VBScript - Guia Rápido

Página anterior Próxima página

VBScript - Visão Geral

VB Script significa **V** isual **B** asic Scripting que forma um subconjunto do Visual Basic for Applications (VBA). O VBA é um produto da Microsoft que está incluído NÃO apenas em outros produtos da Microsoft, como o MS Project e o MS Office, mas também em ferramentas de terceiros, como o AUTO CAD.

Recursos do VBScript

O VBScript é uma linguagem de script leve, que possui um intérprete muito rápido.

O VBScript, na maior parte, não faz distinção entre maiúsculas e minúsculas. Tem uma sintaxe muito simples, fácil de aprender e implementar.

Ao contrário de C ++ ou Java, o VBScript é uma linguagem de script baseada em objetos e NÃO uma linguagem de programação orientada a objetos.

Ele usa **COM (** Component Object Model **)** para acessar os elementos do ambiente em que está sendo executado.

A execução bem-sucedida do VBScript pode acontecer somente se for executada no Host Environment, como o Internet Explorer **(IE)** , o **IIS (** Serviços de Informações da Internet **)** e o **WSH (** Windows Scripting Host **).**

VBscript - Histórico de Versões e Usos

O VBScript foi introduzido pela Microsoft em 1996 e sua primeira versão foi 1.0. A versão estável atual do VBScript é 5.8, que está disponível como parte do IE8 ou do Windows 7. As áreas de uso do VBScript são abundantes e não estão restritas à lista abaixo.

VBScript é usado como uma linguagem de script em uma das ferramentas de teste de automação populares - Quick Test Professional abreviado como **QTP**

Windows Scripting Host, que é usado principalmente pelos administradores do sistema Windows para automatizar o Windows Desktop.

Active Server Pages **(ASP)** , um ambiente de script do lado do servidor para criar páginas da Web dinâmicas que usam o VBScript ou o Java Script.

O VBScript é usado para scripts do lado do cliente no Microsoft Internet Explorer.

Os formulários do Microsoft Outlook geralmente são executados no VBScript; no entanto, a programação em nível de aplicativo depende do VBA (Outlook 2000 em diante).

Desvantagens

O VBscript é usado apenas pelos navegadores do IE. Outros navegadores, como o Chrome, o Firefox DONOT Support VBScript. Portanto, o JavaScript é preferido em relação ao VBScript.

VBScript tem um suporte de linha de comando limitado.

Como não há ambiente de desenvolvimento disponível por padrão, a depuração é difícil.

Onde o VBScript é hoje?

A versão atual do VBScript é 5.8, e com o recente desenvolvimento do .NET framework, a Microsoft decidiu fornecer suporte futuro do VBScript dentro do ASP.NET para desenvolvimento web. Portanto, NÃO haverá mais novas versões do mecanismo VBScript, mas todas as correções de defeitos e problemas de segurança estão sendo resolvidos pela equipe de engenharia de suporte da Microsoft. No entanto, o mecanismo VBScript seria enviado como parte de todo o Microsoft Windows e IIS por padrão.

VBScript - sintaxe

Seu primeiro VBScript

Vamos escrever um VBScript para imprimir "Hello World".

```
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         document.write("Hello World!")
      </script>
   </body>
</html>
```

No exemplo acima, chamamos uma função *document.write* , que grava uma string no documento HTML. Esta função pode ser usada para escrever texto, HTML ou ambos. Então, o código acima exibirá o seguinte resultado -

```
Hello World!
```

Espaço em branco e quebras de linha

VBScript ignora espaços, tabulações e novas linhas que aparecem dentro de programas VBScript. Pode-se usar espaços, abas e novas linhas livremente dentro do programa, para que você esteja livre para formatar e recuar seus programas de uma forma organizada e consistente que facilite a leitura e a compreensão do código.

Formatação

O VBScript é baseado no Visual Basic da Microsoft. Ao contrário do JavaScript, nenhum terminador de instrução, como ponto-e-vírgula, é usado para finalizar uma instrução específica.

Sintaxe de Linha Única

Colons são usados quando duas ou mais linhas de VBScript devem ser escritas em uma única linha. Portanto, no VBScript, os Colons atuam como um separador de linha.

```
<script language = "vbscript" type = "text/vbscript">
   var1 = 10 : var2 = 20
</script>
```

Sintaxe de Linha Múltipla

Quando uma instrução no VBScript é longa e se o usuário desejar dividi-la em várias linhas, o usuário terá que usar o sublinhado "_". Isso melhora a legibilidade do código. O exemplo a seguir ilustra como trabalhar com várias linhas.

```
<script language = "vbscript" type = "text/vbscript">
   var1 = 10
   var2 = 20
   Sum = var1 + var2
   document.write("The Sum of two numbers"&_"var1 and var2 is " & Sum)
</script>
```

Palavras reservadas

A lista a seguir mostra as palavras reservadas no VBScript. Estas palavras reservadas NÃO DEVEM ser usadas como constantes ou variáveis ou quaisquer outros nomes de identificadores.

Loop	LSet	Eu
Mod	Novo	Próximo
Não	Nada	Nulo
Em	Opção	Opcional
Ou	ParamArray	Preservar

Privado	Público	RaiseEvent
ReDim	Rem	Currículo
RSet	Selecione	Conjunto
Compartilhado	solteiro	Estático
Pare	Sub	Então
Para	Verdade	Tipo
E	Como	boleano
ByRef	Byte	ByVal
Ligar	Caso	Classe
Const	Moeda	Depurar
Dim	Faz	em dobro
Cada	Outro	ElseIf
Esvaziar	Fim	Fim se
Enum	Eqv	Evento
Saída	Falso	Para
Função	Pegue	Vamos para
E se	Criança levada	Implementos
Em	Inteiro	É
Deixei	Gostar	Longo
Tipo de	Até	Variante
Wend	Enquanto	Com
Xor	Eval	Executar
Msgbox	Apagar	ExecuteGlobal
Option Explicit	Aleatória	SendKeys

Caso de sensibilidade

O VBScript é uma **linguagem que não diferencia maiúsculas de minúsculas** . Isso significa que palavras-chave de linguagem, variáveis, nomes de funções e quaisquer outros identificadores NÃO precisam ser digitados com uma capitalização consistente de

letras. Então os identificadores int_counter, INT_Counter e INT_COUNTER têm o mesmo significado dentro do VBScript.

Comentários no VBScript

Os comentários são usados para documentar a lógica do programa e as informações do usuário com as quais outros programadores podem trabalhar perfeitamente no mesmo código no futuro. Pode incluir informações tais como desenvolvidas por, modificadas por e também podem incluir lógica incorporada. Comentários são ignorados pelo intérprete durante a execução. Comentários no VBScript são denotados por dois métodos.

1. Qualquer declaração que comece com um Single Quote (') é tratada como comentário.

A seguir está o exemplo -

```
<script language = "vbscript" type = "text/vbscript">
   <!–
      ' This Script is invoked after successful login
      ' Written by : TutorialsPoint
      ' Return Value : True / False
   //- >
</script>
```

2. Qualquer declaração que comece com a palavra-chave “REM”.

A seguir está o exemplo -

```
<script language = "vbscript" type = "text/vbscript">
   <!–
      REM This Script is written to Validate the Entered Input
      REM Modified by  : Tutorials point/user2
   //- >
</script>
```

Ativando o VBScript em navegadores

Nem todos os navegadores modernos suportam VBScript. O VBScript é suportado apenas pelo Internet Explorer da Microsoft, enquanto outros navegadores (Firefox e Chrome) suportam apenas JavaScript. Portanto, os desenvolvedores normalmente preferem JavaScript ao VBScript.

Embora o Internet Explorer (IE) suporte o VBScript, talvez seja necessário ativar ou desativar esse recurso manualmente. Este tutorial fará com que você conheça o procedimento de habilitar e desabilitar o suporte a VBScript no Internet Explorer.

VBScript no Internet Explorer

Aqui estão os passos simples para ativar ou desativar o VBScript no seu Internet Explorer -

Siga as Ferramentas → Opções da Internet no menu

Selecione a guia Segurança na caixa de diálogo.

Clique no botão Nível personalizado

Role para baixo até encontrar a opção Scripting

Selecione o botão de opção Ativar em script ativo

Por fim, clique em OK e saia

Para desabilitar o suporte a VBScript em seu Internet Explorer, você precisa selecionar o botão de opção *Desativar* em **Script ativo** .

VBScript - canais

Posicionamento VBScript no arquivo HTML

Há uma flexibilidade para incluir código VBScript em qualquer lugar em um documento HTML. Mas a maneira mais preferida de incluir o VBScript em seu arquivo HTML é a seguinte -

Script na seção <head> ... </ head>.

Script na seção <body> ... </ body>.

Script nas seções <body> ... </ body> e <head> ... </ head>.

Script em um arquivo externo e, em seguida, incluir na seção <head> ... </ head>.

Na próxima seção, veremos como podemos colocar o VBScript de maneiras diferentes -

VBScript na seção <head> ... </ head>

Se você quiser que um script seja executado em algum evento, por exemplo, quando um usuário clicar em algum lugar, você colocará esse script na cabeça da seguinte maneira:

```
<html>
   <head>
      <script type = "text/Vbscript">
         <!--
            Function sayHello()
               Msgbox("Hello World")
            End Function
         //-->
      </script>
   </head>

   <body>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
   </body>
</html>
```

Ele produzirá o seguinte resultado - um botão com o nome SayHello. Ao clicar no botão, a caixa de mensagem é exibida para o usuário com a mensagem "Hello World".

```
Say Hello
```

VBScript na seção <body> ... </ body>

Se você precisar de um script para ser executado enquanto a página é carregada para que o script gere conteúdo na página, o script entra na parte <body> do documento. Neste caso, você não teria nenhuma função definida usando VBScript -

```
<html>
   <head> </head>
   <body>
      <script type = "text/vbscript">
         <!--
            document.write("Hello World")
         //-->
      </script>
      <p>This is web page body </p>
   </body>
</html>
```

Isso produzirá o seguinte resultado -

```
Hello World
This is web page body
```

VBScript nas seções <body> e <head>

Você pode colocar seu código VBScript na seção <head> e <body> como segue -

```
<html>
   <head>
      <script type = "text/vbscript">
         <!--
            Function sayHello()
               msgbox("Hello World")
            End Function
         //-->
      </script>
   </head>

   <body>
      <script type = "text/vbscript">
         <!--
         document.write("Hello World")
         //-->
      </script>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
   </body>
</html>
```

Ele produzirá o seguinte resultado - mensagem Hello World com um botão 'Say Hello'. Ao clicar no botão, uma caixa de mensagem com a mensagem "Hello World" é exibida ao usuário.

Hello World Say Hello

VBScript em arquivo externo

À medida que você começa a trabalhar mais extensivamente com o VBScript, provavelmente encontrará casos em que está reutilizando código VBScript idêntico em várias páginas de um site. Você não está restrito a manter código idêntico em vários arquivos HTML.

A tag de *script* fornece um mecanismo para permitir que você armazene o VBScript em um arquivo externo e, em seguida, inclua-o em seus arquivos HTML. Aqui está um exemplo para mostrar como você pode incluir um arquivo VBScript externo em seu código HTML usando a tag *script* e seu atributo *src* -

```
<html>
   <head>
      <script type = "text/vbscript" src = "filename.vbs" ></script>
   </head>
   <body>
      .......
   </body>
</html>
```

Para usar o VBScript a partir de uma fonte de arquivo externa, você precisa escrever o código-fonte do VBScript em um arquivo de texto simples com a extensão ".vbs" e incluir esse arquivo como mostrado acima. Por exemplo, você pode manter o seguinte conteúdo no arquivo filename.vbs e, em seguida, usar a função *sayHello* no arquivo HTML depois de incluir o arquivo filename.vbs.

```
Function sayHello()
   Msgbox "Hello World"
End Function
```

Posicionamento VBScript no QTP

O VBScript é colocado na ferramenta QTP (Quick Test Professional), mas não é incluído nas tags HTML. O arquivo de script é salvo com a extensão .vbs e é executado pelo mecanismo de execução do Quick Test Professional.

VBScript - Variáveis

Variáveis VBScript

Uma variável é um local de memória nomeado usado para manter um valor que pode ser alterado durante a execução do script. O VBScript possui apenas **UM** tipo de dado fundamental, **Variant** .

Regras para Declarar Variáveis -

- O nome da variável deve começar com um alfabeto.

Os nomes das variáveis não podem exceder 255 caracteres.

Variáveis NÃO devem conter um ponto (.)

Os nomes das variáveis devem ser únicos no contexto declarado.

Declarando Variáveis

As variáveis são declaradas usando a palavra-chave "dim". Como existe apenas UM tipo de dado fundamental, todas as variáveis declaradas são variantes por padrão. Portanto, um usuário **NÃO precisa** mencionar o tipo de dados durante a declaração.

Exemplo 1 - Neste exemplo, IntValue pode ser usado como um string, inteiro ou mesmo matrizes.

```
Dim Var
```

Exemplo 2 - Duas ou mais declarações são separadas por vírgula (,)

```
Dim Variable1,Variable2
```

Atribuindo Valores às Variáveis

Valores são atribuídos de forma semelhante a uma expressão algébrica. O nome da variável no lado esquerdo seguido por um símbolo igual a (=) e depois o seu valor no lado direito.

Regras

Os valores numéricos devem ser declarados sem aspas duplas.

Os valores de String devem ser colocados entre aspas duplas (")

As variáveis de data e hora devem estar entre o símbolo de hash (#)

Exemplos

```
' Below Example, The value 25 is assigned to the variable.
Value1 = 25

' A String Value 'VBScript' is assigned to the variable StrValue.
StrValue = "VBScript"

' The date 01/01/2020 is assigned to the variable DToday.
Date1 = #01/01/2020#

' A Specific Time Stamp is assigned to a variable in the below example.
Time1 = #12:30:44 PM#
```

Escopo das Variáveis

Variáveis podem ser declaradas usando as seguintes instruções que determinam o escopo da variável. O escopo da variável desempenha um papel crucial quando usado dentro de um procedimento ou classes.

Dim

Público

Privado

Dim

As variáveis declaradas usando a palavra-chave "Dim" em um nível de Procedimento estão disponíveis apenas no mesmo procedimento. Variáveis declaradas usando a palavra-chave "Dim" no nível de script estão disponíveis para todos os procedimentos dentro do mesmo script.

Exemplo - No exemplo abaixo, o valor de Var1 e Var2 é declarado no nível do script enquanto Var3 é declarado no nível do procedimento.

Nota - O escopo deste capítulo é entender Variáveis. As funções seriam tratadas em detalhes nos próximos capítulos.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim Var1
         Dim Var2

         Call add()
         Function add()
            Var1 = 10
            Var2 = 15
            Dim Var3
            Var3 = Var1 + Var2
            Msgbox Var3 'Displays 25, the sum of two values.
         End Function

         Msgbox Var1   ' Displays 10 as Var1 is declared at Script level
         Msgbox Var2   ' Displays 15 as Var2 is declared at Script level
         Msgbox Var3   ' Var3 has No Scope outside the procedure. Prints Empty
      </script>
   </body>
</html>
```

Público

As variáveis declaradas usando palavras-chave "públicas" estão disponíveis para todos os procedimentos em todos os scripts associados. Ao declarar uma variável do tipo "public", a palavra-chave Dim é substituída por "Public".

Exemplo - No exemplo a seguir, Var1 e Var2 estão disponíveis no nível de script, enquanto Var3 está disponível nos scripts e procedimentos associados, conforme declarado como Público.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim Var1
         Dim Var2
```

```
         Public Var3

         Call add()

         Function add()
            Var1 = 10
            Var2 = 15
            Var3 = Var1+Var2
            Msgbox Var3 'Displays 25, the sum of two values.
         End Function

         Msgbox Var1   ' Displays 10 as Var1 is declared at Script level
         Msgbox Var2   ' Displays 15 as Var2 is declared at Script level
         Msgbox Var3   ' Displays 25 as Var3 is declared as Public

      </script>
   </body>
</html>
```

Privado

Variáveis que são declaradas como "Particulares" têm escopo somente dentro desse script no qual elas são declaradas. Ao declarar uma variável do tipo "Privado", a palavra-chave Dim é substituída por "Particular".

Exemplo - No exemplo a seguir, Var1 e Var2 estão disponíveis no nível de script. Var3 é declarado como Particular e está disponível apenas para este script específico. O uso de variáveis "Privadas" é mais pronunciado dentro da Classe.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim Var1
         Dim Var2
         Private Var3

         Call add()
         Function add()
            Var1 = 10
            Var2 = 15
            Var3 = Var1+Var2
            Msgbox Var3 'Displays the sum of two values.
         End Function

         Msgbox Var1   ' Displays 10 as Var1 is declared at Script level
         Msgbox Var2   ' Displays 15 as Var2 is declared at Script level
         Msgbox Var3   ' Displays 25 but Var3 is available only for this script.
      </script>
   </body>
</html>
```

VBScript - Constantes

Constante é uma localização de memória nomeada usada para manter um valor que NÃO PODE ser alterado durante a execução do script. Se um usuário tentar alterar um valor constante, a execução do script terminará com um erro. Constantes são declaradas da mesma forma que as variáveis são declaradas.

Constantes declarantes

Sintaxe

```
[Public | Private] Const Constant_Name = Value
```

A constante pode ser do tipo público ou privado. O uso de público ou privado é opcional. As constantes públicas estão disponíveis para todos os scripts e procedimentos, enquanto as constantes particulares estão disponíveis no procedimento ou na classe. Pode-se atribuir qualquer valor, como número, string ou data, à constante declarada.

Exemplo 1

Neste exemplo, o valor de pi é 3.4 e exibe a área do círculo em uma caixa de mensagem.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim intRadius
         intRadius = 20
         const pi = 3.14
         Area = pi*intRadius*intRadius
         Msgbox Area

      </script>
   </body>
</html>
```

Exemplo 2

O exemplo abaixo ilustra como atribuir um valor de string e data a uma constante.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Const myString = "VBScript"
         Const myDate = #01/01/2050#
         Msgbox myString
         Msgbox myDate

      </script>
   </body>
</html>
```

Exemplo 3

No exemplo abaixo, o usuário tenta alterar o valor constante; daqui, terminará com um **erro de execução.**

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim intRadius
         intRadius = 20
```

```
         const pi = 3.14
         pi = pi*pi     'pi VALUE CANNOT BE CHANGED.THROWS ERROR'
         Area = pi*intRadius*intRadius
         Msgbox Area

      </script>
   </body>
</html>
```

VBScript - Operadores

O que é um operador?

Vamos tomar uma expressão *4 + 5 é igual a 9* . Aqui, 4 e 5 são chamados **operandos** e + é chamado de **operador** . A linguagem VBScript suporta os seguintes tipos de operadores -

- Operadores aritméticos
- Operadores de Comparação
- Operadores lógicos (ou relacionais)
- Operadores de Concatenação

Os operadores aritméticos

O VBScript suporta os seguintes operadores aritméticos -

Suponha que a variável A mantenha 5 e a variável B tenha 10, então -

Mostrar exemplos

Operador	Descrição	Exemplo
+	Adiciona dois operandos	A + B dará 15
-	Subtrai o segundo operando do primeiro	A - B vai dar -5
*	Multiplique os dois operandos	A * B dará 50
/	Divida o numerador por denumerador	B / A vai dar 2
%	Operador de módulo e resto depois de uma divisão inteira	B MOD A dará 0
^	Operador de Exponenciação	B ^ A dará 100000

Para entender melhor esses operadores, experimente você mesmo .

Os operadores de comparação

Existem os seguintes operadores de comparação suportados pela linguagem VBScript -

Suponha que a variável A detenha 10 e a variável B detenha 20, então -

Mostrar exemplos

Operador	Descrição	Exemplo
=	Verifica se o valor de dois operandos é igual ou não, se sim, a condição se torna verdadeira.	(A == B) é falso.
<>	Verifica se o valor de dois operandos é igual ou não, se os valores não forem iguais, a condição se tornará verdadeira.	(A <> B) é verdadeiro.
>	Verifica se o valor do operando esquerdo é maior que o valor do operando direito, se sim, a condição se torna verdadeira.	(A> B) é falso.
<	Verifica se o valor do operando esquerdo é menor que o valor do operando direito, se sim, a condição se torna verdadeira.	(A <B) é verdadeiro.
> =	Verifica se o valor do operando esquerdo é maior ou igual ao valor do operando direito, se sim, a condição se torna verdadeira.	(A> = B) é Falso.
<=	Verifica se o valor do operando esquerdo é menor ou igual ao valor do operando direito, se sim, a condição se torna verdadeira.	(A <= B) é verdadeiro.

Para entender melhor esses operadores, experimente você mesmo .

Os operadores lógicos

Existem os seguintes operadores lógicos suportados pela linguagem VBScript -

Suponha que a variável A detenha 10 e a variável B tenha 0, então -

Mostrar exemplos

Operador	Descrição	Exemplo
E	Chamado Lógico E operador. Se ambas as condições forem verdadeiras, a expressão se tornará verdadeira.	um <> 0 AND b <> 0 é Falso.
OU	Chamado Lógico OU Operador. Se alguma das duas condições for True, a condição se tornará True.	um <> 0 OU b <> 0 é verdadeiro.
NÃO	Operador NÃO Lógico Chamado. Ele inverte o estado lógico de seu operando. Se uma condição for True, o operador Lógico NOT fará False.	NOT (a <> 0 OR b <> 0) é falso.
XOR	Exclusão Lógica Chamada. É a combinação de NOT e OR Operator. Se uma e somente uma das expressões for avaliada como True, o resultado será True.	(a <> 0 XOR b <> 0) é verdadeiro.

Para entender melhor esses operadores, experimente você mesmo .

Os Operadores de Concatenação

Existem os seguintes operadores de Concatenação suportados pela linguagem VBScript -

Suponha que a variável A detenha 5 e a variável B detenha 10 então -

Mostrar exemplos

Operador	Descrição	Exemplo
+	Adiciona dois valores como valores variáveis são numéricos	A + B dará 15
E	Concatena dois valores	A & B vai dar 510

Suponha que a variável A = "Microsoft" e a variável B = "VBScript", então -

Operador	Descrição	Exemplo
+	Concatena dois valores	A + B dará o MicrosoftVBScript
E	Concatena dois valores	A & B dará o MicrosoftVBScript

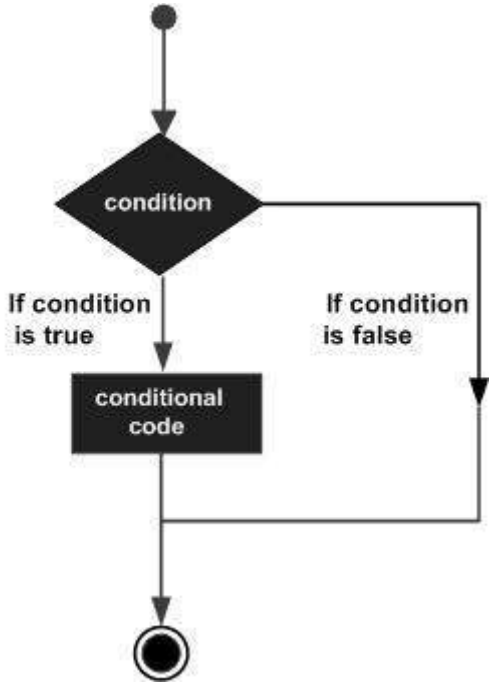
Nota - Operadores de Concatenação podem ser usados para números e strings. A Saída depende do contexto se as variáveis contiverem valores numéricos ou Valor da Sequência.

Para entender melhor esses operadores, você pode experimentar você mesmo .

VBScript - Tomada de Decisão

A tomada de decisão permite que os programadores controlem o fluxo de execução de um script ou de uma de suas seções. A execução é governada por uma ou mais instruções condicionais.

A seguir está a forma geral de uma estrutura de tomada de decisão típica encontrada na maioria das linguagens de programação -



VBScript fornece os seguintes tipos de declarações de tomada de decisão.

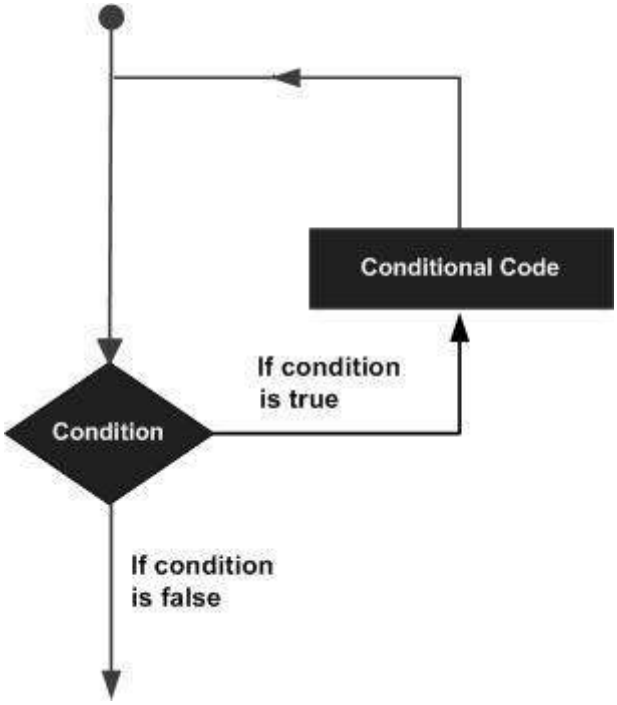
Declaração	Descrição
se declaração	Uma instrução **if** consiste em uma expressão booleana seguida de uma ou mais instruções.
if..else statement	Uma declaração **if else** consiste em uma expressão booleana seguida de uma ou mais instruções. Se a condição for True, as instruções sob as instruções **If** serão executadas. Se a condição é falsa, então a parte **Else** do script é Executada
if ... elseif..else statement	Uma instrução **if** seguida por uma ou mais instruções **ElseIf** , que consistem em expressões booleanas e, em seguida, seguidas por uma instrução else opcional, que é executada quando todas as condições se tornam falsas.
aninhado se instruções	Um **caso** ou **elseif** declaração dentro de outro **se** ou **elseif** declaração (s).
mudar a indicação	Uma instrução **switch** permite que uma variável seja testada quanto à igualdade em relação a uma lista de valores.

VBScript - Loops

Pode haver uma situação em que você precise executar um bloco de código várias vezes. Em geral, as instruções são executadas seqüencialmente: a primeira instrução em uma função é executada primeiro, seguida pela segunda e assim por diante.

As linguagens de programação fornecem várias estruturas de controle que permitem caminhos de execução mais complicados. Uma instrução de loop nos permite executar

uma instrução ou grupo de instruções várias vezes e seguir é o geral de uma instrução de loop no VBScript.



O VBScript fornece os seguintes tipos de loops para lidar com os requisitos de loop. Clique nos links a seguir para verificar seus detalhes.

Tipo de loop	Descrição
para loop	Executa uma seqüência de instruções várias vezes e abrevia o código que gerencia a variável de loop.
para .. cada ciclo	Ele é executado se houver pelo menos um elemento no grupo e reiterado para cada elemento em um grupo.
while..wend loop	Ele testa a condição antes de executar o corpo do loop.
loops while..while	As instruções do..While serão executadas contanto que a condição seja True (ou seja,) O Loop deve ser repetido até que a condição seja False.
fazer .. até loops	As instruções do..Until serão executadas desde que a condição seja False (ou seja,) O Loop deve ser repetido até que a condição seja True.

Declarações de controle de loop

As instruções de controle de loop alteram a execução de sua sequência normal. Quando a execução deixa um escopo, todas as instruções restantes no loop NÃO são executadas.

O VBScript suporta as seguintes instruções de controle. Clique nos links a seguir para verificar seus detalhes.

Declaração de controle	Descrição
Exit For statement	Encerra a instrução **For loop** e transfere a execução para a instrução imediatamente após o loop
Declaração Exit Do	Encerra a instrução **Do While** e transfere a execução para a instrução imediatamente após o loop

VBScript - Eventos

O que é um evento?

A interação do VBScript com HTML é tratada por meio de eventos que ocorrem quando o usuário ou navegador manipula uma página. Quando a página é carregada, isso é um evento. Quando o usuário clica em um botão, esse clique também é um evento. Outros exemplos de eventos incluem pressionar qualquer tecla, janela de fechamento, janela de redimensionamento, etc. Os desenvolvedores podem usar esses eventos para executar respostas codificadas VBScript, que fazem com que os botões fechem janelas, mensagens sejam exibidas aos usuários, dados a serem validados e virtualmente qualquer outro tipo de resposta que se possa imaginar.

Os eventos fazem parte do DOM (Document Object Model) e cada elemento HTML possui um determinado conjunto de eventos, que podem acionar o código VBScript. Por favor, vá através deste pequeno tutorial para uma melhor compreensão HTML Event Reference . Aqui, veremos alguns exemplos para entender uma relação entre o evento e o VBScript.

onclick Tipo de Evento

Esse é o tipo de evento mais usado, que ocorre quando um usuário clica no botão esquerdo do mouse. Você pode colocar sua validação, aviso, etc., neste tipo de evento.

Exemplo

```
<html>
   <head>
      <script language = "vbscript" type = "text/vbscript">
         Function sayHello()
            msgbox "Hello World"
         End Function
      </script>
   </head>

   <body>
      <input type = "button" onclick = "sayHello()" value = "Say Hello"/>
   </body>
</html>
```

Ele produzirá o resultado a seguir e, quando você clicar no botão Hello, ocorrerá o evento onclick, que acionará a função sayHello ().

Say Hello

tipo de evento onsubmit

Outro tipo de evento mais importante é o *onsubmit* . Este evento ocorre quando você tenta enviar um formulário. Então você pode colocar a validação do seu formulário contra esse tipo de evento. O formulário é enviado clicando no botão Enviar, a caixa de mensagem é exibida.

O formulário é enviado clicando no botão Enviar, a caixa de mensagem é exibida.

Exemplo

```
<html>
   <head> </head>
   <body>
      <script language = "VBScript">
         Function fnSubmit()
            Msgbox("Hello Tutorialspoint.Com")
         End Function
      </script>

      <form action = "/cgi-bin/test.cgi" method = "post" name = "form1" onSubmit = "fnSubmit()
         <input name = "txt1" type = "text"><br>
         <input name = "btnButton1" type = "submit" value="Submit">
      </form>
   </body>
</html>
```

onmouseover e onmouseout

Esses dois tipos de eventos ajudarão você a criar efeitos interessantes com imagens ou até mesmo com texto. O evento *onmouseover* ocorre quando você passa o mouse sobre qualquer elemento e o *onmouseout* ocorre quando você tira o mouse desse elemento.

Exemplo

```
<html>
   <head> </head>
   <body>
      <script language = "VBScript">
         Function AlertMsg
            Msgbox("ALERT !")
         End Function

         Function onmourse_over()
            Msgbox("Onmouse Over")
         End Function

         Sub txt2_OnMouseOut()
            Msgbox("Onmouse Out !!!")
         End Sub
```

```
         Sub btnButton_OnMouseOut()
            Msgbox("onmouse out on Button !")
         End Sub
      </script>

      <form action = "page.cgi" method = "post" name = "form1">
         <input name = "txt1" type = "text" OnMouseOut = "AlertMsg()"><br>
         <input name = "txt2" type = "text" OnMouseOver = "onmourse_over()">
         <br><input name = "btnButton" type = "button" value = "Submit">
      </form>
   </body>
</html>
```

Ele produzirá um resultado quando você passar o mouse sobre a caixa de texto e também quando você afastar o foco da caixa de texto e do botão.

Eventos padrão HTML 4

Os eventos padrão do HTML 4 estão listados aqui para sua referência. Aqui, o script indica uma função VBScript a ser executada contra esse evento.

Evento	Valor	Descrição
em mudança	roteiro	O script é executado quando o elemento é alterado
onsubmit	roteiro	Script é executado quando o formulário é enviado
onreset	roteiro	Script é executado quando o formulário é redefinido
onblur	roteiro	Script é executado quando o elemento perde o foco
No foco	roteiro	Script é executado quando o elemento recebe foco
onkeydown	roteiro	Script é executado quando a tecla é pressionada
onkeypress	roteiro	Script é executado quando a tecla é pressionada e liberada
onkeyup	roteiro	Script é executado quando a chave é liberada
onclick	roteiro	Script é executado quando um clique do mouse
ondblclick	roteiro	Script é executado quando um mouse clica duas vezes
onmousedown	roteiro	Script é executado quando o botão do mouse é pressionado
onmousemove	roteiro	Script é executado quando o ponteiro do mouse se move
onmouseout	roteiro	Script é executado quando o ponteiro do mouse sai de um elemento
onmouseover	roteiro	Script é executado quando o ponteiro do mouse se move sobre um elemento
onmouseup	roteiro	Script é executado quando o botão do mouse é liberado

VBScript e Cookies

O que são cookies?

Os navegadores e servidores da Web usam o protocolo HTTP para se comunicar e o HTTP é um protocolo sem estado. Mas para um site comercial, é necessário manter as informações da sessão entre diferentes páginas. Por exemplo, um registro de usuário termina após a conclusão de várias páginas. Mas como manter as informações da sessão do usuário em todas as páginas da web. Em muitas situações, o uso de cookies é o método mais eficiente de lembrar e rastrear preferências, compras, comissões e outras informações necessárias para uma melhor experiência do visitante ou estatísticas do site.

Como funciona?

Seu servidor envia alguns dados para o navegador do visitante na forma de um cookie. O navegador pode aceitar o cookie. Em caso afirmativo, ele é armazenado como um registro de texto simples no disco rígido do visitante. Agora, quando o visitante chega em outra página em seu site, o navegador envia o mesmo cookie para o servidor para recuperação. Uma vez recuperado, seu servidor sabe / lembra o que foi armazenado anteriormente. Cookies são um registro de dados de texto simples de 5 campos de comprimento variável -

> **Expira** - A data em que o cookie irá expirar. Se estiver em branco, o cookie expirará quando o visitante sair do navegador.
>
> **Domínio** - o nome de domínio do seu site.
>
> **Caminho** - O caminho para o diretório ou página da web que define o cookie. Isso pode ficar em branco se você quiser recuperar o cookie de qualquer diretório ou página.
>
> **Seguro** - Se este campo contiver a palavra "seguro", o cookie só poderá ser recuperado com um servidor seguro. Se este campo estiver em branco, essa restrição não existe.
>
> **Name = Value** - Os cookies são definidos e recuperados na forma de pares de chave e valor.

Os cookies foram originalmente projetados para programação CGI e os dados dos cookies são transmitidos automaticamente entre o navegador da web e o servidor da web, portanto, os scripts CGI no servidor podem ler e gravar valores de cookies armazenados no cliente.

O VBScript também pode manipular cookies usando a propriedade cookie do objeto *Document* . O VBScript pode ler, criar, modificar e excluir o cookie ou cookies que se aplicam à página da web atual.

Armazenando Cookies

A maneira mais simples de criar um cookie é atribuir um valor de string ao objeto *document.cookie* , que se parece com isso -

Sintaxe

```
document.cookie = "key1 = value1;key2 = value2;expires = date"
```

Aqui *expira* atributo é opcional. Se você fornecer a este atributo uma data ou hora válida, o cookie expirará na data ou hora indicadas e após esse valor os cookies não estarão acessíveis.

Exemplo

A seguir, o exemplo para definir um nome de cliente no cookie de *entrada* .

```
<html>
   <head>
      <script type = "text/vbscript">
         Function WriteCookie
            If document.myform.customer.value = "" Then
               msgbox "Enter some value!"
            Else
               cookievalue = (document.myform.customer.value)
               document.cookie = "name = " + cookievalue
               msgbox "Setting Cookies : " & "name = " & cookievalue
            End If
         End Function
      </script>
   </head>

   <body>
      <form name = "myform" action = "">
         Enter name: <input type = "text" name = "customer"/>
         <input type = "button" value = "Set Cookie" onclick = "WriteCookie()"/>
      </form>
   </body>
</html>
```

Isso produzirá o seguinte resultado. Agora digite algo na caixa de texto e pressione o botão "Set Cookie" para definir os cookies.

Enter name: [ ] [Set Cookie]

Agora, seu sistema tem um cookie chamado *nome* . Você pode definir vários cookies usando vários pares *chave = valor* separados por vírgula. Você aprenderá a ler este cookie na próxima seção.

Lendo cookies

Ler um cookie é tão simples quanto escrever um, porque o valor do objeto *document.cookie* é o cookie. Então, você pode usar essa string sempre que quiser acessar o cookie. A string *document.cookie* manterá uma lista de pares *name = value*

separados por ponto-e-vírgula, em que *name* é o *nome* de um cookie e value é seu valor de string. Você pode usar a função *split ()* das strings para dividir a string em key e values como segue -

Exemplo

A seguir, o exemplo para obter os cookies definidos na seção anterior -

```
<html>
   <head>
      <script type = "text/vbscript">
         Function ReadCookie
            allcookies = document.cookie
            msgbox "All Cookies : " + allcookies
            cookiearray = split(allcookies,";")

            For i = 0 to ubound(cookiearray)
               Name  = Split(cookiearray(i),"=")
               Msgbox "Key is : " + Name(0) + " and Value is : " + Name(1)
            Next
         End Function
      </script>
   </head>

   <body>
      <form name = "myform" action = "">
         <input type = "button" value = "Get Cookie" onclick = "ReadCookie()"/>
      </form>
   </body>
</html>
```

Nota - Aqui, o *UBound* é um método da classe *Array* , que retorna o tamanho de uma matriz. Vamos discutir Arrays em um capítulo separado; até lá, por favor, tente digerir.

Isso produzirá o seguinte resultado. Agora, pressione o botão "Get Cookie" para ver os cookies, que você definiu na seção anterior.

Get Cookie

Nota - Pode haver alguns outros cookies já definidos em sua máquina. Portanto, o código acima mostrará todos os cookies definidos na sua máquina.

Definindo a data de expiração dos cookies

Você pode prolongar a vida útil de um cookie além da sessão atual do navegador definindo uma data de expiração e salvando a data de expiração dentro do cookie. Isso pode ser feito definindo o atributo *expires* como uma data e hora.

Exemplo

O exemplo a seguir ilustra como definir a data de expiração do cookie após 1 mês -

```
<html>
   <head>
```

```
      <script type = "text/vbscript">
         Function WriteCookie()
            x = now()
            y = dateadd("m",1,now())  ' Making it to expire next
            cookievalue = document.myform.customer.value
            document.cookie = "name = "  & cookievalue
            document.cookie = "expires = " & y
            msgbox("Setting Cookies : " & "name=" & cookievalue )
         End Function
      </script>
   </head>
   <body>
      <form name = "myform" action = "">
         Enter name: <input type = "text" name = "customer"/>
         <input type = "button" value = "Set Cookie" onclick = "WriteCookie()"/>
      </form>
   </body>
</html>
```

Excluindo um Cookie

Às vezes, você desejará excluir um cookie para que as tentativas subsequentes de ler o cookie não retornem nada. Para fazer isso, você só precisa definir a data de vencimento para um horário no passado.

Exemplo

O exemplo a seguir ilustra como excluir um cookie definindo sua data de expiração em 1 mês no passado -

```
<html>
   <head>
      <script type = "text/vbscript">
         Function WriteCookie()
            x = now()
            x = now()
            a = Month(x)-1
            b = day(x)
            c = year(x)
            d = DateSerial(c,a,b)
            e = hour(x)

            msgbox e
            f = minute(x)

            msgbox f
            d = cdate(d & " " & e & ":" & f)

            msgbox d
            cookievalue = document.myform.customer.value
            document.cookie = "name = "  & cookievalue
            document.cookie = "expires = " & d
            msgbox("Setting Cookies : " & "name=" & cookievalue )
         End Function
      </script>
   </head>
   <body>
      <form name = "myform" action = "">
         Enter name: <input type = "text" name = "customer"/>
         <input type = "button" value = "Set Cookie" onclick = "WriteCookie()"/>
```

```
      </form>
   </body>
</html>
```

VBScript - números

As funções numéricas ajudam os desenvolvedores a manipular números de uma maneira eficiente e também os ajuda a converter seus subtipos. Também os ajuda a fazer uso das funções matemáticas inerentes associadas ao VBScript.

Funções de conversão de números

As funções numéricas nos ajudam a converter um determinado número de um subtipo de dados para outro subtipo de dados.

Mostrar exemplos

Sr. Não	Descrição da função
1	**CDbl** Uma função, que converte um dado número de qualquer subtipo de variante em duplo
2	**CInt** Uma função, que converte um determinado número de qualquer subtipo variante em inteiro
3	**CLng** Uma função, que converte um determinado número de qualquer subtipo de variante em Long
4	**CSng** Uma função, que converte um determinado número de qualquer subtipo de variante em único
5	**Hex** Uma função, que converte um dado número de qualquer subtipo variante em hexadecimal

Funções de formatação de números

As funções de formatação de números ajudam os desenvolvedores a expressarem o número fornecido em um formato que desejem.

Mostrar exemplos

Sr. Não	Descrição da função
1	**FormatNumber** Uma função, que retornaria uma expressão formatada como um número
2	**FormatPercent** Uma função, que retornaria uma expressão formatada como uma porcentagem

Funções Matemáticas

As funções matemáticas nos ajudam a avaliar as funções matemáticas e trigonométricas de um dado número de entrada.

Mostrar exemplos

Sr. Não	Descrição da função
1	**Int** Uma função, que retorna a parte inteira do número dado
2	**Consertar** Uma função, que retorna a parte inteira do número dado
3	**Registro** Uma função, que retorna o logaritmo natural do número fornecido. Números negativos não permitidos
4	**Out** Uma função, que retorna o valor Octal da porcentagem dada
5	**Hex** Uma função, que retorna o valor hexadecimal do número fornecido
6	**Rnd** Uma função, que retorna um número aleatório entre 0 e 1
7	**Sgn**

	Uma função, que retorna um número correspondente ao sinal do número especificado
8	**Sqr** Uma função, que retorna a raiz quadrada do número fornecido. Números negativos não permitidos
9	**Abs** Uma função, que retorna o valor absoluto do número dado
10	**Exp.** Uma função, que retorna o valor de e elevado para o número especificado
11	**Pecado** Uma função, que retorna o valor seno do número dado
12	**Cos** Uma função, que retorna o valor de cosseno do número dado
13	**bronzeado** Uma função, que retorna o valor de tan do número dado

VBScript - Strings

Strings são uma sequência de caracteres, que pode consistir em alfabetos ou números ou caracteres especiais ou todos eles. Uma variável é considerada uma string se estiver entre aspas duplas "".

Sintaxe

```
variablename = "string"
```

Exemplos

```
str1 = "string"    ' Only Alphabets
str2 = "132.45"    ' Only Numbers
str3 = "!@#$;*"    ' Only Special Characters
Str4 = "Asc23@#"   ' Has all the above
```

Funções de String

Existem funções VBScript String predefinidas, que ajudam os desenvolvedores a trabalhar com as strings de forma muito eficaz. Abaixo estão os métodos String que são suportados no VBScript. Por favor, clique em cada um dos métodos para conhecer detalhadamente.

Nome da Função	Descrição
InStr	Retorna a primeira ocorrência da substring especificada. A pesquisa acontece da esquerda para a direita.
InstrRev	Retorna a primeira ocorrência da substring especificada. A pesquisa acontece da direita para a esquerda.
Lcase	Retorna o minúsculo da string especificada.
Ucase	Retorna a maiúscula da string especificada.
Esquerda	Retorna um número específico de caracteres do lado esquerdo da string.
Certo	Retorna um número específico de caracteres do lado direito da string.
Mid	Retorna um número específico de caracteres de uma string com base nos parâmetros especificados.
Ltrim	Retorna uma string depois de remover os espaços no lado esquerdo da string especificada.
Rtrim	Retorna uma string depois de remover os espaços no lado direito da string especificada.
aparar	Retorna um valor de sequência após remover os espaços em branco iniciais e finais.
Len	Retorna o comprimento da string dada.
Substituir	Retorna uma string depois de substituir uma string por outra string.
Espaço	Preenche uma string com o número especificado de espaços.
StrComp	Retorna um valor inteiro depois de comparar as duas strings especificadas.
Corda	Retorna um String com um caractere especificado o número especificado de vezes.
StrReverse	Retorna uma String depois de reverter a seqüência dos caracteres da string dada.

VBScript - matrizes

O que é uma matriz?

Sabemos muito bem que uma variável é um contêiner para armazenar um valor. Às vezes, os desenvolvedores estão em posição de manter mais de um valor em uma única variável por vez. Quando uma série de valores é armazenada em uma única variável, ela é conhecida como uma **variável de matriz** .

Declaração de Array

As matrizes são declaradas da mesma forma que uma variável foi declarada, exceto que a declaração de uma variável de matriz usa parênteses. No exemplo a seguir, o tamanho da matriz é mencionado nos colchetes.

```
'Method 1 : Using Dim
Dim arr1() 'Without Size

'Method 2 : Mentioning the Size
Dim arr2(5) 'Declared with size of 5

'Method 3 : using 'Array' Parameter
Dim arr3
arr3 = Array("apple","Orange","Grapes")
```

Embora o tamanho da matriz seja indicado como 5, ele pode conter 6 valores, pois o índice da matriz é iniciado a partir de ZERO.

O índice de matriz não pode ser negativo.

VBScript Arrays pode armazenar qualquer tipo de variável em uma matriz. Portanto, um array pode armazenar um inteiro, string ou caracteres em uma única variável de array.

Atribuindo Valores a uma Matriz

Os valores são atribuídos à matriz, especificando o valor do índice da matriz em relação a cada um dos valores a serem atribuídos. Pode ser uma string.

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim arr(5)
         arr(0) = "1"            'Number as String
         arr(1) = "VBScript"     'String
         arr(2) = 100            'Number
         arr(3) = 2.45           'Decimal Number
         arr(4) = #10/07/2013#   'Date
         arr(5) = #12.45 PM#     'Time

         document.write("Value stored in Array index 0 : " & arr(0) & "<br />")
         document.write("Value stored in Array index 1 : " & arr(1) & "<br />")
         document.write("Value stored in Array index 2 : " & arr(2) & "<br />")
         document.write("Value stored in Array index 3 : " & arr(3) & "<br />")
         document.write("Value stored in Array index 4 : " & arr(4) & "<br />")
```

```
         document.write("Value stored in Array index 5 : " & arr(5) & "<br />")

      </script>
   </body>
</html>
```

Quando o código acima é salvo como .HTML e executado no Internet Explorer, ele produz o seguinte resultado -

```
Value stored in Array index 0 : 1
Value stored in Array index 1 : VBScript
Value stored in Array index 2 : 100
Value stored in Array index 3 : 2.45
Value stored in Array index 4 : 7/10/2013
Value stored in Array index 5 : 12:45:00 PM
```

Matrizes Multi Dimensão

As matrizes não estão limitadas apenas à dimensão única e podem ter um máximo de 60 dimensões. Matrizes bidimensionais são as mais usadas.

Exemplo

No exemplo a seguir, um array multidimensional é declarado com 3 linhas e 4 colunas.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim arr(2,3)   ' Which has 3 rows and 4 columns
         arr(0,0) = "Apple"
         arr(0,1) = "Orange"
         arr(0,2) = "Grapes"
         arr(0,3) = "pineapple"

         arr(1,0) = "cucumber"
         arr(1,1) = "beans"
         arr(1,2) = "carrot"
         arr(1,3) = "tomato"

         arr(2,0) = "potato"
         arr(2,1) = "sandwitch"
         arr(2,2) = "coffee"
         arr(2,3) = "nuts"

         document.write("Value in Array index 0,1 : " &  arr(0,1) & "<br />")
         document.write("Value in Array index 2,2 : " &  arr(2,2) & "<br />")

      </script>
   </body>
</html>
```

Quando o código acima é salvo como .HTML e executado no Internet Explorer, ele produz o seguinte resultado -

```
Value stored in Array index : 0 , 1 : Orange
Value stored in Array index : 2 , 2 : coffee
```

Declaração Redim

A instrução ReDim é usada para declarar variáveis de matriz dinâmica e alocar ou realocar espaço de armazenamento.

```
ReDim [Preserve] varname(subscripts) [, varname(subscripts)]
```

Preservar - Um parâmetro opcional usado para preservar os dados em uma matriz existente quando você altera o tamanho da última dimensão.

varname - Um parâmetro necessário, que denota o nome da variável, que deve seguir as convenções de nomenclatura da variável padrão.

subscripts - Um parâmetro necessário, que indica o tamanho da matriz.

Exemplo

No exemplo abaixo, uma matriz foi redefinida e, em seguida, preservada os valores quando o tamanho existente da matriz é alterado.

Nota - Ao redimensionar um array menor do que era originalmente, os dados nos elementos eliminados serão perdidos.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim a()
         i = 0
         redim a(5)
         a(0) = "XYZ"
         a(1) = 41.25
         a(2) = 22

         REDIM PRESERVE a(7)
         For i = 3 to 7
         a(i) = i
         Next

         'to Fetch the output
         For i = 0 to ubound(a)
            Msgbox a(i)
         Next
      </script>
   </body>
</html>
```

Quando salvamos o script acima como HTML e o executamos no Internet Explorer, ele produz o seguinte resultado.

```
XYZ
41.25
```

```
22
3
4
5
6
7
```

Métodos de Array

Existem várias funções embutidas no VBScript que ajudam os desenvolvedores a lidar efetivamente com as matrizes. Todos os métodos usados em conjunto com matrizes estão listados abaixo. Por favor, clique no nome do método para saber em detalhes.

Função	Descrição
Lound	Uma função, que retorna um inteiro que corresponde ao menor subscrito dos arrays fornecidos.
UBound	Uma função, que retorna um inteiro que corresponde ao maior subscrito dos arrays fornecidos.
Dividido	Uma função, que retorna uma matriz que contém um número especificado de valores. Dividido com base em um Delimitador.
Junte-se	Uma função, que retorna um String que contém um número especificado de substrings em um array. Essa é uma função exatamente oposta do método de divisão.
Filtro	Uma função, que retorna uma matriz baseada em zero que contém um subconjunto de uma matriz de seqüência de caracteres com base em um critério de filtro específico.
IsArray	Uma função, que retorna um valor booleano que indica se a variável de entrada é ou não uma matriz.
Apagar	Uma função, que recupera a memória alocada para as variáveis da matriz.

VBScript - Funções de data e hora

As funções de data e hora do VBScript ajudam os desenvolvedores a converter a data e a hora de um formato para outro ou a expressar o valor de data ou hora no formato adequado a uma condição específica.

Funções de data

Função	Descrição
Encontro	Uma função, que retorna a data atual do sistema

CDate	Uma função, que converte uma determinada entrada em data
DateAdd	Uma função, que retorna uma data na qual um intervalo de tempo especificado foi adicionado
DateDiff	Uma função, que retorna a diferença entre dois períodos de tempo
DataParte	Uma função, que retorna uma parte especificada do valor de data de entrada fornecido
DateSerial	Uma função, que retorna uma data válida para o ano, mês e data
FormatDateTime	Uma função, que formata a data com base nos parâmetros fornecidos
IsDate	Uma função, que retorna um valor booleano, quer o parâmetro fornecido seja ou não uma data
Dia	Uma função, que retorna um inteiro entre 1 e 31 que representa o dia da data especificada
Mês	Uma função, que retorna um inteiro entre 1 e 12 que representa o mês da data especificada
Ano	Uma função, que retorna um inteiro que representa o ano da data especificada
MonthName	Uma função, que retorna o nome do mês específico para a data especificada
Dia da semana	Uma função, que retorna um inteiro (1 a 7) que representa o dia da semana para o dia especificado.
WeekDayName	Uma função, que retorna o nome do dia da semana para o dia especificado.

Funções de tempo

Função	Descrição
Agora	Uma função, que retorna a data e hora atuais do sistema
Hora	Uma função, que retorna e inteiro entre 0 e 23, que representa a parte Hour do tempo determinado
Minuto	Uma função, que retorna e inteiro entre 0 e 59, que representa a parte Minutos do tempo determinado
Segundo	Uma função, que retorna e inteiro entre 0 e 59, que representa a parte Segundos do tempo determinado
Tempo	Uma função, que retorna a hora atual do sistema

Cronômetro	Uma função, que retorna o número de segundos e milésimos de segundo desde as 12h.
TimeSerial	Uma função, que retorna o tempo para a entrada específica de hora, minuto e segundo
Valor do tempo	Uma função, que converte a string de entrada em um formato de hora

VBScript - Procedimentos

O que é uma função?

Uma função é um grupo de código reutilizável que pode ser chamado em qualquer parte do programa. Isso elimina a necessidade de escrever o mesmo código repetidas vezes. Isso permitirá que os programadores dividam um grande programa em várias funções pequenas e gerenciáveis. Além de funções internas, o VBScript nos permite escrever funções definidas pelo usuário também. Esta seção irá explicar como escrever suas próprias funções no VBScript.

Definição de Função

Antes de usarmos uma função, precisamos definir essa função específica. A maneira mais comum de definir uma função no VBScript é usando a palavra-chave Function, seguida por um nome de função exclusivo e pode ou não ter uma lista de parâmetros e uma instrução com uma palavra-chave **End Function** , que indica o fim da função .

A sintaxe básica é mostrada abaixo -

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Function Functionname(parameter-list)
            statement 1
            statement 2
            statement 3
            .......
            statement n
         End Function

      </script>
   </body>
</html>
```

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Function sayHello()
            msgbox("Hello there")
         End Function

      </script>
   </body>
</html>
```

Chamando uma Função

Para invocar uma função em algum lugar no script, você precisaria escrever o nome dessa função com a palavra-chave **Call** .

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Function sayHello()
            msgbox("Hello there")
         End Function

         Call sayHello()

      </script>
   </body>
</html>
```

Parâmetros de Função

Até agora, vimos a função sem um parâmetro, mas há um recurso para passar parâmetros diferentes ao chamar uma função. Esses parâmetros passados podem ser capturados dentro da função e qualquer manipulação pode ser feita sobre esses parâmetros. As funções são chamadas usando a **palavra-** chave de **chamada** .

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Function sayHello(name, age)
            msgbox( name & " is " & age & " years old.")
         End Function

         Call sayHello("Tutorials point", 7)

      </script>
   </body>
</html>
```

Retornando um valor de uma função

Uma função VBScript pode ter uma instrução de retorno opcional. Isso é necessário se você quiser retornar um valor de uma função. Por exemplo, você pode passar dois

números em uma função e, em seguida, você pode esperar da função para retornar sua multiplicação em seu programa de chamada.

NOTA - Uma função pode retornar vários valores separados por vírgula como uma matriz atribuída ao próprio nome da função.

Exemplo

Essa função usa dois parâmetros e concatena-os e retorna o resultado no programa de chamada. No VBScript, os valores são retornados de uma função usando o nome da função. Caso você queira retornar dois ou mais valores, o nome da função é retornado com uma matriz de valores. No programa de chamada, o resultado é armazenado na variável de resultado.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Function concatenate(first, last)
            Dim full
            full = first & last
            concatenate = full  'Returning the result to the function name itself
         End Function

      </script>
   </body>
</html>
```

Agora, podemos chamar essa função da seguinte maneira -

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Function concatenate(first, last)
            Dim full
            full = first & last
            concatenate = full  'Returning the result to the function name itself
         End Function
         ' Here is the usage of returning value from  function.
         dim result
            result = concatenate("Zara", "Ali")
        msgbox(result)
      </script>
   </body>
</html>
```

Sub procedimentos

Os subprocessos são semelhantes às funções, mas há poucas diferenças.

> Subprocedimentos DONOT Retorna um valor enquanto as funções podem ou não retornar um valor.
>
> Subprocedimentos Podem ser chamados sem palavra-chave de chamada.

Os subprocedimentos são sempre incluídos nas instruções **Sub** e **End Sub** .

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Sub sayHello()
            msgbox("Hello there")
         End Sub

      </script>
   </body>
</html>
```

Procedimentos de Chamada

Para invocar um Procedimento em algum lugar posterior no script, basta escrever o nome desse procedimento com ou sem a palavra-chave **Call** .

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Sub sayHello()
            msgbox("Hello there")
         End Sub
         sayHello()

      </script>
   </body>
</html>
```

Conceitos Avançados para Funções

Há muito o que aprender sobre as funções do VBScript. Podemos passar o parâmetro por valor ou por referência. Por favor, clique em cada um deles para saber mais.

ByVal- Passar os parâmetros por valor

ByRef- Passe os parâmetros pela referência

VBScript - caixas de diálogo

O que é uma caixa de diálogo?

O VBScript permite que os desenvolvedores interajam com o usuário de maneira eficaz. Pode ser uma caixa de mensagem para exibir uma mensagem para um usuário ou uma caixa de entrada com a qual o usuário pode inserir os valores.

Função MsgBox VBScript

A função MsgBox exibe uma caixa de mensagem e aguarda o usuário clicar em um botão e, em seguida, uma ação é executada com base no botão clicado pelo usuário.

Sintaxe

```
MsgBox(prompt[,buttons][,title][,helpfile,context])
```

Parâmetro Descrição

Prompt - um parâmetro obrigatório. Um String que é exibido como uma mensagem na caixa de diálogo. O comprimento máximo do prompt é de aproximadamente 1024 caracteres. Se a mensagem se estender para mais de uma linha, podemos separar as linhas usando um caractere de retorno de carro (Chr (13)) ou um caractere de avanço de linha (Chr (10)) entre cada linha.

botões - um parâmetro opcional. Uma expressão numérica que especifica o tipo de botões a serem exibidos, o estilo de ícone a ser usado, a identidade do botão padrão e a modalidade da caixa de mensagem. Se deixado em branco, o valor padrão para os botões é 0.

Título - um parâmetro opcional. Uma expressão String exibida na barra de título da caixa de diálogo. Se o título for deixado em branco, o nome do aplicativo é colocado na barra de título.

helpfile - Um parâmetro opcional. Uma expressão String que identifica o arquivo de Ajuda a ser usado para fornecer ajuda sensível ao contexto para a caixa de diálogo.

contexto - um parâmetro opcional. Uma expressão numérica que identifica o número do contexto da Ajuda atribuído pelo autor da Ajuda ao tópico da Ajuda apropriado. Se o contexto for fornecido, o arquivo de ajuda também deverá ser fornecido.

O parâmetro **Buttons** pode pegar qualquer um dos seguintes valores -

0 vbOKOnly Exibe apenas o botão OK.

1 vbOKCancel Exibe os botões OK e Cancelar.

2 vbAbortRetryIgnore Exibe os botões Anular, Repetir e Ignorar.

3 vbYesNoCancel Exibe os botões Sim, Não e Cancelar.

4 vbYesNo Exibe os botões Sim e Não.

5 vbRetryCancel Exibe os botões Repetir e Cancelar.

16 vbCritical Exibe o ícone Mensagem Crítica.

32 vbQuestion Exibe o ícone Consulta de Aviso.

48 vbExclamação Exibe o ícone Mensagem de Aviso.

64 vbInformation Exibe o ícone Message Message.

0 vbDefaultButton1 O primeiro botão é o padrão.

256 vbDefaultButton2 O segundo botão é o padrão.

512 vbDefaultButton3 O terceiro botão é o padrão.

768 vbDefaultButton4 O quarto botão é o padrão.

0 modal do aplicativo vbApplicationModal. O aplicativo atual não funcionará até que o usuário responda à caixa de mensagem.

4096 vbSystemModal System modal. Todos os aplicativos não funcionarão até que o usuário responda à caixa de mensagens.

Os valores acima são logicamente divididos em quatro grupos: O primeiro grupo (0 a 5) indica os botões a serem exibidos na caixa de mensagem. O segundo grupo (16, 32, 48, 64) descreve o sytle do ícone a ser exibido, o terceiro grupo (0, 256, 512, 768) indica qual botão deve ser o padrão e o quarto grupo (0, 4096) determina a modalidade da caixa de mensagem.

Valores de retorno

A função MsgBox pode retornar um dos seguintes valores -

1 - vbOK - OK foi clicado

2 - vbCancel - Cancelar foi clicado

3 - vbAbort - Abort foi clicado

4 - vbRetry - Repetir foi clicado

5 - vbIgnore - Ignore foi clicado

6 - vBYes - Sim foi clicado

7 - vbNo - Não foi clicado

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         'Message Box with just prompt message
         MsgBox("Welcome")

         'Message Box with title, yes no and cancel Butttons
         a = MsgBox("Do you like blue color?",3,"Choose options")
         ' Assume that you press No Button
         document.write("The Value of a is " & a)

      </script>
   </body>
</html>
```

Quando o script acima é executado, a caixa de mensagem é exibida e, se você pressionar No Button, o valor de a é 7.

```
The Value of a is 7
```

Função InputBox VBScript

A função InputBox ajuda o usuário a obter os valores do usuário. Depois de inserir os valores, se o usuário clicar no botão OK ou pressionar ENTER no teclado, a função InputBox retornará o texto na caixa de texto. Se o usuário clicar no botão Cancelar, a função retornará uma seqüência vazia ("").

Sintaxe

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

Parâmetro Descrição

Prompt - um parâmetro obrigatório. Um String que é exibido como uma mensagem na caixa de diálogo. O comprimento máximo do prompt é de aproximadamente 1024 caracteres. Se a mensagem se estender para mais de uma linha, podemos separar as linhas usando um caractere de retorno de carro (Chr (13)) ou um caractere de avanço de linha (Chr (10)) entre cada linha.

Título - um parâmetro opcional. Uma expressão String exibida na barra de título da caixa de diálogo. Se o título for deixado em branco, o nome do aplicativo é colocado na barra de título.

Padrão - um parâmetro opcional. Um texto padrão na caixa de texto que o usuário gostaria de exibir.

XPos - um parâmetro opcional. A posição do eixo X, que representa a distância do prompt do lado esquerdo da tela horizontalmente. Se deixado em branco, a caixa de entrada é centralizada horizontalmente.

YPos - um parâmetro opcional. A posição do eixo Y, que representa a distância do prompt do lado esquerdo da tela verticalmente. Se deixado em branco, a caixa de entrada é centralizada verticalmente.

helpfile - Um parâmetro opcional. Uma expressão String que identifica o arquivo de Ajuda a ser usado para fornecer ajuda sensível ao contexto para a caixa de diálogo.

contexto - um parâmetro opcional. Uma expressão numérica que identifica o número do contexto da Ajuda atribuído pelo autor da Ajuda ao tópico da Ajuda apropriado. Se o contexto for fornecido, o arquivo de ajuda também deverá ser fornecido.

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         ' Input Box with only Prompt
         InputBox("Enter a number")

         ' Input Box with a Title
         a = InputBox("Enter a Number","Enter Value")
         msgbox a

         ' Input Box with a Prompt,Title and Default value
         a = InputBox("Enter a Number","Enter Value",123)
         msgbox a

         ' Input Box with a Prompt,Title,Default and XPos
         a = InputBox("Enter your name","Enter Value",123,700)
         msgbox a

         ' Input Box with a Prompt,Title and Default and YPos
         a = InputBox("Enter your name","Enter Value",123,,500)
         msgbox a

      </script>
   </body>
</html>
```

Quando o script acima é executado, a caixa de entrada é exibida e exibe o valor inserido pelo usuário.

VBScript Orientado a Objetos

O que é um objeto

Os objetos de tempo de execução do VBScript nos ajudam a realizar várias tarefas. Esta seção ajudará você a entender como instanciar um objeto e trabalhar com ele.

Sintaxe

Para trabalhar com objetos sem problemas, precisamos declarar o objeto e instanciá-lo usando **Set** Keyword.

```
Dim objectname    'Declare the object name
Set objectname = CreateObject(object_type)
```

Exemplo

No exemplo abaixo, estamos criando um objeto do tipo **Scripting.Dictionary** .

```
Dim obj
Set obj = CreateObject("Scripting.Dictionary")
```

Destruindo os Objetos

O significado de destruir o objeto é liberar a memória e redefinir a variável de objeto.

Sintaxe

Para destruir os objetos, precisamos usar **Set** Keyword seguido do nome do objeto e apontá-lo para **Nothing** .

```
Set objectname = Nothing 'Destroy the object.
```

Exemplo

No exemplo abaixo, estamos criando um objeto do tipo **Scripting.Dictionary** .

```
Dim obj
Set obj = CreateObject("Scripting.Dictionary")
Set obj = Nothing.
```

Uso de Objeto

Por favor, clique em cada um dos tipos de objetos para saber mais.

Tipo de objeto	Descrição
Classe	Classe é um contêiner, que contém métodos e variáveis associadas a ele e é acessado pela criação de um objeto de Type Class.
Scripting.FileSystemObject	É o grupo de objetos com os quais podemos trabalhar com o sistema de arquivos.
Scripting.Dictionary	Um grupo de objetos, que são usados para criar os objetos do dicionário.
Depurar	Um Objeto Global com o qual podemos enviar a saída para o depurador de scripts da Microsoft.

VBScript - expressões regulares

Regular Expressions é uma seqüência de caracteres que forma um padrão, que é usado principalmente para pesquisa e substituição. O propósito de criar um padrão é combinar cadeias específicas, para que o desenvolvedor possa extrair caracteres com base nas condições e substituir determinados caracteres.

Objeto RegExp

O objeto RegExp ajuda os desenvolvedores a corresponderem ao padrão de strings e as propriedades e métodos nos ajudam a trabalhar com expressões regulares facilmente. É semelhante ao RegExp em JavaScript

Propriedades

Padrão - O método Padrão representa uma string usada para definir a expressão regular e deve ser definida antes de usar o objeto de expressão regular.

IgnoreCase - Uma propriedade booleana que representa se a expressão regular deve ser testada em relação a todas as correspondências possíveis em uma string, se for verdadeira ou falsa. Se não for especificado explicitamente, o valor IgnoreCase será definido como False.

Global - Uma propriedade booleana que representa se a expressão regular deve ser testada em relação a todas as correspondências possíveis em uma string. Se não for especificado explicitamente, o valor Global será definido como Falso.

Métodos

Test (string de pesquisa) - O método Test usa uma string como argumento e retorna True se a expressão regular puder ser correspondida com sucesso na string, caso contrário, False será retornado.

Substituir (string de pesquisa, string de substituição) - O método Replace usa 2 parâmetros. Se a pesquisa for bem-sucedida, ela substituirá essa correspondência pela string de substituição e a nova string será retornada. Se não houver correspondências, a string de pesquisa original será retornada.

Execute (search-string) - O método Execute funciona como Replace, exceto que ele retorna um objeto de coleção Matches, contendo um objeto Match para cada correspondência bem-sucedida. Não modifica a string original.

Objeto de coleta de correspondências

O objeto da coleção Matches é retornado como resultado do método Execute. Esse objeto de coleção pode conter zero ou mais objetos Match e as propriedades desse objeto são somente leitura.

Contagem - O método Count representa o número de objetos de correspondência na coleção.

Item - O método Item permite que os objetos de correspondência sejam acessados a partir do objeto de coleções de correspondências.

Corresponder objeto

O objeto Match está contido no objeto de coleta de correspondências. Esses objetos representam a correspondência bem-sucedida após a pesquisa por uma string.

FirstIndex - Representa a posição dentro da string original onde a correspondência ocorreu. Este índice é baseado em zero, o que significa que a

primeira posição em uma string é 0.

Comprimento - Um valor que representa o comprimento total da cadeia combinada.

Valor - um valor que representa o valor ou o texto correspondente. Também é o valor padrão ao acessar o objeto Match.

Tudo sobre o parâmetro padrão

O edifício padrão é semelhante ao PERL. A construção de padrões é a coisa mais importante ao trabalhar com expressões regulares. Nesta seção, abordaremos como criar um padrão com base em vários fatores.

Correspondência de posição

A importância da correspondência de posição é garantir que colocamos as expressões regulares nos locais corretos.

Símbolo	Descrição
^	Corresponde apenas ao começo de uma string.
$	Corresponde apenas ao final de uma string.
\ b	Corresponde a qualquer limite de palavra
\ B	Corresponde a qualquer limite não relacionado a palavras

Correspondência de literais

Qualquer forma de caractere, como alfabeto, número ou caractere especial, ou até decimal, hexadecimal pode ser tratada como um literal. Como poucos dos caracteres já possuem um significado especial no contexto da Expressão Regular, precisamos escapar deles usando sequências de escape.

Símbolo	Descrição
Alfanumérico	Corresponde apenas aos caracteres alfabéticos e numéricos.
\ n	Corresponde a uma nova linha.
\ [	Correspondências [apenas literal
\]	Jogos] literal apenas
\ (	Correspondências (apenas literal
\)	Correspondências) apenas literal
\ t	Corresponde a guia horizontal
\ v	Corresponde à guia vertical

\ |	Jogos | apenas literal
\ {	Correspondências {apenas literal
\}	Corresponde apenas} literal
\\	Somente correspondências \ literal
\?	Fósforos ? apenas literal
\ *	Corresponde apenas a literais
\ +	Somente correspondências literais
\.	Fósforos . apenas literal
\ b	Corresponde a qualquer limite de palavra
\ B	Corresponde a qualquer limite não relacionado a palavras
\ f	Corresponde a um feed de formulário
\ r	Corresponde ao retorno de carro
\ xxx	Corresponde ao caractere ASCII de um número octal xxx.
\ xdd	Corresponde ao caractere ASCII de um número hexadecimal dd.
\ uxxxx	Corresponde ao caractere ASCII de um xxxx literal UNICODE.

Classes de Personagem Correspondentes

As classes de caracteres são o padrão formado pelo agrupamento personalizado e entre colchetes []. Se estamos esperando uma classe de caracteres que não deveria estar na lista, então devemos ignorar essa classe de caracteres em particular usando o símbolo negativo, que é um limite ^.

Símbolo	Descrição
[xyz]	Corresponder qualquer uma das classes de caracteres contidas no conjunto de caracteres.
[^ xyz]	Corresponde a qualquer uma das classes de caracteres que NÃO estão contidas no conjunto de caracteres.
.	Corresponde a qualquer classe de personagem, exceto \ n
\W	Corresponda qualquer classe de caractere de palavra. Equivalente a [a-zA-Z_0-9]
\W	Corresponda a qualquer classe de caracteres que não seja de palavra. Equivalente a [^ a-zA-Z_0-9]
\ d	Corresponder a qualquer classe de dígitos. Equivalente a [0-9].

\ D	Corresponder qualquer classe de caracteres não dígitos. Equivalente a [^ 0-9].
\ s	Corresponder qualquer classe de caractere de espaço. Equivalente a [\ t \ r \ n \ v \ f]
\ S	Corresponder qualquer classe de caractere de espaço. Equivalente a [^ \ t \ r \ n \ v \ f]

Correspondência de Repetição

A correspondência de repetição permite várias pesquisas na expressão regular. Também especifica o número de vezes que um elemento é repetido em uma expressão regular.

Símbolo	Descrição
*	Corresponde a zero ou mais ocorrências da Expressão regular fornecida. Equivalente a {0,}.
+	Corresponde a uma ou mais ocorrências da Expressão regular fornecida. Equivalente a {1,}.
?	Corresponde a zero ou uma ocorrência da Expressão regular fornecida. Equivalente a {0,1}.
{x}	Corresponde exatamente ao número x de ocorrências da expressão regular dada.
{x,}	Corresponde pelo menos x ou mais ocorrências da expressão regular dada.
{x, y}	Corresponde x ao número y de ocorrências da expressão regular dada.

Alternação e Agrupamento

A alternância e o agrupamento ajudam os desenvolvedores a criar Expressões Regulares mais complexas, especialmente lidando com cláusulas complexas em uma Expressão Regular, o que proporciona uma grande flexibilidade e controle.

Símbolo	Descrição
0	Agrupando uma cláusula para criar uma cláusula. "(xy)? (z)" combina "xyz" ou "z".
|	Alternação combina uma cláusula de expressão regular e, em seguida, corresponde a qualquer uma das cláusulas individuais. "(ij) | (23) | (pq)" combina "ij" ou "23" ou "pq".

Construindo Expressões Regulares

Dada a seguir são alguns exemplos que explicam claramente como construir uma expressão regular.

Expressão regular	Descrição
"^ \ s * .." e ".. \ s * $"	Representa que pode haver qualquer número de caracteres de espaço à esquerda e à direita em uma única linha.
"((\ $ \ s?)	(# \ s?))?"	Representa um sinal $ ou # opcional seguido por um espaço opcional.
"((\ d + (\. (\ d \ d)?)?))"	Representa que pelo menos um dígito está presente seguido por uma decimais opcional e dois dígitos após decimais.

Exemplo

O exemplo abaixo verifica se o usuário inseriu um ID de email cujo formato deve corresponder, de forma que haja um ID de email seguido por '@' e, em seguida, seguido por um nome de domínio.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         strid = "welcome.user@tutorialspoint.co.us"
         Set re = New RegExp
         With re
            .Pattern    = "^[\w-\.]{1,}\@([\da-zA-Z-]{1,}\.){1,}[\da-zA-Z-]{2,3}$"
            .IgnoreCase = False
            .Global     = False
         End With

         ' Test method returns TRUE if a match is found
         If re.Test( strid ) Then
            Document.write(strid & " is a valid e-mail address")
         Else
            Document.write(strid & " is NOT a valid e-mail address")
         End If

        Set re = Nothing
      </script>
   </body>
</html>
```

VBScript - tratamento de erros

Existem três tipos de erros na programação: (a) Erros de sintaxe, (b) Erros de tempo de execução e (c) Erros lógicos.

Erros de sintaxe

Erros de sintaxe, também chamados de erros de análise, ocorrem no momento da interpretação para o VBScript. Por exemplo, a linha a seguir causa um erro de sintaxe porque está faltando um parêntese de fechamento -

```
<script type = "text/vbscript">

   dim x,y
   x = "Tutorialspoint"
   y = Ucase(x

</script>
```

Erros de Tempo de Execução

Erros de tempo de execução, também chamados de exceções, ocorrem durante a execução, após a interpretação. Por exemplo, a linha a seguir causa um erro de tempo de execução porque a sintaxe aqui está correta, mas no tempo de execução está tentando chamar fnmultiply, que é uma função não existente -

```
<script type = "text/vbscript">
   Dim x,y
   x = 10
   y = 20
   z = fnadd(x,y)
   a = fnmultiply(x,y)

   Function fnadd(x,y)
      fnadd = x+y
   End Function

</script>
```

Erros lógicos

Erros lógicos podem ser o tipo mais difícil de erros a serem rastreados. Esses erros não são o resultado de uma sintaxe ou erro de tempo de execução. Em vez disso, ocorrem quando você comete um erro na lógica que direciona seu script e não obtém o resultado esperado. Você não pode pegar esses erros, porque depende do seu requisito de negócio que tipo de lógica você deseja colocar no seu programa. Por exemplo, dividindo um número por zero ou um script que está escrito, que entra em loop infinito.

Objeto Err

AAssume se tivermos um erro de tempo de execução, a execução será interrompida exibindo a mensagem de erro. Como desenvolvedor, se quisermos capturar o erro, o objeto de **erro** é usado.

Exemplo

No exemplo abaixo, **Err.Number** fornece o número do erro e **Err.Description** fornece a descrição do erro.

```
<script type = "text/vbscript">

  Err.Raise 6     ' Raise an overflow error.
```

```
  MsgBox "Error # " & CStr(Err.Number) & " " & Err.Description
  Err.Clear       ' Clear the error.

</script>
```

Declarações Diversas do VBScript

O VBScript tem algumas outras instruções importantes para ajudar os desenvolvedores a desenvolver um script eficiente. A tabela a seguir lista um conjunto de instruções importantes. Neste capítulo, discutiremos cada uma dessas instruções em detalhes com exemplos.

Categoria	Nome da Função / Nome da Declaração
Opções	Option Explicit
ID do mecanismo de script	ScriptEngine
variantes	IsArray, IsEmpty, IsNull, IsNumeric, IsObject, TypeName
Expressão	Eval, Executar
Declaração de controle	Com ... Fim Com
Função matemática	Aleatória

Option Explicit

Option Explicit força o desenvolvedor a declarar as variáveis usando a instrução **Dim** antes de serem usadas em alguma parte do código.

Sintaxe

```
Option Explicit
```

Exemplo

Se usarmos **Option Explicit** e se não declararmos as variáveis, o interpretador irá lançar e errar.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Option Explicit
         Dim x,y,z,a
         x = 10
         y = 20
         z = fnadd(x,y)
         a = fnmultiply(x,y)
```

```
         Function fnadd(x,y)
            fnadd = x+y
         End Function

      </script>
   </body>
</html>
```

ScriptEngine

ScriptEngine representa os detalhes da linguagem de script em uso. Ele também é usado em combinação com **ScriptEngineMajorVersion, ScriptEngineMinor Version, ScriptEngineBuildVersion,** que fornece a versão principal do mecanismo vbscript, a versão secundária do mecanismo vbscript e a versão de compilação do vbscript, respectivamente.

Sintaxe

```
ScriptEngine
```

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim scriptdetails
         scriptdetails =  " Version " & ScriptEngine & " - "
         'For getting Major version, use ScriptEngineMajorVersion'

         scriptdetails = scriptdetails &  ScriptEngineMajorVersion & "."

         'For getting Minor version, use ScriptEngineMinorVersion'
         scriptdetails = scriptdetails & ScriptEngineMinorVersion & "."

         'For getting Build version, use ScriptEngineBuildVersion'
         scriptdetails = scriptdetails & ScriptEngineBuildVersion

         Document.write scriptdetails

      </script>
   </body>
</html>
```

Salve o arquivo com a extensão .html ao executar o script no IE, o seguinte resultado é exibido na tela.

```
Version VBScript - 5.8.16996
```

Está vazia

A função IsEmpty é usada para verificar se a expressão está vazia ou não. Retorna um valor booleano. **IsEmpty** retorna True se a variável for não inicializada ou explicitamente definida como Empty. Caso contrário, a expressão retorna False.

Sintaxe

```
IsEmpty(expression)
```

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim var, MyCheck
         MyCheck = IsEmpty(var)
         Document.write "Line 1 : " & MyCheck & "<br />"

         var = Null   ' Assign Null.
         MyCheck = IsEmpty(var)
         Document.write "Line 2 : " & MyCheck & "<br />"

         var = Empty   ' Assign Empty.
         MyCheck = IsEmpty(var)
         Document.write "Line 3 : " & MyCheck & "<br />"

      </script>
   </body>
</html>
```

Salve o arquivo com a extensão .html ao executar o script no IE, o seguinte resultado é exibido na tela.

```
Line 1 : True
Line 2 : False
Line 3 : True
```

É nulo

A função IsNull é usada para verificar se a expressão possui ou não dados válidos. Retorna um valor booleano. **IsNull** retorna True se a variável for Null. Caso contrário, a expressão retornará False.

Sintaxe

```
IsNull(expression)
```

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim var, res
         res = IsNull(var)
         document.write "Line 1 : " & res & "<br />"

         var = Null
         res = IsNull(var)
         document.write "Line 2 : " & res & "<br />"
```

```
         var = Empty
         res = IsNull(var)
         document.write "Line 3 : " & res & "<br />"

      </script>
   </body>
</html>
```

Salve o arquivo com a extensão .html ao executar o script no IE, o seguinte resultado é exibido na tela.

```
Line 1 : False
Line 2 : True
Line 3 : False
```

IsObject

A função IsObject é usada para verificar se a expressão possui ou não um objeto válido. Retorna um valor booleano. **IsObject** retorna True se a expressão contiver um subtipo de objeto, caso contrário, a expressão retornará False.

Sintaxe

```
IsObject(expression)
```

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
        Dim fso,b
        b = 10
        set fso = createobject("Scripting.Filesystemobject")

        x = isobject(fso)
        Document.write "Line 1 : " &  x  & "<br />"

        y = isobject(b)
        Document.write "Line 2 : " &  y  & "<br />"

      </script>
   </body>
</html>
```

Salve o arquivo com a extensão .html ao executar o script no IE, o seguinte resultado é exibido na tela.

```
Line 1 : True
Line 2 : False
```

IsNumeric

A função IsNumeric é usada para verificar se a expressão possui ou não um subtipo de número. Retorna um valor booleano. **IsObject** retorna True se a expressão contiver um

subtipo numérico, caso contrário, a expressão retornará False.

Sintaxe

```
IsNumeric(expression)
```

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim var, chk
         var = 20
         chk = IsNumeric(var)
         Document.write "Line 1 : " &  chk  & "<br />"

         var = "3.1415935745"
         chk = IsNumeric(var)
         Document.write "Line 2 : " &  chk  & "<br / >"

         var = "20 Chapter 23.123 VBScript"
         chk = IsNumeric(var)
         Document.write "Line 3 : " &  chk  & "<br / >"

      </script>
   </body>
</html>
```

Salve o arquivo com a extensão .html ao executar o script no IE, o seguinte resultado é exibido na tela.

```
Line 1 : True
Line 2 : True
Line 3 : False
```

Digite o nome

A função TypeName é usada para retornar as informações do subtipo de variante da variável.

Sintaxe

```
TypeName(varname)
```

A função Typename pode retornar qualquer um dos seguintes valores.

Byte - Valor de Byte

Inteiro - Valor Inteiro

Valor Inteiro Longo Longo

Single - Valor de ponto flutuante de precisão simples

Duplo - Valor de ponto flutuante de precisão dupla

Moeda - Valor da Moeda

Decimal - Valor Decimal

Data - Data ou Hora Valor

Cadeia - Cadeia de Caracteres Valor

Booleano - Valor booleano

Vazio - Valor não inicializado

Nulo - sem dados válidos

Objeto - typename do objeto

Nothing - Variável de objeto que ainda não se refere a uma instância de objeto

Erro

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim ArrVar(2), vartype
         NullVar = Null   ' Assign Null value.

         vartype = TypeName(3.1450)
         Document.write "Line 1 : " &  vartype  & "<br />"

         vartype = TypeName(432)
         Document.write "Line 2 : " &  vartype  & "<br />"

         vartype = TypeName("Microsoft")
         Document.write "Line 3 : " &  vartype  & "<br />"

         vartype = TypeName(NullVar)
         Document.write "Line 4 : " &  vartype  & "< br />"

         vartype = TypeName(ArrVar)
         Document.write "Line 5 : " &  vartype  & "<br />"

      </script>
   </body>
</html>
```

Salve o arquivo com a extensão .html ao executar o script no IE, o seguinte resultado é exibido na tela.

```
Line 1 : Double
Line 2 : Integer
Line 3 : String
Line 4 : Null
Line 5 : Variant()
```

Eval

A função Eval executa uma expressão e retorna o resultado como uma string ou um número.

Sintaxe

```
Eval(expression)
```

O argumento Expression pode ser uma expressão de string ou um número. Se você passar para a função Eval uma seqüência de caracteres que não contenha uma expressão numérica ou um nome de função, mas apenas uma cadeia de texto simples, ocorrerá um erro em tempo de execução. Por exemplo, Eval ("VBScript") resulta em um erro.

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Document.write Eval("10 + 10") & "<br />"
         Document.write Eval("101 = 200") & "<br />"
         Document.write Eval("5 * 3") & "<br />"

      </script>
   </body>
</html>
```

Salve o arquivo com a extensão .html ao executar o script no IE, o seguinte resultado é exibido na tela.

```
20
false
15
```

Executar

A instrução Execute aceita o argumento que é uma expressão de cadeia contendo uma ou mais instruções para execução.

Sintaxe

```
Execute(expression)
```

No VBScript, a = b pode ser interpretado de duas maneiras. Ele pode ser tratado como uma instrução de atribuição em que o valor de x é atribuído a y. Ele também pode ser interpretado como uma expressão que testa se a e b têm o mesmo valor. Se o fizerem, o resultado é verdadeiro; se não forem, o resultado é Falso. A instrução Execute sempre usa a primeira interpretação, enquanto a instrução Eval sempre usa a segunda.

Exemplo

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
```

```
         Dim x
         x = "Global"
         y = "VBScript"
         Execute("x = y")

         msgbox x
         msgbox y

      </script>
   </body>
</html>
```

Salve o arquivo com a extensão .html ao executar o script no IE, o seguinte resultado é exibido na tela.

```
VBScript
VBScript
```

Com..End Com

A instrução With nos permite executar uma série de operações em um objeto especificado sem mencionar explicitamente o nome do objeto repetidas vezes.

Sintaxe

```
With (objectname)
   statement 1
   statement 2
   statement 3
   ...
   ...
   statement n
End With
```

Exemplo

Após a execução do seguinte script, o Winword é aberto e o texto especificado é inserido.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Msg =  "Vbscript" & vbCrLf & "Programming"
         Set objWord = CreateObject("Word.Application")
         objWord.Visible = True

         ' Objects methods are accessed without requaliyfying the objects again.'
         With objWord
            .Documents.Add
            .Selection.TypeText Msg
            .Selection.WholeStory
         End With

      </script>
   </body>
</html>
```

Aleatória

A instrução Randomize inicializa o gerador de números aleatórios, o que é útil para os desenvolvedores gerarem um número aleatório.

Sintaxe

```
Randomize [number]
```

Exemplo

Após a execução do seguinte script, o Winword é aberto e o texto especificado é inserido.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Dim MyValue
         Randomize
         MyValue = Int((100 * Rnd) + 1)    ' Generate random value between 1 and 100.
         MsgBox MyValue

      </script>
   </body>
</html>
```

Salve o script acima como HTML e, ao executar o script no IE, a seguinte saída será mostrada.

```
42
```

Página anterior Próxima página

Enter email for newsletter vai