VBScript 1.1

Introduction

VBScript is a scripting language which can be used to modify Visual Basic forms and controls at design time.

To use VBScript, add VBSCRIPT.VBX to a Visual Basic project and use the VBScript window to load, save, and edit your scripts.   The VBScript window will remain onscreen until VBSCRIPT.VBX is removed from the project - Alt+F4 and the Exit command in the File menu will minimize the VBScript window, but the window cannot be closed while VBSCRIPT.VBX remains in the project.

When your script is completed, issue the Execute Script command (in the File menu).   VBScript will run the script and make the desired modifications to the forms and controls in the current project.   If an error in the script is found, script execution will stop after the error is reported, and the cursor in the editing window will move to the approximate location of the error.   A script can be stopped while it is running by pressing the Control and Shift keys together.

The Undo Script command (in the File menu) will undo all changes to forms which were made by the most recently executed script.   To reverse the effect of Undo Script, use the Redo Script command. Undo Script can undo virtually all modifications made by scripts (including the creation and deletion of controls).   Undo Script will undo any changes made by a script which contains an error or is aborted with Control-Shift.

Note: Undo Script will only undo changes to forms whose windows are open or iconified when the Undo Script command is issued.

Note: Changes to picture properties can not be undone

Creating a Sample Script

1. Create a new form.
2. Add VBSCRIPT.VBX to the project.
3. In the VBScript window, type:

```
Create Label "NewLabel"
NewLabel.Caption = "Hello"
NewLabel.FontItalic = True
```

4. Select "Execute Script" from the File menu.   VBScript will create a label control named NewLabel and change the label's Caption property to Hello and change its FontItalic property to True.
5. Select "Undo Script" from the File menu.   The label will be deleted from the form.
6. Pull down the File menu.   The Undo Script command will have been replaced by a Redo Script command.   Select this command, and the label will be recreated and will have the modified Caption and FontItalic it had immediately after the script was run.

VBScript can be used to create more complex scripts which incorporate loops, if-then logic, and other standard programming language features.   A complete description of the VBScript language is included in this Help file.

The VBScript Language

Basic Usage

A VBScript script file is composed of variable declarations followed by a series of commands.

Multiple commands may be placed on the same line if a colon (**:**) is used as a separator (for example: `var = "Hello" : MsgBox var : MsgBox "yes"`)

VBScript commands, variable names, control names, and control properties are case insensitive - msgbox, Msgbox, and MsgBox are all valid commands.

To place a comment in a script, use an apostrophe (') .   All text between the apostrophe and the end of the line will be considered part of the comment.

Variables

VBScript supports two variable types: Variant and Control. Variant variables can hold either floating-point numbers or strings and can be used in numerical and string expressions. Control variables can represent controls on a form and can not be used in expressions.

Variables must be declared at the beginning of the script with the **Dim** statement. Variable names can contain digits, but they must begin with a letter.

Syntax: **Dim** *variablename*, [*variablename*, ...] [**As Variant** / **Control}**

```
Dim x As Variant
Dim ctrl1, ctrl2 As Control
```

If no variable type is specified in a **Dim** statement, the variables will be declared as Variants.

```
Dim variant1, variant2
```

Variable Usage

Variant variables can be assigned to any valid numerical or string expression.

```
x = "Foo"
var = 27 / 3 + 2
```

When a script is executed, Variants are initialized with a value of 0 (for numerical expressions) or `""` (for string expressions)

Control variables can be assigned to control identifiers or other control variables

```
ctrl1 = Label1
ctrl2 = Form2!Check1
ctrl1 = ctrl2
```

Controls in VBScript have two special properties, **Parent** and **Container**, which can be used in control identifiers. A control's **Parent** is the form on which the control is located. A control's **Container** is the container (frame, picture box, etc.) in which the control has been placed. If a control is not located inside a container, its **Container** property will be equal to its **Parent**.

```
ctrl1 = Label1.Parent
ctrl2 = Option2.Container
```

Expressions

VBScript supports numerical and string expressions. When appropriate, numerical expressions will automatically be converted to strings and vice versa. (for example: the expression `5 + "6"` is valid and is evaluated as 11)

Expressions can contain control property values and variables. Type conversions will also be performed automatically on these expression elements.

Numerical Expressions

Numerical expressions can contain floating-point numbers combined with several operators and functions.

Operators

+ (addition)	- (subtraction)	* (multiplication)
/ (division)	\ (integer division)	mod (modulo)
^ (exponentiation)		

VBScript follows Visual Basic's operator precedence rules and also supports the use of parentheses in expressions.

Functions

Int(*number*) removes the fractional part of *number* (if *number* is negative, **Int** returns the first integer less than or equal to *number*)

Fix(*number*) removes the fractional part of *number* (if *number* is negative, **Fix** returns the first integer greater than or equal to *number*)

Round(*number*) rounds *number* to the nearest integer

Abs(*number*) absolute value of *number*

Sgn(*number*) equal to -1 if *number* is less than 0, 0 if *number* is equal to 0, and 1 is *number* is greater than 0

RGB(*red,green,blue*) equal to the constant which represents the color with the specified red, green, and blue components

QBColor(*number*) obtains the constant which represents one of 16 predefined colors

Examples

`3 + 5 ^ 2`	is equal to 28
`(3 + 5) ^ 2`	is equal to 64
`int(4.6) + abs(-7.3)`	is equal to 11.3
`round(4.6)`	is equal to 5

String Expressions

String expressions can combine strings (enclosed in quotation marks), the & operator (for string concatenation), and several string-specific functions.. Strings can span multiple lines of a script and contain newlines.

Note: The + operator can not be used to concatenate strings

Functions

Len(*string*)	length of *string*
Mid(*string*,*start,length*)	extracts a portion of *string* starting at character number *start* that is *length* characters long

Left(*string,length*)	extracts the leftmost *length* characters of *string*
Right(*string,length*)	extracts the rightmost *length* characters of *string*
LTrim(*string*)	removes spaces from the left side of *string*
RTrim(*string*)	removes spaces from the right side of *string*
Trim(*string*)	removes spaces from the both sides of *string*
IsNumeric(*string*)	equal to **True** if the string can be converted to a number, otherwise **False**
Chr(*number*)	equal to the character whose ANSI code is *number*

Special Control Properties

A special control property, **Type**, is available in VBScript. A control's Type is a string which represents the control's object type. (for example, `Command1.Type` would be equal to `"CommandButton"`)

Examples

```
"Line 1                 "Line 1<newline>Line 2"
Line 2"
"Yes" & "No"            "YesNo"
Trim("    Hello    ") & Right("Small World",6)  "Hello World"
```

Comparison Operators

The standard =, <>, >, >=, <, and <= operators can be used to compare numerical and string expressions. String comparisons are case-sensitive by default - use the Option Compare statement to change the case-sensitivity setting.

A special **Is** operatior is used to test whether two controls identifiers refer to the same object. For example: if a label named MyLabel is on a form named Form1, then the expression

```
MyLabel.Parent Is Form1
```

is equal to **True**.

Logical Operators

The **And**, **Or**, and **Not** operators are supported in VBScript. Two special functions, **IsSelected** and **HasProperty**, may also be used in logical expressions. **IsSelected**(*control*) will be equal to **True** if *control* has been selected in the Visual Basic design environment and **False** if it has not been. **HasProperty**(*control*,*string*) will be equal to **True** if *control* has a property with the name *string*, otherwise **False**.

Examples

`5 > 3 and "less" < "more"`	is equal to **True**
`not (10 > -5)`	is equal to **False**

Braces

Braces let scripts work easily with collections of similarly named forms, controls, and variables. Expressions inside braces will automatically be converted to VBScript identifiers by the interpreter.

The identifier `Command{x}`, for example, can be used to refer to any control on the form whose name begins with Command. If the value of the variable `x` is set to 2, then `Command{x}` will mean `Command2`. If `x` is equal to "Button", then `Command{x}` will be equivalent to `CommandButton`.

Examples

```
      Dim i

      For i = 1 to 4
            Label{i}.Caption = "foo"
      Next
```

will set the Caption property of four labels named Label1, Label2, Label3, and Label4 to "foo".

Braces can also be used to form arrays from VBScript variables.

```
      Dim arr1, arr2, arr3, arr4, arr5, i

      For i = 1 to 5
            arr{i} = i
      Next
```

will set the values of the variables arr1, arr2, arr3, arr4, and arr5 to 1, 2, 3, 4, and 5, respectively.

Property Assignments

Property changes in VBScript work like their counterparts in Visual Basic.   The following examples will set some properties of a Label control.

```
Label1.Caption = "Hello"
Label1.FontSize = 8.25
Label1.FontItalic = True
```

The properties of control arrays, forms, and control variables can be accessed in the same manner.

```
LabelArray(3).Caption = "Hello"
Form1.Caption = "Program Name"
ctrlVar.Visible = True
```

Controls are ordinarily assumed to belong to the active form (the form which was most recently activated in the Visual Basic design environment).   This form can be referenced with the identifier **ActiveForm**. The active form can be changed with the SetFocus statement.

```
ActiveForm.Left = 2000
```

To reference controls on other forms, use the ! operator.

```
Form2!Label1.FontSize = 8.25
```

The **Forms** and **Controls** collections can be used as they are in Visual Basic.   Both collections have a Count property which is equal to the number of elements in the collection.

```
Forms(2).Caption = "Form Number 2"
var = Controls.Count
Forms(3)!Command1.Left = 200
Forms(0).Controls(6).Enabled = False
```

The **Forms** collection and the ! operator will only access forms whose windows are open or minimized when the script runs.   Any form which has not been opened via the Visual Basic Project window will be ignored by VBScript.

Picture Properties

Picture properties can only be assigned to other picture properties or to the predefined constant **None**. Picture properties can not be assigned to variables.

```
Image1.Picture = Picture1.Picture
Form1.Icon = None
```

Create

Syntax

Create *control-type name* [, *index*] [, *parent-control*]

The Create statement creates a new control of the type *control-type* and gives it the name *name*.

If *index* is specified, the control will become a member of the control array *name* and will have the index *index*.   To create a control which is not part of an array, specify an *index* of -1.

The control will be created as a child of *parent-control*.   If no *parent-control* is specified, the new control will be placed on the active form.

Examples

```
Create Label "Label5"
Create PictureBox "pic", 5
Create OptionButton "OptionArray", 2, Frame1
Create SSCommand "cmd", -1, Form2
```

Delete

Syntax

Delete *control*

The Delete statement deletes a particular control.

Examples

```
Delete Command1
Delete Form2!Label1
```

Move

Syntax

Move *control*, *left* [, *top*] [, *width*] [, *height*]

The Move statement moves a control to the specified *left*, top, width, and *height* coordinates.   It is not necessary to include all four coordinates in the statement; any coordinates which are not included will not be changed.

Examples

```
Move Command1, 1000, 2000, 1500, 500
Move Label1, 750
Move Form1, 3000, 1000
```

SetContainer

Syntax

SetContainer *control*, *parent-control*

The SetContainer statement places *control* within the container *parent-control*.   *parent-control* must be a container control (form, frame, picture box, etc.) and can be located on any form.

Examples

```
SetContainer Option1, Frame1
SetContainer CmdBtn, Form2!Panel3D1
SetContainer Picture1, MDIForm1
```

SetFocus

Syntax

SetFocus *form*

The SetFocus statement sets the active form to *form*.   The active form is the form to which controls are assumed to belong when the `!` operator is not used.

Example

```
SetFocus Form1
```

If...Then...Else

Syntax 1

If *expression* **Then** <commands> **Else** <commands>

Syntax 2

If *expression* **Then**
    <command block>
[**ElseIf** *expression* **Then**
    <command block>]
[**Else**
    <command block>]
End If   (or **EndIf**)

The If...Then...Else statement executes commands conditionally.   The condition expressions can be logical (**True**/**False**) expressions or **TypeOf...Is** expressions, which are true if a certain control is of a specified type.

Examples

```
If x >= y Then Beep Else MsgBox "x is less than y": Beep

If TypeOf Controls(0) Is Label Then
      MsgBox "Control #0 is a Label"
      Beep
ElseIf TypeOf Controls(1) Is Label Then
      MsgBox "Control #1 is a Label" : Beep
Else MsgBox "Neither control is a label"
End If
```

While...Wend

Syntax

While *expression*

<command block>

Wend

Executes the command block as long as *expression* is true.

Example

```
While x < 5
      MsgBox x
      x = x+1
Wend
' X will now be equal to 5
```

For...Next

Syntax

For *counter* = *startExpression* **To** *endExpression* [**Step** *stepExpression*]
<command block>
Next

At the start of the **For** loop, the Variant variable *counter* is given the value of *startExpression*. Each time the command block is executed, *stepExpression* is added to *counter* (1 is added if no **Step** expression is given). When *counter* is greater than or equal to *endExpression*, the loop exits.

You can exit a **For** loop immediately by using the **Exit For** command at any point in the command block.

Note: Do not specify a counter variable in the **Next** statement

---

Example

```
For z = 0 To 7
      ButtonArray(z).Enabled = False
      ButtonArray(z).Caption = z
Next
```

Do...Loop

Syntax 1

Do {**While** / **Until**} *expression*
    <command block>
Loop

Syntax 2

Do
    <command block>
Loop {**While** / **Until**} *expression*

Executes the command block while (or until) *expression* is true. Syntax 1 checks the expression's value upon entering the command block, and Syntax 2 checks upon exiting the block.

You can exit a **Do** loop immediately by using the **Exit Do** command at any point in the command block.

Examples

```
Do Until index = 5
      Forms(index).Left = index * 200
      index = index + 2
Loop

Do
      Forms(index).Left = index * 200
      index = index - 2
Loop While index > 2
```

Select Case

Syntax

Select Case *caseExpression*
    Case *expression*, [*expression*...]
        <command block>
    Case *expression*, [*expression*...]
        <command block>
    [**Case Else**
        <command block>]
End Case

If *caseExpression* is equal to one of the expressions in a expression list, the command block which corresponds to that expression list will be executed. If caseExpression is not equal to any of the expressions, the **Case Else** command block (if one is present) will be executed.

Case expression lists can contain numerical or string expressions. The expression list parts can be of any of these three formats.

expression	tests whether *caseExpression* is equal to *expression*
expression1 **To** *expression2*	tests whether *caseExpression* is between *expression1* and *expression2*
Is *comparison-operator expression*	tests whether *caseExpression* has the relationship indicated by *comparison-operator* with *expression*

Examples

```
Select Case x
      Case 5
            MsgBox "x is equal to 5"
      Case 6 to 9
            MsgBox "x is between 6 and 9"
      Case Is > 9
            MsgBox "x is greater than 9"
End Select
```

MsgBox

Syntax

Statement: **MsgBox** *message* [, *type*] [, *title*]
Function: **MsgBox** (*message* [, *type*] [, *title*])

Displays a message and one or more buttons in a dialog box. *Message* will be displayed in the dialog box, *type* specifies the style of the dialog, and *title* will be displayed as the dialog's caption.

If *type* is omitted, the dialog will only contain an OK button. If *title* is omitted, "VBScript" will be used as the title.

MsgBox Types

Type Number	Buttons in dialog
0	OK button
1	OK, Cancel
2	Abort, Retry, Ignore
3	Yes, No, Cancel
4	Yes, No
5	Retry, Cancel

If MsgBox is used as a function, its return value will indicate the button which was pressed by the user.

MsgBox Return Values

Return Value	Button selected
1	OK
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

Examples

```
MsgBox "Hello"
x = MsgBox ("Do you want to continue", 5)
MsgBox "An error was found", , "Error"
```

InputBox

Syntax

InputBox (*prompt* [, *title*] [, *default*])

The **InputBox** function will display a dialog box that allows the user to enter a string into a text box. The function will return the text box's contents when the user clicks OK. *Prompt* will be displayed as the message in the dialog box, *title* will be the dialog's caption, and *default* will be the text box's default contents.

If *title* is omitted, "VBScript" will be used as the title. If *default* is omitted, the text box will be initially blank.

Examples

```
var = InputBox ("Enter a string")
num = InputBox("Type in a number", "Number",  "5")
```

Beep

Syntax

Beep

Beeps the computer's speaker.

Exit

Syntax

Exit {**Script** / **Do** / **For**}

Exit Script will immediately stop script execution. The **Exit Do** and **Exit For** commands can be used in **Do** and **For** loops to stop looping and jump to the statement immediately following the loop's end.

Option

Syntax

Option Compare {**Binary** / **Text**}
Option MenusInCollection {**On** / **Off**}

The **Option Compare** statement sets the case-sensitivity mode for string comparisons. **Option Compare Binary** will select case-sensitivity, and **Option Compare Text** will select case-insensitivity. The default setting is **Binary**.

The **Option MenusInCollection** statement determines whether Menu controls will be included in the **Controls** collection. **Option MenusInCollection On** will include Menu controls in the collection, and **Option MenusInCollection Off** will exclude them. The default setting is **Off**.

Example Scripts

These scripts are intended as examples of VBScript's syntax and illustrate how to perform common tasks in VBScript.

This script will align all the buttons in the array BtnArray with the top of another button named Command1.

```
Dim i As Variant

While i < Controls.Count
     If Controls(i).Name = "BtnArray" Then
            Controls(i).Top = Command1.Top
     End If
     i = i + 1
Wend
```

This script will prompt the user to enter a number.   It will then change the caption of a label named Label3 to the square root of that number.

```
Dim num As Variant

num = InputBox("Enter a number")
If Not IsNumeric(num) Then
     Beep
     MsgBox num & " is not a number"
     Exit Script
End If
If num < 0 then
     MsgBox "Cannot find square root of a negative number"
Else Label3.Caption = num ^ .5
End If
```

This script will create a design made up of line controls on the active form.

```
Dim pos

For pos = 0 To 10
     Create Line "LineArray1", pos
     LineArray1(pos).x1 = 0
     LineArray1(pos).y1 = Form1.ScaleHeight * pos/10
     LineArray1(pos).x2 = Form1.ScaleWidth * (10-pos)/10
     LineArray1(pos).y2 = 0
Next
```

This script will create labels which illustrate the colors that can be obtained from the QBColor function.

```
Dim i
```

```
For i = 0 To 15
     Create Label "QBDemo", i
     QBDemo(i).BackColor = QBColor(i)
     QBDemo(i).Caption = i
     QBDemo(i).FontSize = 20
     QBDemo(i).AutoSize = True
     Move QBDemo(i), 500, i*500
Next
```

This script will delete the controls Check1, Check2, Check3, and Check4.

```
Dim x

For x = 1 to 4
     Delete Check{x}
Next
```

This script will set the FontBold property to False for all option buttons on the current form which have been selected in the Visual Basic interface designer.

```
Dim index As Variant

Do
     If IsSelected(Controls(index)) Then
            If TypeOf Controls(index) Is OptionButton Then
Controls(index).FontBold = False
     EndIf
     index = index + 1
Loop Until index = Controls.Count
```

This script will display a message box indicating the form name, control name, and array index (if applicable) of every control on the currently active forms.

```
Dim frm, ctrl As Variant

For frm = 0 to Forms.Count - 1
     For ctrl = 0 to Forms(frm).Controls.Count - 1
            If Forms(frm).Controls(ctrl).Index = -1 Then
                  MsgBox Forms(frm).Controls(ctrl).Name , , Forms(frm).Name
            Else
                  ' this control is part of a control array
                  MsgBox Forms(frm).Controls(ctrl).Name & "(" &
Forms(frm).Controls(ctrl).Index & ")" , , Forms(frm).name
            End If
     Next
Next
```

License

VBScript is shareware. You may evaluate this software for 30 days. If you continue to use the software after that period, you must register it.

Disclaimer

This software is provided "as-is" and comes with no warranty of any kind. Use this product at your own risk. In no event will the author be liable for any damages arising as a consequence of the use of this software.

Visual Basic is a trademark of Microsoft Corporation

Copyright © 1995 by Jason Simmons

Registration

To register VBScript, print this form and send it along with a check or money order in US funds to:

Jason Simmons
82 Stonehurst Lane
Dix Hills, NY 11746

Registered users will receive a copy of the latest version of VBScript and are entitled to unlimited free support by mail and e-mail.

If you have any comments, suggestions, questions, or bug reports, please let me know. I can also be reached via Internet at jsimmons@cs.sunysb.edu

Price: $20.00 per copy # of copies ____
Add $2 shipping for disk orders outside the United States

Ship to ____

____

____

____

Phone # ____

E-mail ____

Disk size: __ 5¼ __ 3½ __ Send the registered version by e-mail
(check one)

How did you obtain this copy of VBScript?

____