APOSTILA VBSCRIPT

ActiveX & Scripting O ActiveX Scripting oferece muito mais do que apenas uma linguagem de scripting para navegadores da Web. ActiveX é uma plataforma para desenvolvimento de qualquer quantidade de linguagens de scripting para qualquer finalidade que os desenvolvedores da Web exijam. Usando os serviços de scripting do ActiveX, uma linguagem de scripting pode ser implementada em qualquer plataforma. O ActiveX Scripting é construído a partir de dois componentes principais básicos:

Hosts de Scripting do ActiveX - Os aplicativos em que um scripting é executado.

VBScript em outras Aplicações e Browsers Como um desenvolvedor, você tem licença para usar seus códigos fontes em VBScript em suas aplicações. A Microsoft fornece implementações binárias do VBScript em Windows 16-bits e 32-bits, e para o Macintosh®. VBScript é integrado com browsers da World Wide Web. VBScript e ActiveX Scripting pode também ser usuados como uma línguagem geral em outras aplicações.

Adicionando Códigos do VBScript para uma Página HTML

Você pode usar os elementos de SCRIPT, para adicionar códigos do VBScript em uma página HTML.

A Tag <SCRIPT> Os código do VBScript são escritos dentro da tag <SCRIPT>. Por Exemplo, um procedimento para testar uma data de entrega pôde aparecer como se segue:

```
<SCRIPT LANGUAGE="VBScript">
<!--
Function CanDeliver(Dt)
CanDeliver = (CDate(Dt) - Now()) > 2
End Function
-->
</SCRIPT>
```

Inicia e conclui com a tag <SCRIPT>. O atributo LANGUAGE indica a linguagem de scripting. Você deve especificar a línguagem porque os browsers podem usar outros tipos linguagens de scripting. Note que a função CanDeliver é embutida nas tags de comentário (*<!-- e -->*). Isto previne browsers que não compreende a tag <SCRIPT> de exibir o código.

Você pode incluir o Script na seção HEAD da página:

```
<HTML>
<HEAD>
<TITLE>Place Your Order</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Function CanDeliver(Dt)
CanDeliver = (CDate(Dt) - Now()) > 2
End Function
-->
</SCRIPT>
</HEAD>
<BODY>
...
```

Você pode usar blocos de SCRIPT em qualquer parte de uma página HTML. Você pode colocá-lo na seção BODY e ou HEAD. Entretanto, você provavelmente desejará colocar todo o código de scripting na seção HEAD, com o intuito de organizá-lo. Guardando seu código na seção HEAD você assegura que todo o código está sendo lendo e decodificado antes de qualquer chamadas da seção BODY da página HTML.

Um exceção notável para esta regra é que você pode desejar fornecer código do inline scripting ao responder os eventos de objetos em seu formulário. Por Exemplo, você pode embutir código do scripting para responder a um clique no botão em um formulário:

```
<HTML>
<HEAD>
<TITLE>Test Button Events</TITLE>
</HEAD>
<BODY>
<FORM NAME="Form1">
<INPUT TYPE="Button" NAME="Button1" VALUE="Click">
<SCRIPT FOR="Button1" EVENT="onClick" LANGUAGE="VBScript">
MsgBox "Button Pressed!"
</SCRIPT>
</FORM>
</BODY>
</HTML>
```

A maior parte de seu código aparecerá em procedimentos Sub ou Function, sendo executadas apenas quando forem chamadas. Entretanto, você pode escrever códigos em VBScript fora dos procedimentos, mas ainda dentro um bloco de SCRIPT. Este código é executado apenas uma vez, quando a página HTML for carregada. Isto permite a você iniciar dados ou criar um dinamismo na forma de exibir sua página, enquanto ela é carregada.

Tipos de Dados do VBScript O VBScript tem unicamente um tipo de dado chamado Variant. Um dado Variant é uma espécie especial de tipo de dados que pode conter espécies diferentes de informação, dependendo de como seja usado. Como Variant é o único tipo de dados do VBScript, este será também o tipo de dado retornado pelas funções do VBScript.

Uma variável declarada como Variant pode conter um valor numérico, ou uma cadeia de caracter. Se você está trabalhando com dados que compare números iguais, o VBScript assume que seu tipo de dado é númerico.

Parecidamente, se você está comparando caracteres literais, o VBScript trata seus dados como string. Você pode usar números como strings simplesmente cercando-os com aspas (" ").

Subtipos Variant
Além do tipo numérico ou string, uma Variant podem fazer distinções sobre uma natureza específica de informação numérica. Por Exemplo, você pode ter informação numérica que representa uma data ou um tempo. Certamente, você pode também ter uma variedade rica de informação numérica, essas categorias diferentes de informação que pode ser contida em uma Variant são chamadas subtipos.

A seguinte tabela mostra os subtipos de dados que uma Variant pode conter:

Subtipo	**Descrição**
Descrição Empty (Vazio)	Valor igual a 0 para variáveis numéricas ou ("") zero-comprimento para variáveis string.
Null (Nulo)	Variant contém intencionalmente nenhum valor válido.
Boolean	Contém False (falso) ou True (Verdadeiro) um ou outro.

Byte	Contém inteiro de 0 a 255.
Integer	Contém inteiro de -32,768 a 32,767.
Currency	922,337,203,685,477.5808 a 922,337,203,685,477.5807.
Long	Contém inteiro de -2,147,483,648 a 2,147,483,647.
Single	3.402823E38 a -1.401298E-45 para valores negativos; 1.401298E-45 a 3.402823E38 para valores positivos.
Double	Contém um duplicar-exatidão, boiado-ponto número na série 1.79769313486232E308 para 4.94065645841247E-324 para valores negativos; 4.94065645841247E-324 para 1.79769313486232E308 para valores positivos.
Date (Tempo)	Contém um número que representa uma data entre 1 de Janeiro de 100 até 31 de Dezembro de 9999.
String	Variáveis alfanuméricas, que podem ter um comprimento de 0 até aproximadamente 2 bilhões de caracteres.
Object	Qualquer referência de Objeto.
Error	Contém um número de erro.

Você pode usar funções de conversão para converter dados de um subtipo para o outro. A função VarType retorna informação sobre seus dados, e armazena dentro de uma Variant.

Variáveis do VBScript *O que é uma Variável?*
Uma variável é um nome conveniente que se refere a uma localização de memória do computador, onde você pode armazenar informações de um programa sendo possível alterar seu valor durante o processamento. Por exemplo, você pôde cria uma variável chamada ClickCount para armazenar o número de vezes que um usuário pressionou um objeto em uma página da Web. A forma com que a variável é armazenada na memória do computador é sem importância. O que é importante é que para alterar ou atribuir um valor para essa variável você deve referenciá-la pelo seu nome. No VBScript, as variáveis são sempre tipos de dados Variant.

Declarando Variáveis Você declara variáveis explicitamente em seu script usando a declaração Dim, a declaração Public, e a declaração Private.

Por exemplo:

Dim DegreesFahrenheit

Você pode declarar variáveis múltiplas separando-as por vírgula. Por exemplo:

Dim Top, Bottom, Left, Right

Você pode também declarar uma variável implicitamente simplesmente usando seu nome no script. O que geralmente não é uma boa prática causando resultados inesperados no decorrer de seu script. Uma ótima alternativa é usar a declaração Option Explicit com a finalidade de forçar todas as declarações de variáveis. A declaração Option Explicit deve ser a primeira declaração em seu script.

Especificando Restrições Os nomes de variáveis seguem as regras padronizadas para serem identificadas pelo VBScript.

Um nome de variável deve:

Ter o primeiro caracter do nome da variável sendo uma letra Não deve exceder 255 caracteres.

Escopo e Existência de Variáveis O escopo de uma variável é determinado quando você a declara. Quando você declara uma variável dentro de um procedimento, apenas o código dentro daquele procedimento pode acessar ou muda o valor daquela variável. Isto é um escopo local e é chamado de variável a nível-procedimento. Se você declara uma variável exteriomente a um procedimento, você faz com que todos os procedimentos reconheçam aquela variável, isto é uma variável a nível-escrita.

Uma variável pública é apenas destruída quando ocorre o término do script. No caso da variável privada, ou seja declarada dentro de um procedimento, sua destruição ocorre com o término da execução do procedimento. As variáveis locais são usadas para uma determinada tarefa temporária, liberando assim espaço de memória. Você pode declarar variáveis locais com o mesmo nome em vários procedimentos diferentes pois, elas são apenas visíveis no momento da execução do procedimento.

Designando Valores para Variáveis

Valores são designados para variáveis criando uma expressão do tipo: a variável encontra-se do lado esquerdo da expressão, e o valor que você deseja atribuir no lado direito. Por exemplo:
B = 200

Variáveis Scalar e Variáveis de Array
Em alguns casos você apenas necessita designar um único valor para uma variável. Uma variável contendo um único valor é chamada de scalar.

Entretanto é conveniente designar mais de um valor relacionado apenas a uma variável. Neste caso você pode criar uma variável que contenha uma série de valores. Essa variável é chamada de array. A declaração de uma variável de array é feita dando-se um nome seguindo os parêntesis ( ). No seguinte exemplo, é declarado um array contendo 11 elementos:
Dim Vetor(10)

Embora o número mostrado nos parêntesis seja 10, todos os arrays no VBScript são iniciados com base zero, assim este array realmente contém 11 elementos.

Você referencia cada dado de um elemento de array usando um índice. Começando com o zero e finalizando em 10, os dados podem ser atribuidos aos elementos de um array como se segue:
A(0) = 256
A(1) = 324
A(2) = 100
. . .

A(10) = 55

Os dados podem ser recuperados de qualquer elemento usando um índice dentro do elemento de array. Por exemplo:
. . .

SomeVariable = A(8)
. . .

Arrays não são limitados para uma única dimensão. Você pode ter 60 dimensões, embora a maioria das pessoas não compreendem mais que três ou quatro dimensões. Dimensões múltiplas são declaradas com números relativos a sua dimensão, separados por vírgula entre parêntesis. No seguinte exemplo, a variável MyTable possui duas dimensões consistindo de 6 filas e 11 colunas:

Dim MyTable(5, 10)

Em um array de duas dimensões, o primeiro número será sempre o número de linhas; e o segundo número, o número de colunas.

Você pode também declarar um array cujo tamanho é alterado durante o processamento do script. Este array é chamado de array dinâmico. O array é inicialmente declarado dentro de um procedimento usando a declaração Dim ou ReDim. Entretanto, para um array dinâmico, nenhum tamanho ou dimensão é colocado no interior dos parêntesis. Por exemplo:

Dim MyArray()
ReDim AnotherArray()

Para usar um array dinâmico, você deve subseqüentemente usar a declaração ReDim para determinar o número de dimensões e o tamanho de cada dimensão. No seguinte exemplo, ReDim atribue 25 ao tamanho inicial do array dinâmico. Com a declaração ReDim é feito um redimencionamento do array para 30, usando a palavra-chave Preserve para preservar o conteúdo do array.

ReDim MyArray(25)
. . .

ReDim Preserve MyArray(30)

Não há limite para o número de vezes que você pode redimensionar um array dinâmico, mas você deve saber que se o array for redimensionado com uma dimensão menor que a anterior, o conteúdo dos elementos eliminados serão perdidos.

O que é uma Constante? Uma constante é um nome significativo que é recebe um valor númerico ou caracter. O VBScript define um número de constantes intrínsecas. Você pode obter informação sobre essas constantes intrínsecas na Referência da Linguagem VBScript.

Criando Constantes Você pode criar constantes definidas pelo usuário no VBScript usando a declaração Const. Usando a declaração Const, você pode criar constantes strings ou numéricas com nomes significativos que designam seus valores literais. Por exemplo:

Const MinhaString = "Isto é meu minha cadeia de caracter."
Const MinhaIdade = 49

Note que a string literal é cercada entre aspas ou marcas de citação (" "). As aspas são caminho óbvio para diferenciar valores de string de valores numéricos. Datas e valores do tempo são representados tendo o sinal (#) cercando o valor. Por Exemplo:

Const MeuAniversario = #23-8-76#

Você pode desejar adotar um esquema específico para diferenciar constantes de variáveis. Este procedimento evita durante a execução do script, ocorrer um engano e usar constantes como variáveis e vice-versa. Por exemplo, você pôde usar prefixos "vb" ou "con" nos nomes de suas constantes, ou criar sua própria nomeclatura. Diferenciando constantes de variáveis, você elimina a possibilidade de ocorrer um erro ao desenvolver scripts mais complexos.

Operadores do VBScript

O VBScript tem uma série de operadores, incluindo operadores de aritmética, operadores de comparação, operadores de concatenação, e , operadores lógicos.

Precedência do Operador
Quando várias operações ocorrem em uma expressão, cada parte é avaliada e resolvida em uma ordem predeterminada chamada precedência do operador. Você pode usar parêntesis para alterar a ordem de precedência e forçar a avaliação de algumas partes de uma expressão. Operações dentro de parêntesis

são sempre resolvidas primeiro independendo da ordem de resolução dos operadores. Dentro dos parêntesis, entretanto, a ordem de resolução dos operadores é mantida.

Quando expressões contêm operadores de mais de uma categoria, os operadores aritméticos são avaliados primeiros, depois os operadores de comparação, e os operadores lógicos são avaliados por último. Todos os operadores de comparação tem precedências iguais; estes, são avaliados da esquerda-para-direita. Os operadores Lógicos e de Aritmética são avaliados na seguinte ordem.

Aritmética
Comparação
Lógico
Descrição
Símbolo
Descrição
Símbolo
Descrição
Símbolo
Exponenciação (^)
Igualdade (=)
Negação
Lógica
Not
Negação do Unary (-)
Desigualdade (<>)
Conjunção
Lógica
And
Multiplicação (*)
Menor que (<)
Disjunction
Lógico
Or
Divisão (/)
Maior que (>)
Exclusão
Lógica
Xor
Divisão Inteira (\)
Menor que Ou igual a (<=)
Equivalencia
Lógica
Eqv
Aritmética de Módulo
Mod
Maior que ou Igual a (>=)
Implicação
Lógica
Imp
Adição (+)
Equivalencia de Objeto
Is
Subtração (-)
Concatenação de String (&)

Quando ocorrer a multiplicação e a divisão juntamente em uma expressão, cada operação é avaliada da esquerda para direita. Igualmente, quando ocorre a adição e a subtração juntamente em uma expressão,

cada operação é avaliada em ordem da esquerda para direita.

O operador de concatenação de string (&) não é um operador aritmético, mas por convenção tornou-se e na ordem de resolução, ele está acima de todos os operadores de comparação. O operador Is é um operador de comparação de objetos. Ele não compara objetos ou seus valores; ele apenas checa e determina se duas referências de objeto, referem-se ao mesmo objeto.

Controlando a Execução do Programa Você pode controlar o fluxo de seu script com declarações condicionais e declarações do looping. Usando declarações condicionais, você pode escrever código no VBScript que faz decisões e repete ações. As seguintes declarações condicionais são disponíveis no VBScript: *Declaração If...Then...Else*
Declaração Select Case

Fazendo Decisões Usando If...Then...Else
O If...Then...Else é uma declaração usada para avaliar uma condição seja ela Falsa (False) ou Verdadeira (True) e, contando com o resultado, para executar um ou mais comandos. Usualmente a condição é uma expressão que usa um operador de comparação para comparar um valor ou variável com outra. Para informação sobre operadores de comparação, ver Operadores de Comparação. If...Then...Else pode estar identadas em muitos níveis dependendo de sua necessidade.

Executando uma Declarações se a Condição for Verdadeira Para executar unicamente um declaração quando uma condição é Verdadeira, use uma única linha com a sintaxe If...Then...Else. Note que neste exemplo foi omitido a palavra chave Else.

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

Para executar mais de uma linha de código, você deve usar múltiplas-linhas (ou bloco). Esta sintaxe inclue a declaração End If, como mostra o exemplo a seguir:

```
Sub AlertUser(value)
If value = 0 Then
AlertLabel.ForeColor = vbRed
AlertLabel.Font.Bold = True
AlertLabel.Font.Italic = True
End If
End Sub
```

Executa um Conjunto de Instruções se a Condição for Verdadeira e um Outro Conjunto de Instruções se a condição for Falsa.

Você pode usar um If...Then...Else para definir dois blocos de instruções: um bloco para executar se a condição for Verdadeira, e um outro bloco para executar se a condição for Falsa.

```
Sub AlertUser(value)
If value = 0 Then
AlertLabel.ForeColor = vbRed
AlertLabel.Font.Bold = True
AlertLabel.Font.Italic = True
Else
AlertLabel.Forecolor = vbBlack
```

```
AlertLabel.Font.Bold = False
AlertLabel.Font.Italic = False
End If
End Sub
```

Decidindo Entre Várias Alternativas
Uma variação na declaração If...Then...Else permite a você escolher várias alternativas. Somando-se cláusulas ElseIf você tem a possibilidade de expandir a funcionalidade da declaração If...Then...Else controlando o fluxo do programa baseado em difentes possibilidades. Por Exemplo:

```
Sub ReportValue(value)
If value = 0 Then
MsgBox value
ElseIf value = 1 Then
MsgBox value
ElseIf value = 2 then
Msgbox value
Else
Msgbox "Value out of range!"
End If
```

Você pode adicionar muitas cláusulas ElseIf dependendo da sua necessidade durante o fluxo do algorítimo.

Uso extenso das cláusulas ElseIf freqüentemente torna-se incômodo. Um melhor caminho para escolher entre várias alternativas é a declaração Select Case.

Fazendo Decisões com Select Case
A estrutura Select Case fornece uma alternativa para If...Then...ElseIf durante a execução seletiva de um bloco de código dentre vários blocos de código. Uma declaração Select Case fornece funcionalidade semelhante à If...Then...Else, porém, o código torna-se mais legível e eficiente.

Uma estrutura Select Case avalia uma única expressão no topo da estrutura. O resultado da expressão é então comparada com os valores para cada Case da estrutura. Se há uma afirmação verdadeira, o bloco de declarações associadas com aquele Case é executado:

```
Select Case
Document.Form1.CardType.Options(SelectedIndex).Text
Case "MasterCard"
DisplayMCLogo
ValidateMCAccount
Case "Visa"
DisplayVisaLogo
ValidateVisaAccount
Case "American Express"
DisplayAMEXCOLogo
ValidateAMEXCOAccount
Case Else
DisplayUnknownImage
PromptAgain
End Select
```

Note que a estrutura Select Case avalia a expressão uma única vez no topo da estrutura. Em contraste, a estrutura If...Then...ElseIf pode avaliar diferentes expressões para cada declaração ElseIf. Você pode substituir uma estrurura If...Then...ElseIf pôr uma única estrutura Select Case se cada declaração do ElseIf avalia a mesma expressão.

Usando Laços para repetir Códigos O Looping permite a você executar uma seqüência de declarações dependendo de uma condição. Alguns laços repetem as declarações até uma condição ser Falsa; outros repetem declarações até uma condição ser Verdadeira. Há também laços que repetem declarações num número específico de vezes.

As seguintes declarações de looping são disponíveis no VBScript:

Do...Loop:	Permanece no laço, enquanto ou até uma condição ser Verdadeira.
While...Wend:	Permanece no laços enquanto uma condição é Verdadeira.
For...Next:	Usa um contador para executar as declarações num número específico de vezes. For
Each...Next:	Repete um grupo de declarações para cada item de uma coleção ou cada elemento de um array.

Usando Do Loops Você pode usar a declaração Do...Loop para executar um bloco de declarações num número indeterminado de vezes. As declarações são repetidas enquanto a condição for Verdadeira ou até uma condição torna-se Verdadeira

] **Repetindo Declarações Enquanto uma Condição é Verdadeira** Use a palavra-chave While para checar a condição em uma declaração Do...Loop. Você pode checar a condição antes de entrar no laço (como é mostrado no exemplo ChkFirstWhile), ou você pode checar depois de ter entrado no laço no mínimo uma vez (como é mostrado no exemplo ChkLastWhile). No procedimento ChkFirstWhile, se myNum é recebe o valor igual a 9 em vez de 20, o código no interior do laço nunca será executado. No procedimento ChkLastWhile, o código no interior do laço é executado apenas uma vez porque a condição já é Falsa.

```
Sub ChkFirstWhile()
Dim counter, myNum
counter = 0
myNum = 20
Do While myNum > 10
myNum = myNum - 1
counter = counter + 1
Loop
MsgBox "The loop made " & counter & " repetitions."
End Sub
Sub ChkLastWhile()
Dim counter, myNum
counter = 0
myNum = 9
Do
myNum = myNum - 1
counter = counter + 1
Loop While myNum > 10
MsgBox "The loop made " & counter & " repetitions."
End Sub
```

Repetindo uma Declaração Até uma Condição Torna-se Verdadeira Você pode usar a palavra-chave Until de duas maneiras para checar uma condição da declaração Do...Loop.

Você pode checar a condição antes de entrar no laço (como é mostrado no exemplo ChkFirstUntil), ou você pode checar depois de ter entrado no laço no mínimo uma vez (como é mostrado no exemplo

ChkLastUntil).

Enquanto a condição é Falsa, o looping ocorre.

```
Sub ChkFirstUntil()
Dim counter, myNum
counter = 0
myNum = 20
Do Until myNum = 10
myNum = myNum - 1
counter = counter + 1
Loop
MsgBox "The loop made " & counter & " repetitions."
End Sub
Sub ChkLastUntil()
Dim counter, myNum
counter = 0
myNum = 1
Do
myNum = myNum + 1
counter = counter + 1
Loop Until myNum = 10
MsgBox "The loop made " & counter & " repetitions."
End Sub
```

Saindo do Interior de um laço Do...Loop Você pode sair de um laço Do...Loop usando a declaração Exit Do Ocorrerá casos em que você desejará sair do laço em certas situações. No seguinte exemplo, myNum é designado um valor que cria um laço infinito. A declaração If...Then...Else checa a condição, prevenindo a repetição infinita.

```
Sub ExitExample()
Dim counter, myNum
counter = 0
myNum = 9
Do Until myNum = 10
myNum = myNum - 1
counter = counter + 1
If myNum < 10 Then Exit Do
Loop
MsgBox "The loop made " & counter & " repetitions."
End Sub
```

Usando While...Wend
A declaração While...Wend é fornecida no VBScript para familiarizar a linguagem. Entretanto, por falta de flexibilidade na declaração While...Wend, é recomendado que você use a declaração Do...Loop.

Usando For...Next
Você pode usar a declaração For...Next para executar um bloco de declarações num número específico de vezes. É usado uma variável de contador cujo valor é aumentado ou decrescido com cada repetição do laço. Por exemplo, o seguinte procedimento executa 50 vezes um chamado no procedimento MyProc. A declaração For especifica um contador, uma variável x que recebe um valor inicial, precedida de uma palavra-chave To e depois de um valor final. A declaração Next incrementa a variável contador de 1 unidade.

```
Sub DoMyProc50Times()
```

```
Dim x
For x = 1 To 50
MyProc
Next
End Sub
```

Usando a palavra-chave Step, você pode aumentar ou decresce a variável contador pelo valor especificado na declaração Step. No seguinte exemplo, o contador uma variável j é incrementado de 2 unidades. Quando o laço é terminado, a variável total terá a soma de 2, 4, 6, 8, e 10.

```
Sub TwosTotal()
Dim j, total
For j = 2 To 10 Step 2
total = total + j
Next
MsgBox "The total is " & total
End Sub
```

Para descrescer a variável contador, deve-se atribuir um valor negativo para a declaração Step. Você deve especificar um valor final que é menor que o valor inicial. No seguinte exemplo, o contador a variável myNum decrescido der 2 unidades. Quando o laço é terminado, a variável total terá a soma de 16, 14, 12, 10, 8, 6, 4, e 2.

```
Sub NewTotal()
Dim myNum, total
For myNum = 16 To 2 Step -2
total = total + myNum
Next
MsgBox "The total is " & total
End Sub
```

Você pode sair da declaração For...Next antes do contador alcançar seu valor final usando a declaração Exit For. Ocorrerá casos em que você desejará sair do laço em certas situações.

Usando For Each...Next

Um laço For Each...Next é parecido com um laço For...Next. Em vez de de repetir as declarações num número específico de vezes, um laço For Each...Next repete um grupo de declarações para cada item em uma coleção de objetos ou para cada elemento de um array. Este laço é muito útil quando você fazer não sabe quantos elementos estão em uma coleção.

No seguinte exemplo de código HTML, os conteúdos de um objeto Dicionário é usado para colocar texto em várias caixas de texto:

```
<HTML>
<HEAD><TITLE>Forms and Elements</TITLE></HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChange_OnClick
Dim d 'Create a variable
Set d = CreateObject("Scripting.Dictionary")
d.Add "0", "Athens" 'Add some keys and items
d.Add "1", "Belgrade"
d.Add "2", "Cairo"


For Each I in d
```

```
Document.frmForm.Elements(I).Value = D.Item(I)
Next
End Sub
-->
</SCRIPT>
<BODY>
<CENTER>
<FORM NAME="frmForm">

<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Button" NAME="cmdChange" VALUE="Click Here"><p>
</FORM>
</CENTER>
</BODY>
</HTML>
```

Espécies de Procedimentos No VBScript há duas espécies de procedimentos; o procedimento Sub e o procedimento Function.

Procedimentos Sub
Um procedimento Sub é uma série de declarações do VBScript, dentro das declarações Sub e End Sub. Um procedimento Sub não retorna valor, mas você pode tomar argumentos (constantes, variáveis, ou expressões que são passados durante a chamada da Sub). Se um procedimento Sub não tem argumentos, sua declaração deve incluir um conjunto vazio de parêntesis ().

O seguinte procedimento Sub usa duas funções intrínsecas do VBScript, MsgBox e InputBox, para obter do usuário alguma informação. Depois exibe os resultados de um cálculo baseado nas informações. O cálculo é desempenhado em um procedimento Function criado usando o VBScript. O procedimento Function será discutido a seguir.

```
Sub ConvertTemp()
temp = InputBox("Please enter the temperature in degrees F.", 1)
MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub
```

Procedimentos Function Um procedimento Function é uma série de declarações do VBScript dentro das declarações Function e End Function. Um procedimento Function é parecido com um procedimento Sub, com a diferença de você poder retornar um valor. Um procedimento Function pode tomar argumentos (constantes, variáveis, ou expressões que são passadas pelo procedimento de chamada). Se um procedimento Function não tem argumentos, sua declaração deve incluir um conjunto vazio de parêntesis. Uma Function retorna um valor que será atribuido ao nome da função no procedimento de chamada. O tipo do valor retornado por uma Function será sempre Variant.

No seguinte exemplo, a função Celsius calcula a temperatura em graus Celsius do grau Fahrenheit. Quando a função é chamada do procedimento Sub ConvertTemp, uma variável contendo o valor do argumento é passado à função. O resultado do cálculo é voltado ao procedimento chamado e exibido em uma caixa de mensagem.

```
Sub ConvertTemp()
temp = InputBox("Please enter the temperature in degrees F.", 1)
MsgBox "The temperature is " & Celsius(temp) & " degrees C."
```

End Sub
Function Celsius(fDegrees)
*Celsius = (fDegrees - 32) * 5 / 9*
End Function

Obtendo dados dentro de um Procedimento Cada valor é passado dentro de seus procedimentos usando um argumentos. Argumentos servem como "pontes de valor" dos dados que você deseja passar dentro de seu procedimento. Você pode especificar seus argumentos com um nome de uma variável. Quando você cria um procedimento usando uma declaração Sub ou uma declaração Function, os parêntesis devem incluir os nomes dos argumentos. Quaisquer argumentos são colocados no interior desses parêntesis, separados por vírgulas. No seguinte exemplo, fDegrees é uma "ponte de valor" que passa o valor da temperatura a ser calculada na função:

Function Celsius(fDegrees)
*Celsius = (fDegrees - 32) * 5 / 9*
End Function

Para obter o valor dos dados de um procedimento, você deve usar uma Function. Lembre-se, que um procedimento Function pode retornar um valor; e um procedimento Sub não pode.

Usando Procedimentos Sub e Function em seu Código você deve usar um procedimento Function em seu código do lado direito a uma expressão. Por Exemplo:

Temp = Celsius(fDegrees)
ou
MsgBox "A temperatura do Celsius está " & Celsius(fDegrees) & " graus."

Para chamar um procedimento Sub de outro procedimento, você deve apenas referenciar o nome da procedure passando ou não argumentos. A declaração Call não é requerida, mas se você usá-la, deve colocar os argumentos entre parêntesis.

O seguinte exemplo mostra duas chamadas ao procedimento MyProc. Um usa a declaração Call no código; o outro não. Ambos fazer exatamente a mesma coisa.

Call MyProc(firstarg, secondarg)
MyProc firstarg, secondarg

Note que os parêntesis são omitidos na chamada quando a declaração Call não é usada.

O que é Convenções de Código ? Convenções do Coding são sugestões que podem ajudá-lo a escreve seus códigos usando Microsoft Visual Basic Scripting Edition. As convenções do Coding podem inclui os seguintes ítens:
Especificando convenções para objetos, variáveis, e procedimentos, comentando as convenções, formatando textos e indentando diretrizes

A principal razão para usar um conjunto consistente de convenções do coding é padronizar a estrutura e estilo do código de um script, ou atribuir um modo de escrever seus códigos de forma que você e outros possam facilmente ler e compreendem o algorítimo. Usando boas convenções do coding o resultado é legível, preciso, com um código consistente com outras convenções da línguagem.

Constante Convenções de Nomes Versões mais antecipadas de VBScript não tem nenhum mecanismo para criar constantes definidas pelo usuário. Constantes, se forem implementadas como variáveis se distinguem de outras variáveis usando todos caracteres maiúsculos. Palavras múltiplas foram separadas usando o caracter sublinhado (_). Por Exemplo:

USER_LIST_MAX
NEW_LINE
Enquanto isto ainda é um caminho aceitável para indentificar suas constantes, você pode usar uma alternativa especificando agora um esquema verdadeiro na declaração das constantes usando o Const. Esta convenção usa uma mescla de formato em que os nomes das constantes têm um prefixo "con". Por Exemplo:

conYourOwnConstant

Convenção dos Nomes das Variáveis Com o propósitos de consistência, use os seguintes prefixos com nomes descritivos para variáveis em seu código do VBScript.

- *Subtype*
- *Prefixo*
- *Exemplo*
- *Boolean*
- *bln*
- *blnFound*
- *Byte*
- *byt*
- *bytRasterData Date (Time)*
- *dtm*
- *dtmStart*
- *Double*
- *dbl*
- *dblTolerance*
- *Error*
- *err*
- *errOrderNum*
- *Integer*
- *int*
- *intQuantity*
- *Long*
- *lng*
- *lngDistance*
- *Object*
- *obj*
- *objCurrent*
- *Single*
- *sng*
- *sngAverage*
- *String*
- *str*
- *strFirstName*

Escopo Variável
Variáveis deveriam sempre ser definidas com o menor escopo possível. Variáveis do VBScript podem ter o seguinte escopo.

Escopo onde Variável é declarada visibilidade, nível-Procedure, eventos, função, ou procedures Sub Visível na procedure em que foi declarada Nível-Script, seção HEAD de uma página HTML, exteriormente em qualquer procedure Visível em todas as procedures do script.

Prefixos de Escopo de Variáveis
Como o tamanho do Script cresce, assim faz o valor de existência capaz para rapidamente diferencia o escopo de variáveis. Um um-letra prefixo de escopo precedendo o prefixo de tipo fornece este, sem o unduly aumentando o tamanho de nomes variáveis.

Escopo
Prefixo
Exemplo
Nível-Procedure
None
dblVelocity
Nível-Script
sblnCalcInProgress

Variável Descritiva e Nomes De Procedimento

O corpo de uma variável ou nome de uma procedurce deve ser usada para descrever a necessidade de seu propósito. Em resumo, os nomes de procedures deveriam começar com um verbo, tal como IniciarVariáveis ou FecharJanela.

Para termos freqüentes e longos, abreviações padronizadas são recomendadas para minimizar o comprimento do nome. No geral, nomes de variáveis maiores que 32 caracteres podem ser de difíceis leituras. Quando usar abreviações, certifique-se que elas serão consistentes para todo o Script. Por Exemplo, aleatoriamente mudando de Cnt para Conta dentro de um Script pode gerar um erro no seu código.

Objeto Especificando Convenções
As seguintes listas da tabela são convenções recomendadas para objetos que você pode encontrar enquanto programa em VBScript.

Tipo Objeto
Prefixo
Exemplo
3D Panel
pnl
pnlGroup
Animated Button
ani
aniMailBox
Check Box
chk
chkReadOnly
Combo Box, drop-down list
box
cbo
cboEnglish
Command Button
cmd
cmdExit
Common Dialog
dlg
dlgFileOpen
Frame
fra
fraLanguage

Horizontal Scroll Bar
hsb
hsbVolume
Image
img
imgIcon
Label
lbl
lblHelpMessage
Line
lin
linVertical
List Box
lst
lstPolicyCodes
Spin
spn
spnPages
Text Box
txt
txtLastName
Vertical Scroll Bar
vsb
vsbRate
Slider
sld
sldScale

Comentando Convenções de Códigos Todos os procedimentos deveriam começar com um breve comentário descrevendo o que eles fazem. Este comentário não deveria descrever os detalhes de implementação (como: isto faz isto) porque os códigos mudam freqüentemente, resultando num enorme trabalho e perca de tempo ne manutenção de comentários desnecessários, ou comentários errôneos, pior. O código por si mesmo e quaisquer breves comentários necessários descrevem a implementação.

Argumentos passados para um procedimento devem ser descritos quando seu propósito não está óbvio no código. Valores que voltam para funções e variáveis que são mudadas por um procedimento, especialmente através de argumentos por referência, devem ser descritos no começo de cada procedure.

Comentários no cabeçalho da Procedure devem incluir os seguintes títulos. Por Exemplo:

- Título de Seção
- Comentário do Conteúdo
- Propósito
- Que o procedimento faz (não como).
- Suposições
- Lista de qualquer variável externa, controle, ou outro elemento que declare ações nesta procedimento.
- Efeitos
- Lista de efeito do procedimento em cada variável externa, controle, ou outro elemento.
- Entradas
- Explicação de cada argumento que não esteja óbvio. Cada argumento deve ter uma linha separada com os comentários.
- Valores

- Retornados
- Explicação do valor voltado.

Toda declaração de uma importante variável deve incluir um comentário descrevendo o uso e a existência da variável declarada.

Variáveis, controles, e procedimentos devem ter comentários claros e específicos pois, deles resulta a complexidade da implementação. No começo de seu script, você deve incluir um resumo que descreva, enumerando objetos, procedimentos, algorítimos, caixas de diálogo, e outras dependências do sistema. Às vezes um pedaço de pseudocódigo, descrevendo o algorítimo pode ser bastande prestativo.

Formatando Seu Código
O espaço da tela dever ser conservado, enquanto ainda permita formatar seu código refletindo a estrutura lógica do algorítimo. Estes são uns pontos básicos:

Blocos identados padronizados devem ser indentados com quatro espaços.
O resumo do comentário de um procedimento dever ser indentar com um espaço.

```
'*******************************************************
' Purpose:  Locates the first occurrence of a specified user
'           in the UserList array.
' Inputs:   strUserList():   the list of users to be searched.
'           strTargetUser:   the name of the user to search for.
' Returns:  The index of the first occurrence of the strTargetUser
'           in the strUserList array.
'           If the target user is not found, return -1.
'*******************************************************

Function intFindUser (strUserList(), strTargetUser)
        Dim i           ' Loop counter.
        Dim blnFound ' Target found flag
        intFindUser = -1
        i = 0           ' Initialize loop counter
        Do While i <= Ubound(strUserList) and Not blnFound
        If strUserList(i) = strTargetUser Then
                blnFound = True   ' Set flag to True
                intFindUser = i   ' Set return value to loop count
        End If
        i = i + 1       ' Increment loop counter
Loop
End Function


'
```

Nível de Procedimento Descreve declarações localizadas dentro de um procedimento Function ou Sub. Declarações são realizadas primeiro, seguido por designações e outro código do executável.
Por Exemplo:

Sub MySub() ' This statement declares a sub procedure block.
Dim A ' This statement starts the procedure block.
A = "My variable" ' Procedure-level code.
Debug.Print A ' Procedure-level code.
End Sub ' This statement ends a sub procedure block

Controle do ActiveX Um objeto que você coloca em um formulário, para habilitar ou acentuar a interação do usuário com uma página na Web (No caso do VBScript). Controles ActiveX têm eventos e

podem ser incorporados dentro de outros controles. Os controles são arquivos que possuem extensão .OCX.

Objeto do ActiveX
Um objeto que é exibido para outros aplicações ou programado através de ferramentas de interfaces de automação.

Argumento
Uma constante, variável, ou expressão passada para um procedimento.

Array
Um conjunto de seqüências de elementos ordenados, tendo o mesmo tipo de dados. Cada elemento de um array tem um único identificador referenciado por um número (índice). Mudanças feitas para um elemento de um array, não afeta os demais elementos.

Conjunto de Caracter ASCII
American Standard Code for Information Interchange (ASCII) 7-bit caracter atribuídos amplamente para representar letras e símbolos encontrados em um teclado padrão U.S. O conjunto de caracter ASCII é o mesmo dos primeiros 128 caracteres (0–127) do conjunto de caracter ANSI.

Objeto de Automação
Um objeto que é exibido para outros aplicações ou programado através de ferramentas de interfaces de automação.

Comparação bitwise
Uma comparação idêntica de bit-a-bit bits posicionando o bit nas duas expressões numéricas.

Expressão do Boolean
Uma expressão que avalia a condição Falso (False) ou Verdadeiro (True).

Por Referência
Um meio de passar um argumento para um procedimento, usando o endereço, em vez do valor. Isto permite ao procedimento acessar a variável real. Como resultado, o valor real da variável pode ser alterado dentro do procedimento que ela foi passada.

Por Valor
Um meio de passar um argumento para um procedimento, usando o valor, em vez do endereço de memória da variável. Isto permite ao procedimento acessar uma cópia da variável. Como um resultado, o valor real da variável não pode ser alterado dentro do procedimento que ela foi passada.

Código de Caracter
Um número que representa um caracter específico dm um conjunto, tal como o conjunto de caracter da tabela ASCII.

Classe
A definição formal de um objeto. A classe define as propriedades do objeto e os métodos usados para controlar o comportamento do objeto. É possível criar uma classe para um objeto.

Módulo de classe
Um módulo contendo a definição de uma classe (sua propriedade e definições de método).

Coleção
Um objeto que contem um conjunto de objetos relacionados. A posição de um objeto na coleção pode ser alterada, sempre que uma alteração ocorrer na coleção; portanto, a posição de qualquer objeto especifico na coleção pode variar.

Comentário
Texto que contribue na explicação do código facilitando o trabalho do programador. No Visual Basic Scripting Edition, um comentário pode ser feito usando uma apóstofe ('), ou uma palvra chave Rem seguido por um espaço.

Operador de Comparação
Um caracter ou símbolo indicando um relacionamento entre dois ou mais valores ou expressões. Esses operadores incluem menor que (<), menor que ou igual a (<=), maior que (>), maior que ou igual a (>=), diferentel (<>), e igual (=).

Constante
Um nome específico que contém um valor constante para toda a execução de um programa. Constantes podem ser usadas em qualquer parte de seu código. Uma constante pode ser uma string ou número literal, outra constante, ou qualquer combinação que inclua aritmética ou operadores lógicos exceto Is e exponenciação. Por Exemplo:
Const A = "MyString"

Tipos de Dados Cada subtipo do tipo Variant tem uma série especifica de valores:

- Subtype: Série
- Byte: 0 a 255.
- Boolean: False ou True.
- Integer: 32,768 a 32,767.
- Long: 2,147,483,648 a 2,147,483,647.
- Single: 3.402823E38 a -1.401298E-45 para valores negativos; 1.401298E-45 a 3.402823E38 para valores positivos.
- Double: 1.79769313486232E308 a 4.94065645841247E-324 para valores negativos; 4.94065645841247E-324 a 1.79769313486232E308 para valores positivos.
- Currency: 922,337,203,685,477.5808 a 922,337,203,685,477.5807.
- Date: Janeiro 1, 100 a Dezembro 31, 9999, inclusive.
- Object: Qualquer referência de Objeto.
- String: Variáveis alfanuméricas, que podem ter um comprimento de 0 até aproximadamente 2 bilhões de caracteres.

Expressão de Data Qualquer expressão que pode ser interpretada como uma data. Isto inclue qualquer combinação literal de data, números que comparem datas iguais, strings que comparem datas iguais, e datas voltadas de funções. Uma expressão de data é limitada pelo número ou string, em qualquer combinação, limitada 1 Janeiro de 100 até 31 Dezembro de 9999. Datas são armazenadas como partes de um número real. Valores à esquerda do decimal representa a data; valores à direita do decimal representa o tempo. Números negativos representam datas anterior a 30 Dezembro 1899.

Data Literal
Qualquer seqüência de caracteres com um formato válido que é circundado por sinais de número (#). Formatos válidos incluem o formato de data especificado pelos valores localis, ou o formato universal de data. Por Exemplo, #12/31/99# é a data literal que representa Dezembro 31, 1999, onde English-U.S. é o valor da localidade para seu requerimento. No VBScript, o unico formato reconhecido é o US-ENGLISH, apesar da localidade real do usuário. O formato interpretado é mm/dd/yyyy.

Separadores de Data
Caracteres usados para separar o dia, mês, e ano quando o valore da data é formatado.

Vazio (Empty)
Um valor que indica o primeiro valor designado para uma variável. Variáveis Vazias são 0 num contexto numérico, ou zero-comprimento ("") num contexto de string. fio.

Número de Erro
Um número que abrange toda a série de 0 a 65,535, que corresponde ao número do erro refenciado pelo objeto Err. Este número representa uma mensagem de erro particular.

Expressão
Uma combinação de palavras-chave, operadores, variáveis, constantes, número, ou objeto. Uma expressão pode desempenhar um cálculo, manipular caracteres, ou testar dados.

Constante Intrínseca
Uma constante fornecida por uma aplicação. Você não pode desabilitar constantes intrínsecas, e não pode criar uma constante com o mesmo nome da instríseca.

Palavra-Chave
Uma palavra ou símbolo reconhecida pela linguagem VBScript; por exemplo, uma declaração, nome de função, ou operador.

Localidade
O conjunto de informações que corresponde a língua de um país. Uma localidade afeta a funcionalidade de uma aplicação nos itens de valores, conversões, formatos, datas dos locais específicos. Há dois contextos onde a informação da localidade é importante: A localidade de código afeta a linguagem de termos tais como as palavras-chaves, as definições de valores locais como o decimal, formatos de data, e caracteres que classificam pedidos. A localidade do sistema afeta a funcionalidade da aplicação, quando você exibe números ou converte caracteres em data. Você pode modificar a localidade do sistema usando os utilitários do Painel de Controle fornecidos pelo sistema operacional.

Nada (Nothing)
Um valor especial que indica que uma variável de objeto não é mais longa associada com qualquer objeto real.

Nulo (Null)
Um valor indicando que uma variável não contêm dados válidos. Nulo é o resultado de: Uma designação explícita de Nula para uma variável. Qualquer operação entre expressões que contêm valores Nulos.

Expressão Numérica
Qualquer expressão que pode ser avaliada como um número. Elementos da expressão pode incluir qualquer combinação de palavras-chave, variáveis, constantes, e operadores que resultam em um número.

Tipo de Objeto
Um tipo de objeto exposto por uma aplicação, por exemplo, Requerimento, Arquivo. Consulte a documentação das aplicações (Microsoft Excel, Microsoft Project, Microsoft Word) para uma listagem completa dos objetos disponíveis.

Pi
Pi é um valor igual a constante matemática aproximadamente de 3.1415926535897932.

Private
Variáveis que são visíveis apenas ao Script em que elas são declaradas.

Procedimento
Uma seqüência específica de declarações executadas em uma unidade. Por Exemplo, Function e Sub são tipos de procedimentos.

Nível de Procedimento
Descreve declarações localizadas dentro de um procedimento Function ou Sub. Declarações são

realizadas primeiro, seguido por designações e outro código do executável. Por Exemplo:
Sub MySub() ' This statement declares a sub procedure block.
Dim A ' This statement starts the procedure block.
A = "My variable" ' Procedure-level code.
Debug.Print A ' Procedure-level code.
End Sub ' This statement ends a sub procedure block.

Propriedade
Um atributo específico de um objeto. Propriedades definem as características do objeto tais como tamanho, cor, localização na tela, ou o estado de um objeto, tal como habilitado (Enabled) ou desabilitado (Disabled).

Public
Variáveis declaradas Public são visíveis para todos procedimentos em todos módulos de uma aplicação.

Tempo Execução (Run Time)
É momento em que o código está sendo executado. Durante o tempo de execução, você não pode editar o código.

Erro Tempo de Execução
Um erro que ocorre quando código está sendo executado. Um erro em tempo de execução resulta quando uma declaração efetua uma operação inválida.

Escopo
Define a visibilidade de uma variável, procedimento, ou objeto. Por Exemplo, uma variável declarada como Public é visível para todos os procedimentos em todos os módulos. Variáveis declaradas em procedimentos são visíveis unicamente dentro do procedimento que a declarou, perdendo seu valor ao término deste procedimento.

SCODE
Um valor inteiro longo (Long Integer) que é usado para passar informações detalhadas a função API. Os códigos de condição para interfaces de OLE e APIs são definidas no FACILITY_ITF.

Nível de Escrita
Qualquer código exterior a um procedimento é referido como um nível de código.

Seed
Um valor inicial usado para gerar números do pseudo aleatórios. Por exemplo, a declaração Randomize cria um número usado pela função Rnd para criar seqüências de número do pseudo aleatórios.

Comparando Strings
Uma comparação de duas seqüências de caracteres. A menos que especificado na função que fará a comparação, todas comparações de string são do tipo binário.

Expressão com String
Qualquer expressão que avalia para seqüência de caracteres contíguos. Elementos de uma expressão de string podem incluir uma função que retorne uma string, uma string literal, uma constante de string, ou uma variável de string.

Variável
Uma posição de memória para armazenamento de dados que podem ser modificados durante execução do programa. Cada variável tem um nome que a identifica dentro do seu nível de escopo.