目录

VBScript 教程

第一课 什么是 VBScript

Microsoft Visual Basic Scripting Edition 是程序开发语言 Visual Basic 家族的最新成员，它将灵活的 Script 应用于更广泛的领域，包括 Microsoft Internet Explorer 中的 Web 客户机 Script 和 Microsoft Internet Information Server 中的 Web 服务器 Script。

易学易用

如果您已了解 Visual Basic 或 Visual Basic for Applications，就会很快熟悉 VBScript。即使您没有学过 Visual Basic，只要学会 VBScript，就能够使用所有的 Visual Basic 语言进行程序设计。虽然您可以从本教程的几个 Web 页面中学习 VBscript，但是本教程并没有告诉您如何编程。要学习编程，请阅读由 Microsoft Press 出版的《Step by Step》。

ActiveX Script

VBScript 使用 ActiveX™ Script 与宿主应用程序对话。使用 ActiveX Script，浏览器和其他宿主应用程序不再需要每个 Script 部件的特殊集成代码。ActiveX Script 使宿主可以编译 Script、获取和调用入口点及管理开发者可用的命名空间。通过 ActiveX Script，语言厂商可以建立标准 Script 运行时语言。Microsoft 将提供 VBScript 的运行时支持。Microsoft 正在与多个 Internet 组一起定义 ActiveX Script 标准以使 Script 引擎可以互换。 ActiveX Script 可用在 Microsoft® Internet Explorer 和 Microsoft® Internet Information Server 中。

其他应用程序和浏览器中的 VBScript

作为开发者，您可以在您的产品中免费使用 VBScript 源实现程序。Microsoft 为 32 位 Windows® API、16 位 Windows API 和 Macintosh® 提供 VBscript 的二进制实现程序。VBScript 与 World Wide Web 浏览器集成在一起。VBScript 和 ActiveX Script 也可以在其他应用程序中作为普通 Script 语言使用。

第二课 在 HTML 页面中添加 VBscript 代码

SCRIPT 元素用于将 VBScript 代码添加到 HTML 页面中。

<SCRIPT> 标记

VBScript 代码写在成对的 <SCRIPT> 标记之间。例如，以下代码为一个测试传递日期的过程：

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

代码的开始和结束部分都有 <SCRIPT> 标记。LANGUAGE 属性用于指定所使用的 Script 语言。由于浏览器能够使用多种 Script 语言，所以必须在此指定所使用的 Script 语言。注意 CanDeliver 函数被嵌入在注释标记（<!-- 和 -->）中。这样能够避免不能识别 <SCRIPT> 标记的浏览器将代码显示在页面中。
因为以上示例是一个通用函数（不依赖于任何窗体控件），所以可以将其包含在页面的 HEAD 部分：

```
<HTML>
<HEAD>
<TITLE>订购</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Function CanDeliver(Dt)
```

```
CanDeliver = (CDate(Dt) - Now()) > 2
End Function
-->
</SCRIPT>
</HEAD>
<BODY>
...
```

SCRIPT 块可以出现在 HTML 页面的任何地方（BODY 或 HEAD 部分之中）。然而最好将所有的一般目标 Script 代码放在 HEAD 部分中，以使所有 Script 代码集中放置。这样可以确保在 BODY 部分调用代码之前所有 Script 代码都被读取并解码。

上述规则的一个值得注意的例外情况是，在窗体中提供内部代码以响应窗体中对象的事件。例如，以下示例在窗体中嵌入 Script 代码以响应窗体中按钮的单击事件：

```
<HTML>
<HEAD>
<TITLE>测试按钮事件</TITLE>
</HEAD>
<BODY>
<FORM NAME="Form1">
<INPUT TYPE="Button" NAME="Button1" VALUE="单击">
<SCRIPT FOR="Button1" EVENT="onClick" LANGUAGE="VBScript">
MsgBox "按钮被单击！"
</SCRIPT>
</FORM>
</BODY>
</HTML>
```

大多数 Script 代码在 Sub 或 Function 过程中，仅在其他代码要调用它时执行。然而，也可以将 VB Script 代码放在过程之外、SCRIPT 块之中。这类代码仅在 HTML 页面加载时执行一次。这样就可以在加载 Web 页面时初始化数据或动态地改变页面的外观。

第三课 VBScript 数据类型

什么是 VBScript 数据类型？

VBScript 只有一种数据类型，称为 Variant。Variant 是一种特殊的数据类型，根据使用的方式，它可以包含不同类别的信息。因为 Variant 是 VBScript 中唯一的数据类型，所以它也是 VBScript 中所有函数的返回值的数据类型。

最简单的 Variant 可以包含数字或字符串信息。Variant 用于数字上下文中时作为数字处理，用于字符串上下文中时作为字符串处理。这就是说，如果使用看起来象是数字的数据，则 VBScript 会假定其为数字并以适用于数字的方式处理。与此类似，如果使用的数据只可能是字符串，则 VBScript 将按字符串处理。当然，也可以将数字包含在引号 (" ") 中使其成为字符串。

Variant 子类型

除简单数字或字符串以外，Variant 可以进一步区分数值信息的特定含义。例如使用数值信息表示日期或时间。此类数据在与其他日期或时间数据一起使用时，结果也总是表示为日期或时间。当然，从 Boolean 值到浮点数，数值信息是多种多样的。Variant 包含的数值信息类型称为子类型。大多数情况下，可将所需的数据放进 Variant 中，而 Variant 也会按照最适用于其包含的数据的方式进行操作。

下表显示 Variant 包含的数据子类型：

子类型	描述
Empty	未初始化的 Variant。对于数值变量，值为 0；对于字符串变量，值为零长度字符串 (""）。
Null	不包含任何有效数据的 Variant。
Boolean	包含 True 或 False。
Byte	包含 0 到 255 之间的整数。
Integer	包含 -32,768 到 32,767 之间的整数。
Currency	-922,337,203,685,477.5808 到 922,337,203,685,477.5807。
Long	包含 -2,147,483,648 到 2,147,483,647 之间的整数。
Single	包含单精度浮点数，负数范围从 -3.402823E38 到 -1.401298E-45，正数范围从 1.401298E-45 到 3.402823E38。
Double	包含双精度浮点数，负数范围从 -1.79769313486232E308 到 -4.94065645841247E-324，正数范围从 4.94065645841247E-324 到 1.79769313486232E308。
Date (Time)	包含表示日期的数字，日期范围从公元 100 年 1 月 1 日到公元 9999 年 12 月 31 日。
String	包含变长字符串，最大长度可为 20 亿个字符。
Object	包含对象。
Error	包含错误号。

您可以使用转换函数来转换数据的子类型。另外，可使用 VarType 函数返回数据的 Variant 子类型。

第四课 VBScript 变量

什么是变量？

变量是一种使用方便的占位符，用于引用计算机内存地址，该地址可以存储 Script 运行时可更改的程序信息。例如，可以创建一个名为 ClickCount 的变量来存储用户单击 Web 页面上某个对象的次数。使用变量并不需要了解变量在计算机内存中的地址，只要通过变量名引用变量就可以查看或更改变量的值。在 VBScript 中只有一个基本数据类型，即 Variant，因此所有变量的数据类型都是 Variant。

声明变量

声明变量的一种方式是使用 Dim 语句、Public 语句和 Private 语句在 Script 中显式声明变量。例如：

Dim DegreesFahrenheit

声明多个变量时，使用逗号分隔变量。例如：

Dim Top, Bottom, Left, Right

另一种方式是通过直接在 Script 中使用变量名这一简单方式隐式声明变量。这通常不是一个好习惯，因为这样有时会由于变量名被拼错而导致在运行 Script 时出现意外的结果。因此，最好使用 Option Explicit 语句显式声明所有变量，并将其作为 Script 的第一条语句。

命名规则

变量命名必须遵循 VBScript 的标准命名规则。变量命名必须遵循：

·第一个字符必须是字母。
·不能包含嵌入的句点。
·长度不能超过 255 个字符。
·在被声明的作用域内必须唯一。

变量的作用域与存活期

变量的作用域由声明它的位置决定。如果在过程中声明变量，则只有该过程中的代码可以访问或更改变量值，此时变量具有局部作用域并被称为过程级变量。如果在过程之外声明变量，则该变量可以被 Script 中所有过程所识别，称为 Script 级变量，具有 Script 级作用域。

变量存在的时间称为存活期。Script 级变量的存活期从被声明的一刻起，直到 Script 运行结束。对于过程级变量，其存活期仅是该过程运行的时间，该过程结束后，变量随之消失。在执行过程时，局部变量是理想的临时存储空间。可以在不同过程中使用同名的局部变量，这是因为每个局部变量只被声明它的过程识别。

给变量赋值

创建如下形式的表达式给变量赋值：变量在表达式左边，要赋的值在表达式右边。例如：

```
B = 200
```

标量变量和数组变量

多数情况下，只需为声明的变量赋一个值。只包含一个值的变量被称为标量变量。有时候，将多个相关值赋给一个变量更为方便，因此可以创建包含一系列值的变量，称为数组变量。数组变量和标量变量是以相同的方式声明的，唯一的区别是声明数组变量时变量名后面带有括号 ( )。下例声明了一个包含 11 个元素的一维数组：

```
Dim A(10)
```

虽然括号中显示的数字是 10，但由于在 VBScript 中所有数组都是基于 0 的，所以这个数组实际上包含 11 个元素。在基于 0 的数组中，数组元素的数目总是括号中显示的数目加 1。这种数组被称为固定大小的数组。

在数组中使用索引为数组的每个元素赋值。从 0 到 10，将数据赋给数组的元素，如下所示：

```
A(0) = 256
A(1) = 324
A(2) = 100
. . .
A(10) = 55
```

与此类似，使用索引可以检索到所需的数组元素的数据。例如：

```
. . .
SomeVariable = A(8)
. . .
```

数组并不仅限于一维。数组的维数最大可以为 60（尽管大多数人不能理解超过 3 或 4 的维数）。声明多维数组时用逗号分隔括号中每个表示数组大小的数字。在下例中，MyTable 变量是一个有 6 行和 11 列的二维数组：

```
Dim MyTable(5, 10)
```

在二维数组中，括号中第一个数字表示行的数目，第二个数字表示列的数目。
也可以声明动态数组，即在运行 Script 时大小发生变化的数组。对数组的最初声明使用 Dim 语句或 ReDim 语句。但是对于动态数组，括号中不包含任何数字。例如：

```
Dim MyArray()
ReDim AnotherArray()
```

要使用动态数组，必须随后使用 ReDim 确定维数和每一维的大小。在下例中，ReDim 将动态数组的初始大小设置为 25，而后面的 ReDim 语句将数组的大小重新调整为 30，同时使用 Preserve 关键字在重新调整大小时保留数组的内容。

```
ReDim MyArray(25)
. . .
ReDim Preserve MyArray(30)
```

重新调整动态数组大小的次数是没有任何限制的，但是应注意：将数组的大小调小时，将会丢失被删除元素的数据。

第五课 VBScript 常数

什么是常数？

常数是具有一定含义的名称，用于代替数字或字符串，其值从不改变。VBScript 定义了许多固有常数。详细信息，请参阅 VBScript 语言参考。

创建常数

您可以使用 Const 语句在 VBScript 中创建用户自定义常数。使用 Const 语句可以创建名称具有一定含义的字符串型或数值型常数，并给它们赋原义值。例如：

```
Const MyString = "这是一个字符串。"
Const MyAge = 49
```

请注意字符串文字包含在两个引号 (" ") 之间。这是区分字符串型常数和数值型常数的最明显的方法。日期文字和时间文字包含在两个井号 (#) 之间。例如：

```
Const CutoffDate = #6-1-97#
```

最好采用一个命名方案以区分常数和变量。这样可以避免在运行 Script 时对常数重新赋值。例如，可以使用“vb”或“con”作常数名的前缀，或将常数名的所有字母大写。将常数和变量区分开可以在开发复杂的 Script 时避免混乱。

第六课 VBScript 运算符

VBScript 有一套完整的运算符，包括算术运算符、比较运算符、连接运算符和逻辑运算符。

运算符优先级

当表达式包含多个运算符时，将按预定顺序计算每一部分，这个顺序被称为运算符优先级。可以使用括号越过这种优先级顺序，强制首先计算表达式的某些部分。运算时，总是先执行括号中的运算符，然后再执行括号外的运算符。但是，在括号中仍遵循标准运算符优先级。

当表达式包含多种运算符时，首先计算算术运算符，然后计算比较运算符，最后计算逻辑运算符。所有比较运算符的优先级相同，即按照从左到右的顺序计算比较运算符。算术运算符和逻辑运算符的优先级如下所示：

算术运算符		比较运算符		逻辑运算符	
描述	**符号**	**描述**	**符号**	**描述**	**符号**
求幂	^	等于	=	逻辑非	Not
负号	-	不等于	<>	逻辑与	And
乘	*	小于	<	逻辑或	Or
除	/	大于	>	逻辑异或	Xor
整除	\	小于等于	<=	逻辑等价	Eqv
求余	Mod	大于等于	>=	逻辑隐含	Imp
加	+	对象引用比较	Is		
减	-				
字符串连接	&				

当乘号与除号同时出现在一个表达式中时，按从左到右的顺序计算乘、除运算符。同样当加与减同时出现在一个表达式中时，按从左到右的顺序计算加、减运算符。

字符串连接 (&) 运算符不是算术运算符，但是在优先级顺序中，它排在所有算术运算符之后和所有比较运算符之前。Is 运算符是对象引用比较运算符。它并不比较对象或对象的值，而只是进行检查，判断两个对象引用是否引用同一个对象。

第七课 使用条件语句

控制程序执行

使用条件语句和循环语句可以控制 Script 的流程。使用条件语句可以编写进行判断和重复操作的 VBScript 代码。在 VBScript 中可使用以下条件语句：

· If...Then...Else 语句
· Select Case 语句

使用 If...Then...Else 进行判断

If...Then...Else 语句用于计算条件是否为 True 或 False，并且根据计算结果指定要运行的语句。通常，条件是使用比较运算符对值或变量进行比较的表达式。有关比较运算符的详细信息，请参阅比较运算符。If...Then...Else 语句可以按照需要进行嵌套。

条件为 True 时运行语句

要在条件为 True 时运行单行语句，可使用 If...Then...Else 语句的单行语法。下例示范了单行语法。请注意此例省略了关键字 Else。

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

要运行多行代码，必须使用多行（或块）语法。多行（或块）语法包含 End If 语句，如下所示：

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

条件为 True 和 False 时分别运行某些语句

可以使用 If...Then...Else 语句定义两个可执行语句块：条件为 True 时运行某一语句块，条件为 False 时运行另一语句块。

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

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

对多个条件进行判断

If...Then...Else 语句的一种变形允许您从多个条件中选择，即添加 ElseIf 子句以扩充 If...Then...Else 语句的功能，使您可以控制基于多种可能的程序流程。例如：

```
Sub ReportValue(value)
If value = 0 Then
MsgBox value
ElseIf value = 1 Then
MsgBox value
ElseIf value = 2 then
Msgbox value
Else
Msgbox "数值超出范围！"
End If
```

可以添加任意多个 ElseIf 子句以提供多种选择。使用多个 ElseIf 子句经常会变得很累赘。在多个条件中进行选择的更好方法是使用 Select Case 语句。

使用 Select Case 进行判断

Select Case 结构提供了 If...Then...ElseIf 结构的一个变通形式，可以从多个语句块中选择执行其中的一个。Select Case 语句提供的功能与 If...Then...Else 语句类似，但是可以使代码更加简练易读。

Select Case 结构在其开始处使用一个只计算一次的简单测试表达式。表达式的结果将与结构中每个 Case 的值比较。如果匹配，则执行与该 Case 关联的语句块：

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

```
ValidateAMEXCOAccount
Case Else
DisplayUnknownImage
PromptAgain
End Select
```

请注意 Select Case 结构只计算开始处的一个表达式（只计算一次），而 If...Then...ElseIf 结构计算每个 ElseIf 语句的表达式，这些表达式可以各不相同。仅当每个 ElseIf 语句计算的表达式都相同时，才可以使用 Select Case 结构代替 If...Then...ElseIf 结构。

第八课 使用循环语句

使用循环重复执行代码

循环用于重复执行一组语句。循环可分为三类：一类在条件变为 False 之前重复执行语句，一类在条件变为 True 之前重复执行语句，另一类按照指定的次数重复执行语句。

在 VBScript 中可使用下列循环语句：

·Do...Loop： 当（或直到）条件为 True 时循环。
·While...Wend： 当条件为 True 时循环。
·For...Next： 指定循环次数，使用计数器重复运行语句。
·For Each...Next： 对于集合中的每项或数组中的每个元素，重复执行一组语句。

使用 Do 循环

可以使用 Do...Loop 语句多次（次数不定）运行语句块。当条件为 True 时或条件变为 True 之前，重复执行语句块。

当条件为 True 时重复执行语句

While 关键字用于检查 Do...Loop 语句中的条件。有两种方式检查条件：在进入循环之前检查条件（如下面的 ChkFirstWhile 示例）；或者在循环至少运行完一次之后检查条件（如下面的 ChkLastWhile 示例）。在 ChkFirstWhile 过程中，如果 myNum 的初始值被设置为 9 而不是 20，则永远不会执行循环体中的语句。在 ChkLastWhile 过程中，循环体中的语句只会执行一次，因为条件在检查时已经为 False。

```
Sub ChkFirstWhile()
Dim counter, myNum
counter = 0
myNum = 20
Do While myNum > 10
```

```
myNum = myNum - 1
counter = counter + 1
Loop
MsgBox "循环重复了 " & counter & " 次。"
End Sub

Sub ChkLastWhile()
Dim counter, myNum
counter = 0
myNum = 9
Do
myNum = myNum - 1
counter = counter + 1
Loop While myNum > 10
MsgBox "循环重复了 " & counter & " 次。"
End Sub
```

重复执行语句直到条件变为 True

Until 关键字用于检查 Do...Loop 语句中的条件。有两种方式检查条件：在进入循环之前检查条件（如下面的 ChkFirstUntil 示例）；或者在循环至少运行完一次之后检查条件（如下面的 ChkLastUntil 示例）。只要条件为 False，就会进行循环。

```
Sub ChkFirstUntil()
Dim counter, myNum
counter = 0
myNum = 20
Do Until myNum = 10
myNum = myNum - 1
counter = counter + 1
Loop
MsgBox "循环重复了 " & counter & " 次。"
End Sub

Sub ChkLastUntil()
Dim counter, myNum
counter = 0
myNum = 1
Do
myNum = myNum + 1
counter = counter + 1
```

```
Loop Until myNum = 10
MsgBox "循环重复了 " & counter & " 次。"
End Sub
```

退出循环

Exit Do 语句用于退出 Do...Loop 循环。因为通常只是在某些特殊情况下要退出循环（例如要避免死循环），所以可在 If...Then...Else 语句的 True 语句块中使用 Exit Do 语句。如果条件为 False，循环将照常运行。

在下面的示例中，myNum 的初始值将导致死循环。If...Then...Else 语句检查此条件，防止出现死循环。

```
Sub ExitExample()
Dim counter, myNum
counter = 0
myNum = 9
Do Until myNum = 10
myNum = myNum - 1
counter = counter + 1
If myNum < 10 Then Exit Do
Loop
MsgBox "循环重复了 " & counter & " 次。"
End Sub
```

使用 While...Wend

While...Wend 语句是为那些熟悉其用法的用户提供的。但是由于 While...Wend 缺少灵活性，所以建议最好使用 Do...Loop 语句。

使用 For...Next

For...Next 语句用于将语句块运行指定的次数。在循环中使用计数器变量，该变量的值随每一次循环增加或减少。

例如，下面的示例将过程 MyProc 重复执行 50 次。For 语句指定计数器变量 x 及其起始值与终止值。Next 语句使计数器变量每次加 1。

```
Sub DoMyProc50Times()
Dim x
For x = 1 To 50
MyProc
```

```
Next
End Sub
```

关键字 Step 用于指定计数器变量每次增加或减少的值。在下面的示例中，计数器变量 j 每次加 2。循环结束后，total 的值为 2、4、6、8 和 10 的总和。

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

要使计数器变量递减，可将 Step 设为负值。此时计数器变量的终止值必须小于起始值。在下面的示例中，计数器变量 myNum 每次减 2。循环结束后，total 的值为 16、14、12、10、8、6、4 和 2 的总和。

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

Exit For 语句用于在计数器达到其终止值之前退出 For...Next 语句。因为通常只是在某些特殊情况下（例如在发生错误时）要退出循环，所以可以在 If...Then...Else 语句的 True 语句块中使用 Exit For 语句。如果条件为 False，循环将照常运行。

使用 For Each...Next

For Each...Next 循环与 For...Next 循环类似。For Each...Next 不是将语句运行指定的次数，而是对于数组中的每个元素或对象集合中的每一项重复一组语句。这在不知道集合中元素的数目时非常有用。

在以下示例中，Dictionary 对象的内容用于将文本分别放置在多个文本框中：

```
<HTML>
<HEAD><TITLE>窗体与元素</TITLE></HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChange_onClick
```

```
Dim d '创建一个变量
Set d = CreateObject("Scripting.Dictionary")
d.Add "0", "Athens" '添加键和项目
d.Add "1", "Belgrade"
d.Add "2", "Cairo"

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

<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Button" NAME="cmdChange" VALUE="单击此处"><p>
</FORM>
</CENTER>
</BODY>
</HTML>
```

第九课 VBScript 过程

过程分类

在 VBScript 中，过程被分为两类：Sub 过程和 Function 过程。

Sub 过程

Sub 过程是包含在 Sub 和 End Sub 语句之间的一组 VBScript 语句，执行操作但不返回值。Sub 过程可以使用参数（由调用过程传递的常数、变量或表达式）。如果 Sub 过程无任何参数，则 Sub 语句必须包含空括号 ()。

下面的 Sub 过程使用两个固有的（或内置的）VBScript 函数，即 MsgBox 和 InputBox，来提示用户输入信息。然后显示根据这些信息计算的结果。计算由使用 VBScript 创建的 Function 过程完成。此过程在以下讨论之后演示。

```
Sub ConvertTemp()
temp = InputBox("请输入华氏温度。", 1)
MsgBox "温度为 " & Celsius(temp) & " 摄氏度。"
End Sub
```

Function 过程

Function 过程是包含在 Function 和 End Function 语句之间的一组 VBScript 语句。Function 过程与 Sub 过程类似，但是 Function 过程可以返回值。Function 过程可以使用参数（由调用过程传递的常数、变量或表达式）。如果 Function 过程无任何参数，则 Function 语句必须包含空括号 ()。Function 过程通过函数名返回一个值，这个值是在过程的语句中赋给函数名的。Function 返回值的数据类型总是 Variant。

在下面的示例中，Celsius 函数将华氏度换算为摄氏度。Sub 过程 ConvertTemp 调用此函数时，包含参数值的变量被传递给函数。换算结果返回到调用过程并显示在消息框中。

```
Sub ConvertTemp()
temp = InputBox("请输入华氏温度。", 1)
MsgBox "温度为 " & Celsius(temp) & " 摄氏度。"
End Sub

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

过程的数据进出

给过程传递数据的途径是使用参数。参数被作为要传递给过程的数据的占位符。参数名可以是任何有效的变量名。使用 Sub 语句或 Function 语句创建过程时，过程名之后必须紧跟括号。括号中包含所有参数，参数间用逗号分隔。例如，在下面的示例中，fDegrees 是传递给 Celsius 函数的值的占位符：

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

要从过程获取数据，必须使用 Function 过程。请记住，Function 过程可以返回值；Sub 过程不返回值。

在代码中使用 Sub 和 Function 过程

调用 Function 过程时，函数名必须用在变量赋值语句的右端或表达式中。例如：

```
Temp = Celsius(fDegrees)
```
或
```
MsgBox "温度为 " & Celsius(fDegrees) & " 摄氏度。"
```

调用 Sub 过程时，只需输入过程名及所有参数值，参数值之间使用逗号分隔。不需使用 Call 语句，但如果使用了此语句，则必须将所有参数包含在括号之中。

下面的示例显示了调用 MyProc 过程的两种方式。一种使用 Call 语句；另一种则不使用。两种方式效果相同。

```
Call MyProc(firstarg, secondarg)
MyProc firstarg, secondarg
```

请注意当不使用 Call 语句进行调用时，括号被省略。

第十课 VBScript 编码约定

什么是编码约定？

编码约定是帮助您使用 Microsoft Visual Basic Scripting Edition 编写代码的一些建议。编码约定包含以下内容：

·对象、变量和过程的命名约定
·注释约定
·文本格式和缩进指南

使用一致的编码约定的主要原因是使 Script 或 Script 集的结构和编码样式标准化，这样代码易于阅读和理解。使用好的编码约定可以使源代码明白、易读、准确，更加直观且与其他语言约定保持一致。

常数命名约定

VBScript 的早期版本不允许创建用户自定义常数。如果要使用常数，则常数以变量的方式实现，且全部字母大写以和其他变量区分。常数名中的多个单词用下划线 (_) 分隔。例如：

```
USER_LIST_MAX
NEW_LINE
```

这种标识常数的方法依旧可行，但您还可以选择其他方案，用 Const 语句创建真正的常数。这个
约定使用大小写混合的格式，并以“con”作为常数名的前缀。例如：

```
conYourOwnConstant
```

变量命名约定

出于易读和一致性的目的，请在 VBScript 代码中使用以下变量命名约定：

子类型	前缀	示例
Boolean	bln	blnFound
Byte	byt	bytRasterData
Date (Time)	dtm	dtmStart
Double	dbl	dblTolerance
Error	err	errOrderNum
Integer	int	intQuantity
Long	lng	lngDistance
Object	obj	objCurrent
Single	sng	sngAverage
String	str	strFirstName

变量作用域

变量应定义在尽量小的作用域中。VBScript 变量的作用域如下所示：

作用域	声明变量处	可见性
过程级	事件、函数或子过程	在声明变量的过程中可见
Script 级	HTML 页面的 HEAD 部分，任何过程之外	在 Script 的所有过程中可见

变量作用域前缀

随着 Script 代码长度的增加，有必要快速区分变量的作用域。在类型前缀前面添加一个单字符前缀可以实现这一点，而不致使变量名过长。

作用域	前缀	示例

过程级	无	dblVelocity
Script 级	s	sblnCalcInProgress

描述性变量名和过程名

变量名或过程名的主体应使用大小写混合格式，并且尽量完整地描述其目的。另外，过程名应以动词开始，例如 InitNameArray 或 CloseDialog。

对于经常使用的或较长的名称，推荐使用标准缩写以使名称保持在适当的长度内。通常多于 32 个字符的变量名会变得难以阅读。使用缩写时，应确保在整个 Script 中保持一致。例如，在一个 Script 或 Script 集中随意切换 Cnt 和 Count 将造成混乱。

对象命名约定

下表列出了 VBScript 中可能用到的对象命名约定（推荐）：

对象类型	前缀	示例
3D 面板	pnl	pnlGroup
动画按钮	ani	aniMailBox
复选框	chk	chkReadOnly
组合框、下拉列表框	cbo	cboEnglish
命令按钮	cmd	cmdExit
公共对话框	dlg	dlgFileOpen
框架	fra	fraLanguage
水平滚动条	hsb	hsbVolume
图像	img	imgIcon
标签	lbl	lblHelpMessage
直线	lin	linVertical
列表框	lst	lstPolicyCodes
旋钮	spn	spnPages
文本框	txt	txtLastName

垂直滚动条	vsb	vsbRate
滑块	sld	sldScale

代码注释约定

所有过程的开始部分都应有描述其功能的简要注释。这些注释并不描述细节信息（如何实现功能），这是因为细节有时要频繁更改。这样就可以避免不必要的注释维护工作以及错误的注释。细节信息由代码本身及必要的内部注释来描述。

当传递给过程的参数的用途不明显，或过程对参数的取值范围有要求时，应加以说明。如果过程改变了函数和变量的返回值（特别是通过参数引用来改变），也应在过程的开始部分描述该返回值。

过程开始部分的注释应包含以下区段标题。相关样例，请参阅后面的“格式化代码”部分。

区段标题	**注释内容**
目的	过程的功能（不是实现功能的方法）。
假设	其状态影响此过程的外部变量、控件或其他元素的列表。
效果	过程对每个外部变量、控件或其他元素的影响效果的列表。
输入	每个目的不明显的参数的解释。每个参数都应占据单独一行并有其内部注释。
返回	返回值的解释。

请记住以下几点：

·每个重要的变量声明都应有内部注释，描述变量的用途。
·应清楚地命名变量、控件和过程，仅在说明复杂细节时需要内部注释。
·应在 Script 的开始部分包含描述该 Script 的概述，列举对象、过程、运算法则、对话框和其他系统从属物。有时一段描述运算法则的假码是很有用的。

格式化代码

应尽可能多地保留屏空间，但仍允许用代码格式反映逻辑结构和嵌套。以下为几点提示：

·标准嵌套块应缩进 4 个空格。
·过程的概述注释应缩进 1 个空格。
·概述注释后的最高层语句应缩进 4 个空格，每一层嵌套块再缩进 4 个空格。例如：

```
'*************************************************************
' 目的： 返回指定用户在 UserList 数组中第一次出现的位置。
' 输入： strUserList(): 所查找的用户列表。
' strTargetUser: 要查找的用户名。
' 返回： strTargetUser 在 strUserList 数组中第一次出现时的索引。
' 如果目标用户未找到，返回 -1。
'*************************************************************

Function intFindUser (strUserList(), strTargetUser)
Dim i ' 循环计数器。
Dim blnFound ' 发现目标的标记。
intFindUser = -1
i = 0 ' 初始化循环计数器。
Do While i <= Ubound(strUserList) and Not blnFound
If strUserList(i) = strTargetUser Then
blnFound = True ' 标记设为 True。
intFindUser = i ' 返回值设为循环计数器。
End If
i = i + 1 ' 循环计数器加 1。
Loop
End Function
```

第十一课 深入 VBScript

学习高级 VBScript 技术的最快方法是阅读大量的样例。同时更好地理解对象模型有助于深入学习 VBScript。

您可以从以下部分开始学习：

·ActiveX™ 控件常见问题解答

·页面样例

·热点链接页面

第十二课 VBScript 页面的简单样例

一个简单页面

使用 Microsoft? Internet Explorer 可以查看用以下 HTML 代码制作的页面。如果单击页面上的按钮，可看到 VBScript 的运行结果。

```
<HTML>
<HEAD><TITLE>一个简单首页</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Button1_onClick
MsgBox "Mirabile visu."
End Sub
-->
</SCRIPT>
</HEAD>
<BODY>
<H3>一个简单首页</H3><HR>
<FORM><INPUT NAME="Button1" TYPE="BUTTON" VALUE="单击此处"></FORM>
</BODY>
</HTML>
```

结果虽然有点简单：一个对话框显示一个拉丁短语（意为“看起来非常漂亮”）。然而这段代码实际上作了许多事情。

当 Internet Explorer 读取页面时，找到 <SCRIPT> 标记，识别出 VBScript 代码并保存代码。单击按钮时，Internet Explorer 使按钮与代码连接，并运行该过程。

<SCRIPT> 标记中的 Sub 过程是一个事件过程。过程名包含两部分：一部分为按钮名，即 Button1（从 <INPUT> 标记中的 NAME 属性获取），另一部分为事件名，即 *onClick*。两部分由下划线 (_) 连接。单击按钮时，Internet Explorer 查找并运行相应的事件过程，即 Button1_*onClick*。

Internet Explorer 在 Internet Explorer Scripting Object Model 文档中定义了可用于窗体控件的事件。

页面也可以使用控件与过程的组合。VBScript 与窗体显示了控件之间的一些简单交互作用。

向事件附加代码的其他方法

上述的方法也许是最简单和最常用的，但也可以使用另外两种方法向事件附加 VBScript 代码。一种方法是在定义控件的标记中添加较短的内部代码。例如在单击按钮时，下面的 <INPUT> 标记执行与前面示例相同的操作：

```
<INPUT NAME="Button1" TYPE="BUTTON"
VALUE="单击此处" onClick='MsgBox "Mirabile visu."'>
```

请注意函数调用包含在单引号中，MsgBox 函数的字符串包含在双引号中。只要用冒号 (:) 分隔语句，就可以使用多条语句。

另一种方法是在 <SCRIPT> 标记中指定特定的控件和事件：

```
<SCRIPT LANGUAGE="VBScript" EVENT="onClick" FOR="Button1">
<!--
MsgBox "Mirabile visu."
-->
</SCRIPT>
```

由于 <SCRIPT> 标记指定了事件和控件，所以不需要再用 Sub 和 End Sub 语句。

第十三课 VBScript 与窗体

简单验证

使用 Visual Basic Scripting Edition，您可以完成通常要在服务器上进行的大量窗体处理工作，也可以完成不能在服务器上进行的工作。

这是一个简单的客户端验证的样例。HTML 代码的结果是一个文本框和一个按钮。如果使用 Microsoft? Internet Explorer 查看用以下代码制作的页面，您会看到一个旁边带有按钮的小文本框。

```
<HTML>
<HEAD><TITLE>简单验证</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Submit_onClick
Dim TheForm
Set TheForm = Document.ValidForm
If IsNumeric(TheForm.Text1.Value) Then
If TheForm.Text1.Value < 1 Or TheForm.Text1.Value > 10 Then
MsgBox "请输入一个 1 到 10 之间的数字。"
Else
MsgBox "谢谢。"
End If
Else
MsgBox "请输入一个数字。"
End If
End Sub
```

```
-->
</SCRIPT>
</HEAD>
<BODY>
<H3>简单验证</H3><HR>
<FORM NAME="ValidForm">
请输入一个 1 到 10 之间的数字：
<INPUT NAME="Text1" TYPE="TEXT" SIZE="2">
<INPUT NAME="Submit" TYPE="BUTTON" VALUE="提交">
</FORM>
</BODY>
</HTML>
```

这个文本框与 VBScript 页面的简单样例中示例的不同之处在于文本框的 Value 属性被用于检查输入值。要使用文本框的 Value 属性，代码必须引用文本框的名称。

每次引用文本框时都应写出全称，即 Document.ValidForm.Text1。但是，当多次引用窗体控件时，可以按照以下步骤操作：首先声明一个变量，然后使用 Set 语句将窗体 Document.ValidForm 赋给变量 TheForm，这样就能使用 TheForm.Text1 引用文本框。常规的赋值语句（例如 Dim）在这里无效，必须使用 Set 来保持对对象的引用。

使用数字

请注意以上示例直接检测输入值是否是一个数字：使用 IsNumeric 函数确定文本框中的字符串是否是一个数字。虽然 VBScript 能够自动转换字符串和数字，但检测用户输入值的数据子类型，并且在必要时使用转换函数始终是一个好的习惯。在用文本框的 Value 属性进行加法运算时，应将它显式地转换为数字，这是因为加号 (+) 操作符不但可进行加法操作，而且可进行字符串连接操作。例如，如果 Text1 中包含“1”，Text2 中包含“2”，您将会看到以下结果：

```
A = Text1.Value + Text2.Value ' A 为“12”
A = CDbl(Text1.Value) + Text2.Value ' A 为 3
```

验证后将数据传递回服务器

简单验证样例使用的是普通按钮控件。如果使用 Submit 控件，所有数据都会被立即传送到服务器，示例将不会看到数据来进行检查。避免使用 Submit 控件使您可以检查数据，但不能向服务器提交数据。如果要提交数据则需要再添加一行代码，如下所示：

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Submit_onClick
```

```
Dim TheForm
Set TheForm = Document.ValidForm
If IsNumeric(TheForm.Text1.Value) Then
If TheForm.Text1.Value < 1 Or TheForm.Text1.Value > 10 Then
MsgBox "请输入一个 1 到 10 之间的数字。"
Else
MsgBox "谢谢。"
TheForm.Submit ' 数据输入正确，传递到服务器。
End If
Else
MsgBox "请输入一个数字。"
End If
End Sub
-->
</SCRIPT>
```

在数据输入正确时，代码调用窗体对象的 Submit 方法，将数据传递到服务器。除非在数据被传递到服务器之前判断其正误，否则服务器将处理数据，而不论其正确与否。您可以在 Internet Explorer Script Object Model 页面上找到关于 Submit 方法和其他方法的全部信息。

到目前为止，您只看到了标准 HTML <FORM> 对象。Internet Explorer 还可以使您利用 ActiveX? 控件（以前称为 OLE 控件）和 Java? 对象的全部功能创建页面。

第十四课 在 VBScript 中使用对象

使用对象

无论使用的是 ActiveX? 控件（以前称为 OLE 控件）还是 Java? 对象，Microsoft Visual Basic Scripting Edition 和 Microsoft? Internet Explorer 都以相同的方式处理它们。如果您使用的是 Internet Explorer 并且 ActiveX 库中安装了这些控件，就会看到由以下代码制作的页面。

<OBJECT> 标记用来包含对象，<PARAM> 标记用来设置对象属性的初始值。使用 <PARAM> 标记类似于在 Visual Basic 中设置窗体控件的初始属性值。例如，以下代码使用 <OBJECT> 和 <PARAM> 标记将 ActiveX 标签控件添加到页面中：

```
<OBJECT
classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
id=lblActiveLbl
width=250
height=250
align=left
```

```
hspace=20
vspace=0
>
<PARAM NAME="Angle" VALUE="90">
<PARAM NAME="Alignment" VALUE="4">
<PARAM NAME="BackStyle" VALUE="0">
<PARAM NAME="Caption" VALUE="一个简单标签">
<PARAM NAME="FontName" VALUE="宋体">
<PARAM NAME="FontSize" VALUE="20">
<PARAM NAME="FontBold" VALUE="1">
<PARAM NAME="FrColor" VALUE="0">
</OBJECT>
```

象对任何窗体控件一样，可以获取属性、设置属性和调用方法。例如，以下代码包含 <FORM> 控件，可用其对标签控件的两个属性进行操作：

```
<FORM NAME="LabelControls">
<INPUT TYPE="TEXT" NAME="txtNewText" SIZE=25>
<INPUT TYPE="BUTTON" NAME="cmdChangeIt" VALUE="更改文本">
<INPUT TYPE="BUTTON" NAME="cmdRotate" VALUE="旋转标签">
</FORM>
```

通过定义过的窗体，cmdChangeIt 按钮的事件过程可更改标签文本：

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChangeIt_onClick
Dim TheForm
Set TheForm = Document.LabelControls
lblActiveLbl.Caption = TheForm.txtNewText.Value
End Sub
-->
</SCRIPT>
```

代码将对控件和值的引用限定在窗体中，这与简单验证示例中的代码类似。

ActiveX 库中有多个 ActiveX? 控件可用于 Internet Explorer。您可以在那里找到关于属性、方法和事件的全部信息，也可以在编程参考页面上找到关于控件类标识符 (CLSID) 的信息。另外还可以在 Internet Explorer 4.0 Author's Guide and HTML Reference 页面上找到有关 <OBJECT> 标记的详细信息。

注意 Internet Explorer 的早期版本要求用大括号 ({}) 将 classid 属性括起来，不符合 W3C 规格。在当前版本中使用大括号则会产生“此页使用了过期版本的 <OBJECT> 标记”信息。

DateAdd 和 DateDiff 函数的第一个参数

首先参看

DateDiff 函数

DateDiff 函数返回两个日期之间的时间间隔。DateDiff(interval, date1, date2])DateDiff 函数的语法有以下参数：参数 interval 必选项。字符串表达式表示用于计算 date1 和 date2 之间的时间间隔。有关数值，请参阅“设置”部分。 date1, date2 必选项。日期表达式。用于计算的两个日期。Firstdayofweek 可选项。指定星期中第一天的常数。如果没有指定，则默认为星期日。有关数值，请参阅“设置”部分。 Firstwee

2006-4-23 23:03:59

DateAdd 函数

DateAdd 函数返回已添加指定时间间隔的日期。DateAdd(interval, number, date)参数 interval 必选项。字符串表达式，表示要添加的时间间隔。有关数值，请参阅“设置”部分。 number 必选项。数值表达式，表示要添加的时间间隔的个数。数值表达式可以是正数（得到未来的日期）或负数（得到过去的日期）。 date 必选项。Variant 或要添加 interval 的表示日期的文字。 设置 interval 参数可以有以下值：设置 描述 yyyy 年 q 季度 m

2006-4-23 23:02:56

DateAdd 和 DateDiff 的第一个参数
DateAdd 返回一个日期加上特定时间间隔后的值。
语法：DateAdd(interval, number, date)
interval 表示时间单位，即指示 number 是表示年，还是月，还是分，还是其它的，如下：
yyyy 年
q 季度
m 月
y 一年的日数
d 日
w 一周的日数
ww 周
h 小时
n 分钟
s 秒
不好理解的是：y、w，开始我以为 y 的单位是 365 天，w 的单位是 7 天。其实不是的，细读了微软参考并作了测试后发现，这其中 y、w、d 是同意义的，表示天数。
另外，DateDiff 的第一个参数中 y 和 d 是同意义的，w 不再与 d 同意义，而是表示相隔多少个 7 天，这与 ww 不同。假如今天是一周的第一天，至于星期几是一周的第一天得看计算机设置和第四个参数了，那么今天与昨天相隔 0 个 w，相隔 1 个 ww，因为今天与昨天相差不足 1 个 7 天，但今天已经是另外

一周了。
很难想像吧，但事实就是这样，总结一下。

DateAdd：y、w、d 同意义，都表示天数。
DateDiff：y、d 同意义，都表示天数；w 表示多少个 7 天，ww 表示多少周。

Rnd 函数 返回一个随机数

返回一个随机数。

Rnd[(number)]
number 参数可以是任意有效的数值表达式。

说明
Rnd 函数返回一个小于 1 但大于或等于 0 的值。number 的值决定了 Rnd 生成随机数的方式：

aspxuexi.com table	
如果 number 为	Rnd 生成
小于零	每次都相同的值，使用 number 作为种子。
大于零	序列中的下一个随机数。
等于零	最近生成的数。
省略	序列中的下一个随机数。

因每一次连续调用 Rnd 函数时都用序列中的前一个数作为下一个数的种子，所以对于任何最初给定的种子都会生成相同的数列。

在调用 Rnd 之前，先使用无参数的 Randomize 语句初始化随机数生成器，该生成器具有基于系统计时器的种子。

要产生指定范围的随机整数，请使用以下公式：

Int((upperbound - lowerbound + 1) * Rnd + lowerbound)
这里， upperbound 是此范围的上界，而 lowerbound 是此范围内的下界。

注意 要重复随机数的序列，请在使用数值参数调用 Randomize 之前，立即用负值参数调用 Rnd。使用同样 number 值的 Randomize 不能重复先前的随机数序列。

RGB 函数:返回代表 RGB 颜色值的整数

RGB(red, green, blue)
参数
red

必选项。0 到 255 间的整数，代表颜色中的红色成分。
green
必选项。0 到 255 间的整数，代表颜色中的绿色成分。
blue
必选项。0 到 255 间的整数，代表颜色中的蓝色成分。
说明
接受颜色说明的应用程序方法和属性，要求该说明以整数代表 RGB 颜色值。RGB 颜色值指定了红色、绿色、蓝色的相对强度，三色组合形成显示的特定颜色。

低字节值表示红色，中字节值表示绿色，高字节值表示蓝色。

对于要求反转字节顺序的应用程序，下面函数在反转字节顺序下提供相同信息：

```
Function RevRGB(red, green, blue)
RevRGB= CLng(blue + (green * 256) + (red * 65536))
End Function
```

RGB 函数中任一超过 255 的参数都假定为 255。

vbscript Replace 函数

返回字符串，其中指定数目的某子字符串被替换为另一个子字符串。

```
Replace(expression, find, replacewith[, compare[, count[, start]]])
```

参数

expression

必选项。字符串表达式包含要替代的子字符串。

Find

必选项。被搜索的子字符串。

Replacewith

必选项。用于替换的子字符串。

Start

可选项。expression 中开始搜索子字符串的位置。如果省略，默认值为 1。在和 count 关联时必须用

count

可选项。执行子字符串替换的数目。如果省略，默认值为 -1，表示进行所有可能的替换。在和 start 关联时必须用。

Compare

可选项。指示在计算子字符串时使用的比较类型的数值。有关数值，请参阅“设置”部分。如果省略,缺省值为 0 ,这意味着必须进行二进制比较。

设置

compare 参数可以有以下值：

aspxuexi.com table		
常数	值	描述
vbBinaryCompare	0	执行二进制比较。
vbTextCompare	1	执行文本比较。

返回值

Replace 返回以下值：

aspxuexi.com 表格	
如果	Replace 返回
expression 为零长度	零长度字符串 ("")。
expression 为 Null	错误。
find 为零长度	expression 的副本。
replacewith 为零长度	expression 的副本，其中删除了所有由 find 参数指定的内容。
start > Len(expression)	零长度字符串。
count 为 0	expression 的副本。

说明

Replace 函数的返回值是经过替换(从由 start 指定的位置开始到 expression 字符串的结尾)后的字符串，而不是原始字符串从开始至结尾的副本。

下面的示例利用 Replace 函数返回字符串：

```
Dim MyString
MyString = Replace("XXpXXPXXp", "p", "Y") '二进制比较从字符串左端开始。返回 "XXYXXP
XXY"。
MyString = Replace("XXpXXPXXp", "p", "Y", '文本比较从第三个字符开始。返回 "YXXYXXY"。
3, , -1, 1)
```

vbscript InputBox 函数

InputBox 函数

在对话框中显示提示，等待用户输入文本或单击按钮，并返回文本框内容。

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

参数

prompt

字符串表达式，作为消息显示在对话框中。prompt 的最大长度大约是 1024 个字符，这取决于所使用的字

符的宽度。如果 prompt 中包含多个行，则可在各行之间用回车符 (Chr(13))、换行符 (Chr(10)) 或回车换行符的组合 (Chr(13) & Chr(10)) 以分隔各行。

Title

显示在对话框标题栏中的字符串表达式。如果省略 title，则应用程序的名称将显示在标题栏中。

Default

显示在文本框中的字符串表达式，在没有其它输入时作为默认的响应值。如果省略 default，则文本框为空。

Xpos

数值表达式，用于指定对话框的左边缘与屏幕左边缘的水平距离（单位为缇）。如果省略 xpos，则对话框会在水平方向居中。

Ypos

数值表达式，用于指定对话框的上边缘与屏幕上边缘的垂直距离（单位为缇）。如果省略 ypos，则对话框显示在屏幕垂直方向距下边缘大约三分之一处。

Helpfile

字符串表达式，用于标识为对话框提供上下文相关帮助的帮助文件。如果已提供 helpfile，则必须提供 context。

Context

数值表达式，用于标识由帮助文件的作者指定给某个帮助主题的上下文编号。如果已提供 context，则必须提供 helpfile。

说明

如果同时提供了 helpfile 和 context，就会在对话框中自动添加“帮助”按钮。

如果用户单击确定或按下 ENTER，则 InputBox 函数返回文本框中的内容。如果用户单击取消，则函数返回一个零长度字符串 ("")。

下面例子利用 InputBox 函数显示一输入框并且把字符串赋值给输入变量：

```
Dim Input
Input = InputBox("输入名字")
MsgBox ("输入：" & Input)
```

VBscript 脚本引擎 相关函数

返回一个代表当前使用的脚本程序语言的字符串。

ScriptEngine

返回值

ScriptEngine 函数可返回下列字符串：

字符串	描述

VBScript　表明当前使用的编写脚本引擎是 Microsoft® Visual Basic® Scripting Edition。

JScript®　表明当前使用的编写脚本引擎是 Microsoft JScript®。

VBA　表明当前使用的编写脚本引擎是 Microsoft Visual Basic for Applications。

说明

下面的示例利用 ScriptEngine 函数返回描述所用书写语言的字符串：

```
Function GetScriptEngineInfo
Dim s
s = "" ' 用必要的信息形成字符串。
s = ScriptEngine & " Version "
s = s & ScriptEngineMajorVersion & "."
s = s & ScriptEngineMinorVersion & "."
s = s & ScriptEngineBuildVersion
GetScriptEngineInfo = s '返回结果。
End Function
```

ScriptEngineBuildVersion 函数

请参阅

ScriptEngine 函数 | ScriptEngineMajorVersion 函数 | ScriptEngineMinorVersion 函数

要求

版本 2

返回使用的编写脚本引擎的生成版本号。

ScriptEngineBuildVersion

说明

返回值直接对应于所使用的 Scripting 程序语言的 DLL 文件中包含的版本信息。

下面的示例利用 ScriptEngineBuildVersion 函数返回创建的编写脚本引擎版本号：

```
Function GetScriptEngineInfo
Dim s
s = "" ' 用必要的信息形成字符串。
s = ScriptEngine & " Version "
s = s & ScriptEngineMajorVersion & "."
s = s & ScriptEngineMinorVersion & "."
s = s & ScriptEngineBuildVersion
```

```
GetScriptEngineInfo = s ’ 返回结果。
End Function
```

ScriptEngineMajorVersion 函数

请参阅

ScriptEngine 函数 | ScriptEngineBuildVersion 函数 | ScriptEngineMinorVersion 函数

要求

版本 2

返回使用的编写脚本引擎的主版本号。

ScriptEngineMajorVersion

说明

返回值直接对应于所使用的脚本程序语言中 DLL 文件包含的版本信息。

下面的示例利用 ScriptEngineMajorVersion 函数返回编写脚本引擎的版本号：

```
Function GetScriptEngineInfo
Dim s
s = "" ’ 用必要的信息形成字符串。
s = ScriptEngine & " Version "
s = s & ScriptEngineMajorVersion & "."
s = s & ScriptEngineMinorVersion & "."
s = s & ScriptEngineBuildVersion
GetScriptEngineInfo = s ’ 返回结果。
End Function
```

ScriptEngineMinorVersion 函数

请参阅

ScriptEngine 函数 | ScriptEngineBuildVersion 函数 | ScriptEngineMajorVersion 函数

要求

版本 2

返回使用的编写引擎引擎的次版本号。

ScriptEngineMinorVersion

说明

返回值直接对应于所使用的脚本程序语言中 DLL 文件包含的版本信息。

下面的示例利用 ScriptEngineMinorVersion 函数返回编写引擎的副版本号：

```
Function GetScriptEngineInfo
Dim s
s = "" ' 用必要的信息形成字符串。
s = ScriptEngine & " Version "
s = s & ScriptEngineMajorVersion & "."
s = s & ScriptEngineMinorVersion & "."
s = s & ScriptEngineBuildVersion
GetScriptEngineInfo = s ' 返回结果。
End Function
```