deve possuir conhecimentos básicos acerca dos seguintes assuntos:

- WWW, HTML e princípios básicos de construção de uma página *Web*
-

O Que é VBScript?

- *VBScript* é uma linguagem de *script* interpretada do lado do cliente (*client side script*)
- Uma linguagem de *script* é uma linguagem de programação leve e simples
- *VBScript* é uma versão ligeira da linguagem de programação *Visual Basic* da *Microsoft*

Como Funciona?

Quando *VBScript* é inserido num documento HTML, o Internet browser irá ler o HTML e interpretar o *VBScript*. O *VBScript* pode ser executado imediatamente, ou num acontecimento posterior (recorrendo à programação de procedimentos/funções despoletados por eventos).

Onde Inserir o VBScript...

Secção Head

Os *scripts* podem ser colocados na secção *Head* da página. Normalmente é aqui que colocaremos toda a programação dos procedimentos/funções a serem chamados. Isto faz com que o *script* seja carregado antes de ser chamado do bloco principal da página (*Body*).

```
<html>
<head>
<script type="text/vbscript">
...
  instruções
...
</script>
</head>
```

Secção Body

Scripts que se encontram na secção *body* são executados enquanto a página carrega, pela ordem de sequência das instruções da própria secção.

```
<html>
<head>
</head>
<body>
<script type="text/vbscript">
...
  instruções
...
</script>
</body>
```

Podemos colocar um número ilimitado de *scripts* no documento, podendo ter *scripts* tanto na secção *body* como na secção *head*, juntos.

```
<html>
<head>
<script type="text/vbscript">
  ...
  instruções
  ...
</script>
</head>
<body>
<script type="text/vbscript">
  ...
  instruções
  ...
</script>
</body>
```

VARIÁVEIS NO VBSCRIPT

O Que é uma Variável?

Uma variável é um "contentor" de informação que se pretende guardar. O valor de uma variável pode ser modificado durante na sequência de instruções que compõem o *script*. Podemos referenciar uma variável pelo nome para consultar o seu valor ou alterá-lo. No *VBScript*, todas as variáveis são do tipo *variant*, podendo por isso armazenar diferentes tipos de dados.

Regras para os Nomes das Variáveis:

- Têm obrigatoriamente de começar por uma letra
- Não pode ser utilizado o ponto final (.)
- Não podem exceder 255 caracteres

Declaração de Variáveis

Variáveis podem ser declaradas através das instruções *Dim*, *Public* ou *Private*. Exemplo:

```
dim nome
nome = valor
```

Acabou-se de criar uma variável. O nome da variável é *nome*.

Também podem ser criadas variáveis sem as declarar com *Dim*. Exemplo:

```
nome = valor
```

Agora também foi criada uma variável. O nome da variável é *nome*.

Porém, esta forma não constitui uma boa prática de programação, pois podemos enganar no nome da variável posteriormente no *script*, o que pode causar resultados estranhos e aparentemente inexplicáveis e difíceis de detectar quando o *script* for executado. Isto acontece porque quando se escreve erradamente a variável *nome* como *nime*, por exemplo, o *script* vai automaticamente criar uma nova variável chamada *nime* com um valor inicial igual a zero. Para prevenir isto pode ser usada a instrução *Option Explicit*. Quando se usa esta instrução o *VBScript* obriga a declarar todas as variáveis com uma das instruções *Dim*, *Public* ou *Private*. A instrução *Option Explicit* deve ser colocada como primeira instrução no topo do *script*. Exemplo:

```
option explicit
dim nome
nome = valor
```

Atribuindo Valores a Variáveis

Podemos atribuir valores às variáveis da seguinte forma:

```
nome = "Hege"
i = 200
```

O nome da variável fica à esquerda da expressão e o valor que pretendemos que ela assuma fica à direita. Agora a variável *nome* tem o valor "Hege".

Tempo de Vida das Variáveis

Quando se declara uma variável dentro de uma sub-rotina, ela só pode ser acedida dentro dessa sub-rotina. Quando se sai da sub-rotina, a variável é destruída. Estas variáveis são apelidadas de variáveis locais. Podemos ter variáveis locais com o mesmo nome em sub-rotinas diferentes, uma vez que elas só são reconhecidas dentro da própria sub-rotina em que são declaradas.

Se declararmos uma variável fora de uma sub-rotina, todas as sub-rotinas poderão aceder ao seu valor. Estas variáveis são apelidadas de variáveis globais. Apenas serão destruídas quando a página que lhes deu origem for fechada.

Variáveis Vector (Array)

Por vezes queremos guardar mais do que um valor numa só variável. Nestes casos podemos criar uma variável que contenha um conjunto de valores. Este tipo de variável é chamado de vector (*array*). A declaração de uma variável *array* utiliza parentesis () a seguir ao nome da variável. Na instrução seguinte, é definido um vector com espaço para armazenar 3 elementos:

```
dim nomes(2)
```

O número dentro dos parentesis é 2. Começando a numeração dos elementos em zero, implica que o vector contém 3 elementos. Este é um vector de tamanho fixo. Podemos assumir valores para cada elemento da seguinte forma:

```
nomes(0) = "Tove"
nomes(1) = "Jani"
nomes(2) = "Stale"
```

De forma semelhante, os valores guardados podem ser consultados/acedidos indicando o nome da variável e a posição do elemento que se pretende. Por exemplo, se quisermos aceder ao primeiro valor guardado no vector podemos fazê-lo da seguinte forma:

```
mother=nomes(0)
```

Podemos ter até 60 dimensões num vector. Múltiplas dimensões são declaradas separando os números nos parentesis com vírgulas. Exemplo da declaração para obter uma matriz (vector de 2 dimensões) com 5 linhas e 7 colunas:

```
dim tabela(4, 6)
```

OPERADORES DO VBSCRIPT

Aritméticos		Comparação		Lógicos	
Descrição	Símbolo	Descrição	Símbolo	Descrição	Símbolo
Exponenciação	^	Igualdade	=	Negação Lógica	Not
Multiplicação	*	Desigualdade	<>	Conjunção Lógica	And
Divisão	/	Menor que	<	Disjunção Lógica	Or
Divisão inteira	\	Maior que	>	Exclusão Lógica	Xor
Módulus aritmético	Mod	Menor ou igual a	<=	Equivalência Lógica	Eqv
Adição	+	Maior ou igual a	>=	Implicação Lógica	Imp
Subtração	-	Equivalência Objecto	Is		
Junção de <i>Strings</i>	&				

SUB-ROTINAS NO VBSCRIPT

Sub-Rotinas do VBScript

Existem dois tipos de sub-rotinas: O procedimento *Sub* e a função *Function*.

Uma sub-rotina *Sub*:

- é um conjunto de instruções, delimitado pelas instruções *Sub* e *End Sub*
- pode desempenhar acções, mas **não devolve nenhum** resultado
- pode ter parâmetros de entrada ou não
- se não tiver parâmetros de entrada, deve incluir um par de parentesis vazios ()

```
Sub mysub()  
  ...  
  Instruções  
  ...  
End Sub  
  
or  
  
Sub mysub(argument1, argument2)  
  ...  
  Instruções  
  ...  
End Sub
```

Uma sub-rotina *Function*:

- é um conjunto de instruções, delimitado pelas instruções *Function* e *End Function*
- pode desempenhar acções e **pode devolver** um valor como resultado
- pode ter parâmetros de entrada ou não
- se não tiver parâmetros de entrada, deve incluir um par de parentesis vazios ()
- devolve um valor assumindo o nome da função ao valor que se pretende dar como resultado

```
Function myfunction()  
  ...  
  Instruções  
  ...  
  myfunction = valor  
End Function  
  
ou  
  
Function myfunction(argument1, argument2)  
  ...  
  Instruções  
  ...  
  myfunction = valor  
End Function
```

Chamando sub-rotinas Sub ou Function

Quando queremos chamar uma *Function*, podemos fazê-lo da seguinte forma:

```
name = findname()
```

Aqui chamamos uma *Function* denominada *findname*, que devolve um valor que será guardado na variável *name*.

Ou podemos fazê-lo da seguinte forma:

```
msgbox "O seu nome é " & findname()
```

Aqui também chamamos uma *Function* denominada *findname*, que retorna um valor que será mostrado numa caixa de mensagem.

Quando chamamos uma sub-rotina *Sub* podemos utilizar o comando *Call*, como na linha seguinte:

```
Call MyProc(argument)
```

Ou podemos omitir o comando *Call* (note-se que desta forma omitem-se os parentesis):

```
MyProc argument
```

ESTRUTURAS DE CONTROLE NO VBSCRIPT

Estruturas Condicionais

Quando desenvolvemos aplicações, por vezes queremos executar acções diferentes de acordo com decisões condicionais, com estruturas condicionais na programação para o fazer.

Em *VBScript* temos duas instruções condicionais principais:

- **if...then...else** – utilizamos esta estrutura de controlo quando desejamos apenas executar um de dois conjuntos de comandos consoante determinada situação é verdadeira ou falsa
- **select case** – utilizamos esta estrutura de controlo para executar determinados conjuntos de comandos de acordo com determinadas hipóteses de acontecimento

If...Then.....Else

Devemos utilizar a estrutura de controlo *If...Then...Else* se desejarmos:

- executar determinadas instruções caso determinada condição seja verdadeira
- seleccionar um de dois blocos de instruções a executar

Se quisermos executar apenas **uma** instrução quando uma condição se verifica, podemos escrever a instrução como a seguinte:

```
if i=10 Then msgbox "Olá"
```

A instrução acima não tem *..else..* na sua sintaxe. Indicamos apenas que queremos executar **uma acção** (escrever o texto olá numa janela de mensagem) caso a condição seja verdadeira (neste caso, se *i* for igual a 10).

Se desejarmos executar **mais do que uma instrução** quando a condição a testar seja verdadeira, temos de pôr cada instrução em linhas separadas e terminar o bloco de controlo com o comando *End If*:

```
if i=10 Then
  msgbox "Olá"
  i = i+1
end If
```

Esta sintaxe também não possui *..else..*. Apenas dizemos que queremos executar **um conjunto de instruções** se a condição for verdadeira.

Se desejarmos executar uma instrução ou um conjunto de instruções se uma condição for verdadeira e outra instrução ou conjunto de instruções caso a mesma condição for falsa, temos de adicionar o comando *Else*:

```
if i=10 then
  msgbox "Olá"
else
  msgbox "Adeus"
end If
```

O primeiro bloco de instruções será executado se a condição for verdadeira (se *i* for igual a 10) e o outro bloco de instruções será executado em caso contrário (se *i* não for igual a 10).

Select Case

Podemos utilizar o comando **SELECT** caso queiramos seleccionar um entre vários conjuntos de instruções a executar:

```
select case Pagamento
  case "Dinheiro"
    msgbox "Você vai pagar em dinheiro"
  case "MB"
    msgbox "Você vai pagar pelo multibanco"
  case "VISA"
    msgbox "Você vai pagar com cartão de crédito"
  case Else
    msgbox "Forma de pagamento inválida"
end select
```

Como funciona: Primeiro temos a expressão a avaliar, normalmente uma variável (primeira linha de instruções – *select case Pagamento*), que é avaliada. De acordo com o valor da expressão serão executadas as respectivas instruções definidas dentro dos blocos *case* que se refiram a esse mesmo valor. Se não tiver nenhum valor correspondente, será executado o bloco de instruções colocado no comando *case else*.

No exemplo mostrado, caso a variável *Pagamento* tenha o valor "MB", por exemplo, será executada a instrução *msgbox "Você vai pagar pelo multibanco"*.

ESTRUTURAS DE CICLO EM VBSCRIPT

Instruções de Ciclo

Muitas vezes ao programar necessitamos de repetir o mesmo bloco de instruções determinado número de vezes. Neste casos, devem ser utilizadas estruturas de ciclo.

Em *VBScript* existem quatro instruções que permitem definir ciclos:

- **For...Next statement** – executa um conjunto de instruções determinado número de vezes
- **For Each...Next statement** – executa um conjunto de instruções para cada item numa colecção ou cada elemento de um vector (*array*)
- **Do...Loop statement** – executa repetidamente um conjunto de instruções enquanto determinada condição é verdadeira (idêntico a *While...Wend*)

Ciclo For...Next

Podemos utilizar um ciclo **For...Next** para repetir um bloco de instruções, quando sabemos quantas repetições queremos.

Utilizamos uma variável contadora que é incrementada ou decrementada em cada passo de execução do ciclo, como no exemplo seguinte:

```
For i=1 to 6
...
  Instruções
...
Next
```

A instrução **For** especifica a variável contadora (**i**) e o seu valor inicial e valor final. Neste exemplo, a variável contadora (**i**) começa por ter um valor inicial igual a 1, sendo incrementado de 1 para cada passo de execução do ciclo, até atingir um valor igual a 10. Desta forma, a variável (**i**) assumirá os valores 1, 2, 3, 4, 5 e 6 para o 1.º, 2.º, 3.º, 4.º, 5.º e 6.º passo de execução do ciclo, respectivamente.

Comando Step

Utilizando o comando **Step**, podemos especificar qual o valor a incrementar ou decrementar à variável contadora a cada passo de execução do ciclo.

No exemplo seguinte, a variável contadora (**i**) é incrementada de dois em dois a cada passo de execução do ciclo:

```
For i=1 To 11 Step 2
...
  Instruções
...
Next
```

Neste caso, a variável tomará os valores 1, 3, 5, 7, 9 e 11. Para decrementar o valor da variável de ciclo em cada passo de execução, temos de utilizar um valor negativo de **Step**. E, obviamente, devemos especificar um valor final inferior ao valor inicial.

No exemplo seguinte, a variável contadora (**i**) é decrementada em dois de cada passo de execução do ciclo:

```
For i=11 To 2 Step -2
...
  Instruções
...
Next
```

Sair de um For...Next

Podemos sair antecipadamente de um ciclo *For...Next* utilizando o comando *Exit*.

Ciclo For Each...Next

O comando **For Each...Next** é muito idêntico ao commando *For...Next*. A diferença consiste em que não é necessário especificar o número de vezes que o ciclo irá executar, pois depende da colecção/vector de elementos a que se refere.

```
Dim names(2)
names(0)="Tove"
names(1)="Jani"
names(2)="Hege"

For Each x in names
    document.write(x & "<br />")
Next
```

No exemplo, a instrução *document.write* será executada três vezes, tomando *x* os valores de *names(0)*, *names(1)* e *names(2)* para o 1.º, 2.º e 3.º passo de execução do ciclo, respectivamente.

Ciclo Do...Loop

Podemos utilizar o comando **Do...Loop** para executar um bloco de instruções quando não sabemos o número de repetições que precisamos. O conjunto de instruções é repetido enquanto a condição definida for verdadeira (utilizando o comando *While*) ou até que a condição seja verdadeira (utilizando o comando *Until*).

Repetir Instruções enquanto uma Condição seja Verdadeira

O exemplo seguinte utiliza o comando *While* para testar uma condição num comando *Do...Loop*.

```
Do While i>10
    ...
    Instruções
    ...
Loop
```

Se inicialmente *i* for igual a 9, as instruções dentro do ciclo nunca serão executadas.

```
Do
    ...
    Instruções
    ...
Loop While i>10
```

As instruções dentro deste ciclo serão executadas pelo menos uma vez, mesmo que *i* seja menor que 10.

Repetir instruções Até Que uma Condição seja Verdadeira

Podemos utilizar o comando *Until* para testar uma condição num ciclo *Do...Loop*.

```
Do Until i=10
    ...
    instruções
    ...
Loop
```

Se *i* for igual a 10, as instruções dentro deste ciclo nunca serão executadas.

```
Do
  ...
  instruções
  ...
Loop Until i=10
```

As instruções dentro deste ciclo serão executadas pelo menos uma vez, mesmo que *i* seja igual a 10.

Saltar fora de um Do...Loop

Podemos saltar fora de um ciclo *Do...Loop* com o comando *Exit Do*.

```
Do Until i=10
  i=i-1
  If i<10 Then Exit Do
Loop
```

O código dentro deste ciclo será executado enquanto *i* for diferente de 10, e enquanto *i* for maior que 10.

FUNCÕES VBSCRIPT

Função: CDate

Converte uma expressão *string* de data e tempo válida para uma variante do tipo *Date*.

Exemplo 1

```
d="Fevereiro 22, 2001"
if IsDate(d) then
  document.write(CDate(d))
end if
Output:
2/22/01
```

Exemplo 2

```
d=#2/22/01#
if IsDate(d) then
  document.write(CDate(d))
end if
Output:
2/22/01
```

Exemplo 3

```
d="3:18:40 AM"
if IsDate(d) then
  document.write(CDate(d))
end if
Output:
3:18:40 AM
```

Função: Date

Retorna a data actual do sistema.

Exemplo

```
document.write("A data actual do sistema é: ")
document.write(Date)
Output:
A data actual do sistema é: 1/14/2002
```

Função: DateAdd

Retorna uma data à qual um intervalo de tempo específico foi adicionado.

Sintaxe

```
DateAdd(interval,number,date)
```

Parâmetro	Descrição
interval	O intervalo que se quer adicionar. Pode tomar os seguintes valores: <ul style="list-style-type: none">• yyyy – Ano• q – Quarter• m – Mês• y – Dia do ano• d – Dia• w – Dia da semana• ww – Semana do ano• h – Hora• n – Minuto• s – Segundo
Number	O tamanho do intervalo que se quer adicionar. Pode ser positivo para obter datas futuras, ou negativas para obter datas passadas
Date	Variante ou literal representando a data ao qual se pretende adicionar o intervalo de tempo

Exemplo 1

```
'Adicionar um mês a Janeiro 31, 2000
document.write(DateAdd("m",1,"31-Jan-00"))
Output:
2/29/2000
```

Exemplo 2

```
'Adicionar um mês a Janeiro 31, 2001
document.write(DateAdd("m",1,"31-Jan-01"))
Output:
2/28/2001
```

Exemplo 3

```
'Subtrair um mês a Janeiro 31, 2001
document.write(DateAdd("m",-1,"31-Jan-01"))
Output:
12/31/2000
```

Função: DateDiff

Retorna a diferença entre duas datas.

Sintaxe

```
DateDiff(interval,date1,date2[,firstdayofweek[,firstweekofyear]])
```

Parâmetro	Descrição
interval	O tipo de medida do intervalo cuja diferença entre date1 e date2 desejamos calcular. Pode tomar os seguintes valores: <ul style="list-style-type: none"> • yyyy - Ano • q - Quarter • m - Mês • y - Dia do ano • d - Dia • w - Dia da semana • ww - Semana do ano • h - Hora • n - Minuto • s - Segundo
date1,date2	Expressões/valores data. As duas datas a usar para o cálculo pretendido.
firstdayofweek	Opcional. Especifica o dia que é considerado o primeiro da semana. Pode tomar os seguintes valores: <ul style="list-style-type: none"> • 0 = vbUseSystemDayOfWeek - Use (NLS) API setting • 1 = vbSunday - Domingo (default) • 2 = vbMonday - 2ª feira • 3 = vbTuesday - 3ª feira • 4 = vbWednesday - 4ª feira • 5 = vbThursday - 5ª feira • 6 = vbFriday - 6ª feira • 7 = vbSaturday - Sábado
firstweekofyear	Opcional. Especifica a 1.ª semana do ano. Pode tomar os seguintes valores: <ul style="list-style-type: none"> • 0 = vbUseSystem - Use National Language Support (NLS) API setting • 1 = vbFirstJan1 - Começa na semana de 1 de Janeiro (default) • 2 = vbFirstFourDays - Começa na semana que tem pelo menos 4 dias no novo ano • 3 = vbFirstFullWeek - Começa com a semana que tem todos os dias já no novo ano

Exemplo 1

```
document.write(Date & "<br />")
document.write(DateDiff("m",Date,"12/31/2002") & "<br />")
document.write(DateDiff("d",Date,"12/31/2002") & "<br />")
document.write(DateDiff("n",Date,"12/31/2002"))
Output:
1/14/2002
11
351
505440
```

Exemplo 2

```
document.write(Date & "<br />")
document.write(DateDiff("d","12/31/2002",Date))
Output:
1/14/2002
-351
```

Exemplo 3

```
'Quantas semanas existem (começando na 2ªfeira), entra a data actual e  
'10/10/2002  
document.write(Date & "<br />")  
document.write(DateDiff("w",Date,"10/10/2002",vbMonday))  
Output:  
1/14/2002  
38
```

Função: Day

Retorna o dia do mês numa data (de 1 a 31).

Sintaxe

```
Day(date)
```

Parâmetro	Descrição
date	Qualquer expressão que represente uma data.

Exemplo

```
document.write(Date & "<br />")  
document.write(Day(Date))  
Output:  
1/14/2002  
14
```

Função: Month

Idêntica a *Day*, mas para o mês do ano (de 1 a 12).

Função: Year

Idêntica a *Day*, mas para o ano.

Função: Hour

Retorna a hora de determinada hora.

Sintaxe

```
Hour(time)
```

Parâmetro	Descrição
time	Qualquer expressão que represente uma hora.

Exemplo 1

```
document.write(Now & "<br />")  
document.write(Hour(Now))  
Output:  
1/15/2002 10:07:47 AM  
10
```

Exemplo 2

```
document.write(Hour(Time))
Output:
10
```

Função: Minute

Idêntico a *Hour*, mas para o minuto da hora.

Função: Second

Idêntico a *Hour*, mas para o segundo do minuto da hora.

Função: Now

Retorna a data e hora actuais do sistema.

Sintaxe

```
Now
```

Exemplo

```
document.write(Now)
Output:
1/15/2002 10:52:15 AM
```

Função: Weekday

Retorna o dia da semana (de 1 a 7 – por defeito, Domingo a Sábado)

Sintaxe

```
Weekday(date[,firstdayofweek])
```

Parâmetro	Descrição
Date	A expressão data a avaliar.
firstdayofweek	<p>Opcional. Especifica o primeiro dia da semana. Pode tomar os seguintes valores:</p> <p>Can take the following values:</p> <ul style="list-style-type: none"> • 0 = vbUseSystemDayOfWeek - Use National Language Support (NLS) API setting • 1 = vbSunday – Domingo (default) • 2 = vbMonday – 2ª feira • 3 = vbTuesday – 3ª feira • 4 = vbWednesday – 4ª feira • 5 = vbThursday – 5ª feira • 6 = vbFriday – 6ª feira • 7 = vbSaturday – Sábado

Exemplo

```
document.write(Date & "<br />")
document.write(Weekday(Date))
```

```
Output :
1/15/2002
3
```

Função: FormatDateTime

Retorna uma expressão data/hora formatada de acordo com preferências.

Sintaxe

```
FormatDateTime (date, format)
```

Parâmetro	Descrição
date	Qualquer expressão data válida (como Date() ou Now())
format	Opcional. Um valor de Format que especifica o formato data/hora a usar

Exemplo 1

```
document.write("A data actual é: ")
document.write(FormatDateTime(Date()))
Output:
A data actual é: 2/22/2001
```

Exemplo 2

```
document.write("A data actual é: ")
document.write(FormatDateTime(Date(),1))
Output:
A data actual é: Thursday, February 22, 2001
```

Exemplo 3

```
document.write("A data actual é: ")
document.write(FormatDateTime(Date(),2))
Output:
A data actual é: 2/22/2001
```

Format Values

Constant	Value	Descrição
vbGeneralDate	0	Mostra a data em formato mm/dd/yy. Se o parâmetro data for Now(), também devolve a hora, depois da data
vbLongDate	1	Mostra a data utilizando um formato longo: dia da semana, mês, dia do mês, ano
vbShortDate	2	Mostra a data utilizando um formato abreviado: como o default (mm/dd/yy)
vbLongTime	3	Mostra a hora usando o formato time: hh:mm:ss PM/AM
vbShortTime	4	Mostra a hora usando o formato 24h: hh:mm

Função: FormatNumber

Retorna um valor numérico formato de acordo com preferências.

Sintaxe

```
FormatNumber( Expression[ , NumDigAfterDec[ ,  
IncLeadingDig[ , UseParForNegNum[ , GroupDig ] ] ] ] )
```

Parâmetro	Descrição
expression	O valor numérico a ser formatado (valor ou variável)
NumDigAfterDec	Opcional. Indica quantas casas decimais a usar. Default é -1 (utiliza as definições regionais do próprio computador)
IncLeadingDig	Opcional. Indica se deve ou não ser colocado um zero para valores fraccionários: <ul style="list-style-type: none"> • -2 = TristateUseDefault - Usa as definições regionais do computador • -1 = TristateTrue - True • 0 = TristateFalse - False
UseParForNegNum	Opcional. Indica se valores negativos devem ou não ser inscritos entre parentesis: <ul style="list-style-type: none"> • -2 = TristateUseDefault - Usa as definições regionais do computador • -1 = TristateTrue - True • 0 = TristateFalse - False
GroupDig	Opcional. Indica se os grupos dos milhares devem ou não ser separados pelos respectivos delimitadores definidos nas definições regionais do computador: <ul style="list-style-type: none"> • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False

Exemplo 1

```
document.write(FormatNumber(20000))  
Output:  
20,000.00
```

Exemplo 2

```
document.write(FormatNumber(20000.578,2))  
Output:  
20,000.58
```

Exemplo 3

```
document.write(FormatNumber(20000.578,2,,0))  
Output:  
20000.58
```

Função: CBool

Retorna o valor de uma expressão convertido em booleano.

Sintaxe

```
CBool( expressão )
```

Função: CByte

Retorna o valor de uma expressão convertido em byte.

Sintaxe

```
CByte (expressão)
```

Função: CDate

Retorna o valor de uma expressão convertido em data.

Sintaxe

```
CDate (expressão)
```

Parâmetro	Descrição
expressão	Requerido. Qualquer expressão de data válida (como Date() or Now())

Exemplo 1

```
d="Fevereiro 22, 2001"  
if IsDate(d) then  
    document.write(CDate(d))  
end if  
Output:  
2/22/01
```

Exemplo 2

```
d=#2/22/01#  
if IsDate(d) then  
    document.write(CDate(d))  
end if  
Output:  
2/22/01
```

Exemplo 3

```
d="3:18:40 AM"  
if IsDate(d) then  
    document.write(CDate(d))  
end if  
Output:  
3:18:40 AM
```

Função: CDb1

Retorna o valor de uma expressão convertido em real duplo.

Sintaxe

```
CDbl (expressão)
```

Função: CInt

Retorna o valor de uma expressão convertido em inteiro.

Sintaxe

```
CInt ( expressão )
```

Função: CLng

Retorna o valor de uma expressão convertido em inteiro longo.

Sintaxe

```
CLng ( expressão )
```

Função: CSng

Retorna o valor de uma expressão convertido em real simples.

Sintaxe

```
CSng ( expressão )
```

Função: CStr

Retorna o valor de uma expressão convertido em *string*.

Sintaxe

```
CStr ( expressão )
```

Função: Asc

Retorna o valor na tabela ASCII correspondente a determinado caracter.

Sintaxe

```
Asc ( caracter )
```

Função: Chr

Retorna o caracter identificado por determinado número correspondente na tabela ASCII.

Sintaxe

```
Chr ( number )
```

Função: Abs

Retorna o valor absoluto de um valor numérico.

Sintaxe

```
Abs ( number )
```

Função: Int

Retorna apenas a parte inteira de um valor numérico.

Sintaxe

```
Int (number)
```

Função: Rnd

Retorna um número fraccionário aleatório entre 0 e 1.

Sintaxe

```
Rnd[ (number) ]
```

Parâmetro	Descrição
number	Opcional. Um valor numérico válido. Hipóteses para este número são: <ul style="list-style-type: none">• <0 - Rnd devolve o mesmo número sempre• >0 - Rnd devolve o próximo nº aleatório da sequência• =0 - Rnd devolve o número gerado mais recente• Não preenchido - Rnd devolve o nº aleatório seguinte na sequência

Exemplo 1

```
document.write(Rnd)
Output:
0.7055475
```

Exemplo 2

```
'Mandando fazer o refresh da página, usando o código do Exemplo 1,
' o MESMO nº aleatório sera mostrado repetidamente.
' Use a instrução Randomize para gerar um novo nº aleatório cada vez que a
' página é novamente carregada!
Randomize
document.write(Rnd)
Output:
0.4758112
```

Exemplo 3

```
'Aqui está como produzir nºs aleatórios dentro de um intervalo:
dim max,min
max=100
min=1
document.write(Int((max-min+1)*Rnd+min))
Output:
71
```

Função: Sqr

Retorna a raiz quadrada de um valor numérico.

Sintaxe

```
Sqr (number)
```

Função: Array

Retorna um vector contendo a lista dos elementos do seu argumento (o primeiro elemento é o n.º 0).

Sintaxe

```
Array(arglist)
```

Parâmetro	Descrição
arglist	Uma lista (separada por vírgulas) dos valores dos elementos do vector

Exemplo

```
dim a
a=Array(5,10,15,20)
document.write(a(3))
Output:
20
```

Função: InStr

Devolve a posição da primeira ocorrência de uma *string* dentro de outra.

A função InStr devolve um dos seguintes valores:

- Se string1 for "" - InStr devolve 0
- Se string1 for Null - InStr devolve Null
- Se string2 for "" - InStr devolve start
- Se string2 for Null - InStr devolve Null
- Se string2 não existir em string1 - InStr devolve 0
- Se string2 for encontrado em string1 - InStr devolve a posição em string1 na qual encontrou a string2
- Se start > Len(string2) - InStr devolve 0

Sintaxe

```
InStr([start,]string1,string2)
```

Parâmetro	Descrição
Start	Opcional. Especifica a posição a partir da qual procurar. Por defeito, a busca começa a partir do 1.º caracter.
string1	A string na qual procurar
string2	A string que se pretende ver se se encontra dentro de string1

Exemplo 1

```
dim txt,pos
txt="This is a beautiful day!"
pos=InStr(txt,"his")
document.write(pos)
Output:
2
```

Exemplo 2

```
dim txt,pos
txt="This is a beautiful day!"
'A textual comparison starting at position 4
pos=Instr(4,txt,"is",1)
document.write(pos)
Output:
6
```

Função: LCase

Esta função devolve uma *string* toda convertida em caracteres minúsculos.

Sintaxe

```
LCase(string)
```

Parâmetro	Descrição
string	A string a converter integralmente em minúsculas

Exemplo 1

```
dim txt
txt="ESTÁ UM DIA BONITO!"
document.write(LCase(txt))
Output:
está um dia bonito!
```

Exemplo 2

```
dim txt
txt="Está Um Dia Bonito!"
document.write(LCase(txt))
Output:
está um dia bonito!
```

Função: UCase

Esta função devolve uma *string* toda convertida em caracteres maiúsculos. A sua sintaxe é idêntica à função *LCase*.

Sintaxe

```
UCase(string)
```

Função: Len

Esta função devolve o número de caracteres que compõem uma *string*.

Sintaxe

```
Len(string/variável)
```

Parâmetro	Descrição
string	Uma expressão de valor string

variável	O nome de uma variável do tipo string
----------	---------------------------------------

Exemplo 1

```
dim txt
txt="Está um dia lindo!"
document.write(Len(txt))
Output:
18
```

Exemplo 2

```
document.write(Len("Está um dia lindo!"))
Output:
18
```

Função: Trim

Esta função devolve uma *string* retirando-lhe os espaços à esquerda e à direita.

Sintaxe

```
Trim(string)
```

Parâmetro	Descrição
string	Uma expressão string

Exemplo

```
dim txt
txt="  Está um dia bonito!  "
document.write(Trim(txt))
Output:
Está um dia bonito!
```

Função: LTrim, RTrim

Estas funções devolvem uma *string* sem espaços brancos à esquerda ou à direita, respectivamente.

Sintaxe

```
LTrim(string) / RTrim(string)
```

Parâmetro	Descrição
string	Uma expressão string

Exemplo

```
dim txt
txt="  Está um dia lindo!  "
document.write(LTrim(txt))
Output:
Está um dia lindo!  "
```

Função: Mid

Esta função devolve parte de uma *string*.

Sintaxe

```
Mid(string, start[, length])
```

Parâmetro	Descrição
string	A expressão string da qual se quer devolver uma parte
start	Especifica a posição (número do character) a partir da qual queremos começar a contar os caracteres a devolver. Se for superior ao número de caracteres da string devolve uma string vazia ("")
length	Quantos caracteres se pretendem devolver

Exemplo 1

```
dim txt
txt="Está um dia lindo!"
document.write(Mid(txt,1,1))
Output:
E
```

Exemplo 2

```
dim txt
txt="Está um dia lindo!"
document.write(Mid(txt,1,11))
Output:
Está um dia
```

Função: Space

Esta função devolve uma *string* com determinado número de espaços.

Sintaxe

```
Space(number)
```

Parâmetro	Descrição
number	O número de espaços que desejamos na string a devolver

Função: String

Esta função devolve uma *string* contendo a repetição de determinado character determinado número de vezes.

Sintaxe

```
String(number, character)
```

Parâmetro	Descrição
number	O número de vezes que queremos ver repetido o carácter
character	O character a ser repetido

Exemplo 1

```
document.write(String(10, "#"))
Output:
#####
```

Exemplo 2

```
document.write(String(4, "*"))
Output:
****
```

Função: CreateObject

Esta função permite criar para manipulação um objecto da *Microsoft Object Library*.

Sintaxe

```
CreateObject(servername.typename[, location])
```

Parâmetro	Descrição
servername	O nome da aplicação-mãe registada responsável pelo objecto
typename	O tipo/class do objecto
location	Opcional. Onde criar o objecto

Exemplo

```
dim myexcel
Set myexcel=CreateObject("Excel.Sheet")
myexcel.Application.Visible=True
... instruções ...
myexcel.Application.Quit
Set myexcel=Nothing
```

Função: GetLocale

Esta função permite saber a localização geográfica relativa do servidor que provém a ligação Internet ao *browser* cliente.

Sintaxe

```
GetLocale()
```

Exemplo

```
dim c
c=GetLocale
document.write(c)
Output:
1033
```

Locale ID Chart

Locale Descrição	Short String	Hex Value	Decimal Value
Afrikaans	af	0x0436	1078
Albanian	sq	0x041C	1052
Arabic - United Arab Emirates	ar-ae	0x3801	14337

Arabic - Bahrain	ar-bh	0x3C01	15361
Arabic - Algeria	ar-dz	0x1401	5121
Arabic - Egypt	ar-eg	0x0C01	3073
Arabic - Iraq	ar-iq	0x0801	2049
Arabic - Jordan	ar-jo	0x2C01	11265
Arabic - Kuwait	ar-kw	0x3401	13313
Arabic - Lebanon	ar-lb	0x3001	12289
Arabic - Libya	ar-ly	0x1001	4097
Arabic - Morocco	ar-ma	0x1801	6145
Arabic - Oman	ar-om	0x2001	8193
Arabic - Qatar	ar-qa	0x4001	16385
Arabic - Saudi Arabia	ar-sa	0x0401	1025
Arabic - Syria	ar-sy	0x2801	10241
Arabic - Tunisia	ar-tn	0x1C01	7169
Arabic - Yemen	ar-ye	0x2401	9217
Armenian	hy	0x042B	1067
Azeri - Latin	az-az	0x042C	1068
Azeri - Cyrillic	az-az	0x082C	2092
Basque	eu	0x042D	1069
Belarusian	be	0x0423	1059
Bulgarian	bg	0x0402	1026
Catalan	ca	0x0403	1027
Chinese - China	zh-cn	0x0804	2052
Chinese - Hong Kong S.A.R.	zh-hk	0x0C04	3076
Chinese - Macau S.A.R.	zh-mo	0x1404	5124
Chinese - Singapore	zh-sg	0x1004	4100
Chinese - Taiwan	zh-tw	0x0404	1028
Croatian	hr	0x041A	1050
Czech	cs	0x0405	1029
Danish	da	0x0406	1030
Dutch - The Netherlands	nl-nl	0x0413	1043
Dutch - Belgium	nl-be	0x0813	2067
English - Australia	en-au	0x0C09	3081
English - Belize	en-bz	0x2809	10249
English - Canada	en-ca	0x1009	4105
English - Carriibbean	en-cb	0x2409	9225
English - Ireland	en-ie	0x1809	6153
English - Jamaica	en-jm	0x2009	8201
English - New Zealand	en-nz	0x1409	5129
English - Phillippines	en-ph	0x3409	13321
English - South Africa	en-za	0x1C09	7177
English - Trinidad	en-tt	0x2C09	11273
English - United Kingdom	en-gb	0x0809	2057
English - United States	en-us	0x0409	1033

Estonian	et	0x0425	1061
Farsi	fa	0x0429	1065
Finnish	fi	0x040B	1035
Faroese	fo	0x0438	1080
French - France	fr-fr	0x040C	1036
French - Belgium	fr-be	0x080C	2060
French - Canada	fr-ca	0x0C0C	3084
French - Luxembourg	fr-lu	0x140C	5132
French - Switzerland	fr-ch	0x100C	4108
Gaelic - Ireland	gd-ie	0x083C	2108
Gaelic - Scotland	gd	0x043C	1084
German - Germany	de-de	0x0407	1031
German - Austria	de-at	0x0C07	3079
German - Liechtenstein	de-li	0x1407	5127
German - Luxembourg	de-lu	0x1007	4103
German - Switzerland	de-ch	0x0807	2055
Greek	el	0x0408	1032
Hebrew	he	0x040D	1037
Hindi	hi	0x0439	1081
Hungarian	hu	0x040E	1038
Icelandic	is	0x040F	1039
Indonesian	id	0x0421	1057
Italian - Italy	it-it	0x0410	1040
Italian - Switzerland	it-ch	0x0810	2064
Japanese	ja	0x0411	1041
Korean	ko	0x0412	1042
Latvian	lv	0x0426	1062
Lithuanian	lt	0x0427	1063
FYRO Macedonian	mk	0x042F	1071
Malay - Malaysia	ms-my	0x043E	1086
Malay - Brunei	ms-bn	0x083E	2110
Maltese	mt	0x043A	1082
Marathi	mr	0x044E	1102
Norwegian - Bokmål	no-no	0x0414	1044
Norwegian - Nynorsk	no-no	0x0814	2068
Polish	pl	0x0415	1045
Portuguese - Portugal	pt-pt	0x0816	2070
Portuguese - Brazil	pt-br	0x0416	1046
Raeto-Romance	rm	0x0417	1047
Romanian - Romania	ro	0x0418	1048
Romanian - Moldova	ro-mo	0x0818	2072
Russian	ru	0x0419	1049
Russian - Moldova	ru-mo	0x0819	2073
Sanskrit	sa	0x044F	1103

Serbian - Cyrillic	sr-sp	0x0C1A	3098
Serbian - Latin	sr-sp	0x081A	2074
Setsuana	tn	0x0432	1074
Slovenian	sl	0x0424	1060
Slovak	sk	0x041B	1051
Sorbian	sb	0x042E	1070
Spanish - Spain	es-es	0x0C0A	1034
Spanish - Argentina	es-ar	0x2C0A	11274
Spanish - Bolivia	es-bo	0x400A	16394
Spanish - Chile	es-cl	0x340A	13322
Spanish - Colombia	es-co	0x240A	9226
Spanish - Costa Rica	es-cr	0x140A	5130
Spanish - Dominican Republic	es-do	0x1C0A	7178
Spanish - Ecuador	es-ec	0x300A	12298
Spanish - Guatemala	es-gt	0x100A	4106
Spanish - Honduras	es-hn	0x480A	18442
Spanish - Mexico	es-mx	0x080A	2058
Spanish - Nicaragua	es-ni	0x4C0A	19466
Spanish - Panama	es-pa	0x180A	6154
Spanish - Peru	es-pe	0x280A	10250
Spanish - Puerto Rico	es-pr	0x500A	20490
Spanish - Paraguay	es-py	0x3C0A	15370
Spanish - El Salvador	es-sv	0x440A	17418
Spanish - Uruguay	es-uy	0x380A	14346
Spanish - Venezuela	es-ve	0x200A	8202
Sutu	sx	0x0430	1072
Swahili	sw	0x0441	1089
Swedish - Sweden	sv-se	0x041D	1053
Swedish - Finland	sv-fi	0x081D	2077
Tamil	ta	0x0449	1097
Tatar	tt	0X0444	1092
Thai	th	0x041E	1054
Turkish	tr	0x041F	1055
Tsonga	ts	0x0431	1073
Ukrainian	uk	0x0422	1058
Urdu	ur	0x0420	1056
Uzbek - Cyrillic	uz-uz	0x0843	2115
Uzbek - Latin	uz-uz	0x0443	1091
Vietnamese	vi	0x042A	1066
Xhosa	xh	0x0434	1076
Yiddish	yi	0x043D	1085
Zulu	zu	0x0435	1077

Função: GetRef

Esta função permite conectar uma sub-rotina *VBScript* a um acontecimento DHTML no documento *Web*.

Sintaxe

```
Set object.event=GetRef(procname)
```

Parâmetro	Descrição
object	O nome do objecto DHTML com o qual o evento DHTML está associado
event	O nome do evento DHTML ao qual a função está referenciada
procname	O nome da sub-rotina a associar ao evento DHTML

Exemplo

```
Function test()
  dim txt
  txt="GetRef Test" & vbCrLf
  txt=txt & "Hello World!"
  MsgBox txt
End Function
Set Window.Onload=GetRef("test")
```

Função: InputBox

Esta função permite abrir uma janela de diálogo com uma caixa de introdução de texto para o utilizador, devolvendo o valor por ele introduzido.

Sintaxe

```
InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile,context])
```

Parâmetro	Descrição
prompt	Define a mensagem a mostrar na janela de diálogo. O comprimento máximo é de 1024 caracteres. Podemos separar linhas utilizando o caracter carriage return (Chr(13)), um caracter linefeed (Chr(10)), entre cada linha
title	Opcional. O título da janela de diálogo. Por defeito é o nome da aplicação
default	Opcional. Um texto a assumir por defeito na caixa de introdução de texto
xpos	Opcional. A distância da margem esquerda da janela de diálogo relativamente ao limite esquerdo do écran. Se for omitido, a janela aparecerá centrada
ypos	Opcional. A distância da margem superior da janela de diálogo relativamente ao limite superior do écran. Se for omitido, a janela aparecerá centrada
helpfile	Opcional. O nome do ficheiro Help a usar. Tem de ser usado em conjunto com o parâmetro context
context	Opcional. O contexto de Help do número de tópico de ajuda. Tem de ser usado em conjunto com o parâmetro helpfile

Exemplo

```
dim fnome
fnome = InputBox("Introduza o seu nome:")
MsgBox("O seu nome é " & fnome)
```

Função: IsEmpty

Esta função permite testar se determinada variável se encontra inicializada ou não, devolvendo *False* caso não tenha sido ainda atribuído um valor e *False* em caso contrário.

Sintaxe

```
IsEmpty(variável)
```

Parâmetro	Descrição
variável	A variável a testar

Exemplo

```
dim x
document.write(IsEmpty(x) & "<br />")
x=10
document.write(IsEmpty(x) & "<br />")
x=Empty
document.write(IsEmpty(x) & "<br />")
x=NULL
document.write(IsEmpty(x))
Output:
True
False
True
False
```

Função: IsNull

Esta função devolve *True* caso o conteúdo da expressão a testar seja nulo, ou devolve *False* em caso contrário.

Sintaxe

```
IsNull(expressão)
```

Parâmetro	Descrição
Expressão	A expressão a avaliar

Exemplo

```
dim x
document.write(IsNull(x) & "<br />")
x=10
document.write(IsNull(x) & "<br />")
x=Empty
document.write(IsNull(x) & "<br />")
x=NULL
document.write(IsNull(x))
Output:
False
False
False
True
```

Função: IsNumeric

Esta função devolve o valor *True* caso a expressão a avaliar seja um valor numérico, ou *False* em caso contrário.

Sintaxe

```
IsNumeric( expressão )
```

Parâmetro	Descrição
expressão	A expressão a avaliar

Exemplo

```
dim x
x=10
document.write(IsNumeric(x) & "<br />")
x=Empty
document.write(IsNumeric(x) & "<br />")
x=NULL
document.write(IsNumeric(x) & "<br />")
x="10"
document.write(IsNumeric(x) & "<br />")
x="911 Help"
document.write(IsNumeric(x))
Output:
True
True
False
True
False
```

Função: MsgBox

Esta função permite abrir uma caixa de diálogo no écran e devolver um valor consoante o botão premido pelo utilizador para fechá-la. O valor devolvido poderá ser um dos seguintes:

- 1 = vbOK (foi premido o botão OK)
- 2 = vbCancel (foi premido o botão Cancel)
- 3 = vbAbort (foi premido o botão Abort)
- 4 = vbRetry (foi premido o botão Retry)
- 5 = vbIgnore (foi premido o botão Ignore)
- 6 = vbYes (foi premido o botão Yes)
- 7 = vbNo (foi premido o botão No)

Nota: O utilizador poderá premir a tecla F1 para visualizar o tópico de ajuda quando ambos os parâmetros *helpfile* e *context* forem especificados.

Sintaxe

```
MsgBox mensagem[, aspecto][, titulo][, helpfile, context ]
```

Parâmetro	Descrição
mensagem	Requerido. A mensagem a mostrar na janela (comprimento máximo de 1024 caracteres). No caso de se desejar escrever várias linhas, podemos utilizar o carácter 13 (ENTER) para mudar de linha (chr(13)).
aspecto	Opcional. Um valor ou soma de valores que especifica o aspecto do conteúdo a visualizar, nomeadamente: tipo e número de botões, ícone a visualizar, qual o botão por defeito e modalidade da janela. O valor por defeito deste parâmetro é 0. <ul style="list-style-type: none"> • 0 = <i>vbOKOnly</i> - apenas o botão OK button

	<ul style="list-style-type: none"> • 1 = <i>vbOKCancel</i> – botões OK e Cancel • 2 = <i>vbAbortRetryIgnore</i> – botões Abort, Retry, e Ignore • 3 = <i>vbYesNoCancel</i> – botões Yes, No, e Cancel • 4 = <i>vbYesNo</i> – botões Yes e No • 5 = <i>vbRetryCancel</i> – botões Retry e Cancel • 16 = <i>vbCritical</i> – ícone de Critical Message • 32 = <i>vbQuestion</i> – ícone de Confirmação • 48 = <i>vbExclamation</i> – ícone de Aviso • 64 = <i>vbInformation</i> – ícone de Informação • 0 = <i>vbDefaultButton1</i> – O primeiro botão é assumido por defeito • 256 = <i>vbDefaultButton2</i> – O segundo botão é assumido por defeito • 512 = <i>vbDefaultButton3</i> – O terceiro botão é assumido por defeito • 768 = <i>vbDefaultButton4</i> – O quarto botão é assumido por defeito • 0 = <i>vbApplicationModal</i> – Janela Modal (enquanto o utilizador não fechar a janela de diálogo não poderá prosseguir dentro da aplicação actual) • 4096 = <i>vbSystemModal</i> – Janela System Modal (enquanto o utilizador não fechar a janela de diálogo não poderá prosseguir em nenhuma aplicação que esteja aberta) <p>Podemos utilizar a soma dos valores para definição do aspecto da janela de diálogo. Por exemplo, se quisermos uma janela de diálogo onde apareçam os botões <i>Yes</i> e <i>No</i> (4=<i>vbYesNo</i>), com o ícone de Confirmação (32=<i>vbQuestion</i>), em que o botão por defeito é o botão <i>No</i> (256=<i>vbDefaultButton2</i>), então no parâmetro <i>aspecto</i> podemos colocar 4+32+256 ou 292.</p>
titulo	Opcional. O título da janela de diálogo. Por defeito utiliza o nome da janela da aplicação de onde foi chamada a função <i>msgbox</i>
helpfile	Opcional. O nome do ficheiro de ajuda a utilizar. Tem de ser utilizado obrigatoriamente em conjunto com o parâmetro <i>context</i>
context	Opcional. O número de índice do tópico de ajuda. Tem de ser utilizado obrigatoriamente em conjunto com o parâmetro <i>helpfile</i>

Exemplo

```

dim resposta
resposta = MsgBox "Olá a Todos!" & chr(13) & "Clique OK.",65,"Exemplo"
document.write(resposta)
Output:
1 (caso o utilizador prima o botão OK)
2 (caso o utilizador prima o botão Cancel)
    
```

Função: Round

Esta função devolve um valor numérico arredondado ao número de casas decimais desejadas.

Sintaxe

```
Round(expressão[, numcasasdecimais])
```

Parâmetro	Descrição
expressão	Requerido. A expressão numérico a arredondar
numcasasdecimais	Opcional. Especifica o número de casas decimais a arredondar. Por defeito é 0

Exemplo

```

dim x
x=24.13278
document.write(Round(x))
    
```

```
document.write(Round(x,2))
Output:
24
24.13
```

Função: VarType

Esta função devolve o tipo de valor que a variável a avaliar contém. As várias hipóteses são:

- 0 = vbEmpty (não foi ainda inicializada)
- 1 = vbNull (possui conteúdo nulo)
- 2 = vbInteger (possui um numérico inteiro)
- 3 = vbLong (possui um numérico inteiro longo)
- 4 = vbSingle (possui um numérico real simples)
- 5 = vbDouble (possui um numérico real duplo)
- 6 = vbCurrency(possui um numérico moeda)
- 7 = vbDate (possui uma data)
- 8 = vbString (possui um alfanumérico)
- 9 = vbObject (possui um objecto)
- 10 = vbError (indica um erro)
- 11 = vbBoolean(indica um booleano)
- 12 = vbVariant(indica um Variant – utilizado apenas com vectores de Variants)
- 13 = vbDataObject (indica um objecto de acesso/interface a dados)
- 17 = vbByte (indica um numérico Byte)
- 8192 = vbArray(indica um vector)

Sintaxe

```
VarType(variável)
```

Parâmetro	Descrição
variável	Requerido. O nome da variável a avaliar

Exemplo

```
dim x
x="Olá Mundo!"
document.write(VarType(x) & "<br />")
x=4
document.write(VarType(x) & "<br />")
x=4.675
document.write(VarType(x) & "<br />")
x=NULL
document.write(VarType(x) & "<br />")
x=Empty
document.write(VarType(x) & "<br />")
x=True
document.write(VarType(x))
Output:
8
2
5
1
0
11
```

Quadro Resumo para Consulta Rápida das Principais Funções do VBScript

FUNÇÃO	DESCRIÇÃO	SINTAXE
CDate	Converte uma expressão <i>string</i> de data e tempo válida para uma variante do tipo <i>Date</i> .	CDate(expressão)
Date	Retorna a data actual do sistema.	Date
DateAdd	Retorna uma data à qual um intervalo de tempo específico foi adicionado.	DateAdd(interval,number,date)
DateDiff	Retorna a diferença entre duas datas.	DateDiff(interval,date1,date2[,firstdayofweek[,firstweekofyear]])
Day	Retorna o dia do mês numa data (de 1 a 31).	Day(date)
Month	Idêntica a <i>Day</i> , mas para o mês do ano (de 1 a 12).	Month(date)
Year	Idêntica a <i>Day</i> , mas para o ano.	Year(date)
Hour	Retorna a hora de determinada hora.	Hour(time)
Minute	Idêntico a <i>Hour</i> , mas para o minuto da hora.	Minute(time)
Second	Idêntico a <i>Hour</i> , mas para o segundo do minuto da hora.	Second(time)
Now	Retorna a data e hora actuais do sistema.	Now
Weekday	Retorna o dia da semana (de 1 a 7 – por defeito, Domingo a Sábado).	Weekday(date[,firstdayofweek])
FormatDateTime	Retorna uma expressão data/hora formatada de acordo com preferências.	FormatDateTime(date,format)
FormatNumber	Retorna um valor numérico formato de acordo com preferências.	FormatNumber(Expression[,NurdigAfterDec[,IncLeadingDig[,UseParForNegNum[,GroupDig]]]])
CBool	Retorna uma expressão convertida em Booleano.	CBool(Expression)
CByte	Retorna uma expressão convertida em Inteiro Byte.	CByte(Expression)
CDate	Retorna uma expressão convertida em Data.	CDate(Expression)
CDbl	Retorna uma expressão convertida em Real Duplo.	CDbl(Expression)
CInt	Retorna uma expressão convertida em Inteiro.	CInt(Expression)
CLng	Retorna uma expressão convertida em Inteiro Longo.	CLng(Expression)
CSng	Retorna uma expressão convertida em Real Simples.	CSng(Expression)
CStr	Retorna uma expressão convertida em <i>String</i> .	CStr(Expression)

Asc	Retorna o valor da tabela ASCII de um caracter.	Asc(caracter)
Chr	Retorna o caracter de acordo com a sua numeração na tabela ASCII	Chr(number)
Abs	Retorna o valor absoluto de um valor numérico.	Abs(number)
Int	Retorna apenas a parte inteira de um valor numérico.	Int(number)
Rnd	Retorna um número fraccionário aleatório entre 0 e 1.	Rnd[(number)]
Sqr	Retorna a raiz quadrada de um valor numérico.	Sqr(number)
Array	Retorna um vector contendo a lista dos elementos do seu argumento (o primeiro elemento é o n.º 0).	Array(arglist)
InStr	Devolve a posição da primeira ocorrência de uma <i>string</i> dentro de outra.	InStr([start,]string1,string2)
LCase	Esta função devolve uma string toda convertida em caracteres minúsculos.	LCase(string)
UCase	Esta função devolve uma string toda convertida em caracteres maiúsculos. A sua sintaxe é idêntica à função <i>LCase</i> .	UCase(string)
Len	Esta função devolve o número de caracteres que compõem uma string.	Len(string/variável)
Trim	Esta função devolve uma string retirando-lhe os espaços à esquerda e à direita.	Trim(string)
LTrim, RTrim	Estas funções devolvem uma <i>string</i> sem espaços brancos à esquerda ou à direita, respectivamente.	LTrim(string) / RTrim(string)
Mid	Esta função devolve parte de uma <i>string</i> .	Mid(string,start[,length])
Space	Esta função devolve uma <i>string</i> com determinado número de espaços.	Space(number)
String	Esta função devolve uma <i>string</i> contendo a repetição de determinado caracter determinado número de vezes.	String(number,character)
CreateObject	Esta função permite criar para manipulação um objecto da Microsoft Object Library.	CreateObject(servername.type[me[,location]])
GetLocale	Esta função permite saber a localização geográfica relativa do servidor que provém a ligação Internet ao <i>browser</i> cliente.	GetLocale()
GetRef	Esta função permite conectar uma sub-rotina <i>VBScript</i> a um acontecimento DHTML no documento <i>Web</i> .	Set object.event=GetRef(procname)
InputBox	Esta função permite abrir uma janela de diálogo com uma caixa de introdução de texto para o utilizador, devolvendo o valor por ele introduzido.	InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile,context])

IsEmpty	Esta função permite testar se determinada variável se encontra inicializada ou não, devolvendo <i>False</i> caso não tenha sido ainda atribuído um valor e <i>False</i> em caso contrário.	IsEmpty(variável)
IsNull	Esta função devolve <i>True</i> caso o conteúdo da expressão a testar seja nulo, ou devolve <i>False</i> em caso contrário.	IsNull(expressão)
IsNumeric	Esta função devolve o valor <i>True</i> caso a expressão a avaliar seja um valor numérico, ou <i>False</i> em caso contrário.	IsNumeric(expressão)
MsgBox	Esta função permite abrir uma caixa de diálogo no écran e devolver um valor consoante o botão premido pelo utilizador para fechá-la.	MsgBox (mensagem[,aspecto][,titulo][,hlpfile,context])
Round	Esta função devolve um valor numérico arredondado ao número de casas decimais desejadas.	Round(expressão[,numcasasdec mais])
VarType	Esta função devolve o tipo de valor que a variável a avaliar contém.	VarType(variável)

PALAVRAS-CHAVE DO VBSCRIPT

Keyword	Descrição
Empty	<p>Usado para indicar um valor não inicializado de uma variável. Uma variável não está inicializada quando é criada e ainda não tem nenhum valor no seu conteúdo, ou quando o conteúdo da variável é explicitamente igualada a <i>empty</i>. Exemplo:</p> <pre>dim x 'a variável é criada e não está inicializada! x = "ff" 'a variável x já está inicializada x = empty 'a variável x NÃO está inicializada novamente!</pre> <p>Nota: Isto não é o mesmo que igualar o conteúdo da variável a <i>Null</i>!!</p>
False	Tem um valor igual a 0
Nothing	<p>Utilizado para dissociar um objecto variável, libertando todos os recursos de sistema alocados para ele.</p> <p>Exemplo: set meuObject = Nothing</p>
Null	<p>Utilizado para indicar que a variável não contém dados.</p> <p>Nota: Isto não é o mesmo que igualar o conteúdo da variável a <i>Empty</i>!!</p>
True	Tem um valor igual a -1

EXEMPLO PRÁTICO UTILIZANDO VBSCRIPT

O seguinte exemplo consiste numa listagem de um ficheiro HTML incorporando a utilização de rotinas em *VBScript* para simular uma calculadora utilizando as quatro operações básicas da matemática relativamente a dois valores que o utilizador poderá introduzir.

```
<HTML>
<HEAD>

<script language="VBScript">
Function DadosValidos()
    ok=True
    if (Not IsNumeric(Valor1.value)) or IsNull(Valor1.value) then ok=False
    if (Not IsNumeric(Valor2.value)) or IsNull(Valor2.value) then ok=False
    DadosValidos=ok
End Function

Sub Calculo(operacao)
    r=DadosValidos()
    if r=True Then
        resultado=""
        if operacao="+" then resultado=CStr(CDbl(Valor1.value)+CDbl(Valor2.value))
        if operacao="-" then resultado=CStr(CDbl(Valor1.value)-CDbl(Valor2.value))
        if operacao="*" then resultado=CStr(CDbl(Valor1.value)*CDbl(Valor2.value))
        if operacao="/" then
            if CDbl(Valor2.value)=0 then
                resultado = "Impossível"
            else
                resultado = CStr(CDbl(Valor1.value)/CDbl(Valor2.value))
            end if
        end if
        ValorFinal.value=resultado
    else
        msgbox "Existem valores inválidos!",16,"VALORES INVÁLIDOS"
    end if
End Sub

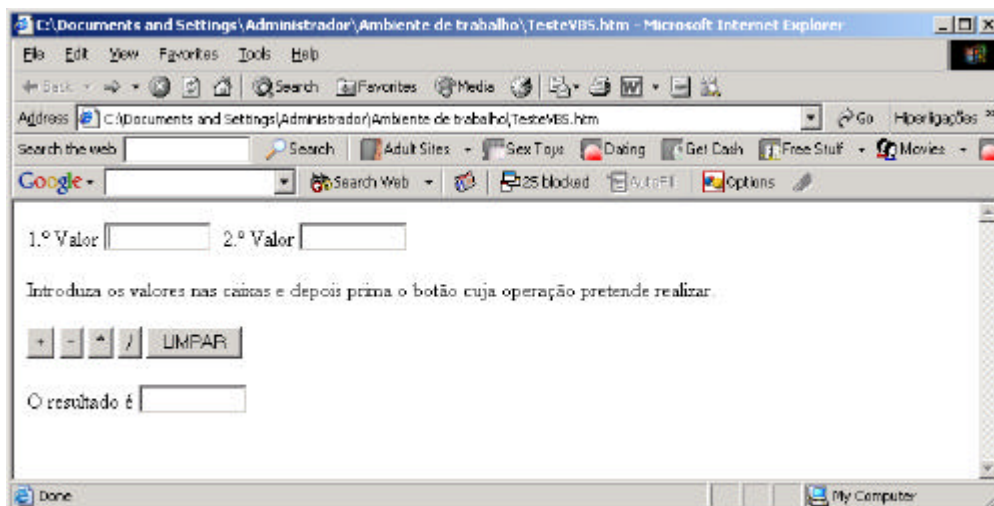
Sub LimpaDados()
    Valor1.value=""
    Valor2.value=""
    ValorFinal.value=""
End Sub
</script>

</HEAD>

<BODY>

1.º Valor <INPUT Name="Valor1" size=10>&nbsp;
2.º Valor <INPUT Name="Valor2" size=10><br><br>
Introduza os valores nas caixas e prima o botão cuja operação pretende
realizar.<br><br>
<INPUT TYPE=BUTTON VALUE=" + " OnClick="Calculo('+')">
<INPUT TYPE=BUTTON VALUE=" - " OnClick="Calculo('-')">
<INPUT TYPE=BUTTON VALUE=" * " OnClick="Calculo('*')">
<INPUT TYPE=BUTTON VALUE=" / " OnClick="Calculo('/')">
<INPUT TYPE=BUTTON VALUE="LIMPAR" OnClick="LimpaDados"><br><br>
O resultado é <INPUT Name="ValorFinal" size=10>

</BODY>
</HTML>
```



Observações: Criaram-se dois objectos *INPUT* do HTML para que o utilizador introduza os dois valores sujeitos a cálculo. Além destes dois objectos do HTML, criaram-se os botões +, -, *, e /, para fazerem o cálculo respectivo sobre os valores introduzidos nos objectos *INPUT* recorrendo a uma sub-rotina em *VBScript*. Foi criado um botão *LIMPAR* para poder apagar os valores introduzidos nos objectos *INPUT*.

Na sub-rotina dos botões de cálculo é executada uma sub-rotina em *VBScript* que se encarrega de verificar se existem valores válidos preenchidos pelo utilizador de forma a poderem ser feitas as contas.

Referências

www.w3schools.com