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.