VBScript Language

What is VBScript

- Based on the Visual Basic family of languages
- Supports object oriented features
- Interpreted
- Loosely Typed
- Implicitly Declared

Contents

- Syntax Rules
- Comments
- Naming Rules
- Data Types
- Storage
  - Variables
  - Constants
- Operators
  - Arithmetic
  - Comparison
  - Logical
  - Assignment
  - Concatenation
- Branching
  - If
  - Select
- Looping
  - While
  - Do…While
  - For
- Functions
  - Built-In
  - User defined

Syntax Rules

- VBScript is not case sensitive (except for Strings)
- Each statement can only exist on one line
- Statements can be either simple statements or compound statements

Comments

Rem commented statement

‘ commented statement

- Can be a complete line or a portion of a line.
- VBScript does not support multi-line comments. The comment symbol must be put at the start of each comment.

Naming Rules

1. Must begin with a letter or the underscore
   - Good: count, strInfo, _data
   - Bad: 2day, 4get, *count violate this rule.
2. Cannot contain special characters except the underscore
   - Good: Names using alphabets, numbers and the underscore.
   - Bad: Names using symbols and other special characters.
3. Cannot match a reserved word
   - Bad: for, if, Select
4. Must be unique within a context
5. Must be less than 256 characters

Naming Convention

objDescriptiveName

- obj
  - Three letter prefix for the data type
- DescriptiveName
  - A descriptive name for the variable
  - First letter of each word in upper case

Literal Types

- Numeric
  - 1
  - 3.14
  - 3E7
- Boolean
  - True
  - False
- Characters
  - “Hello World”
  - “123 Straw Lane”
  - “43210”
- Date
  - #07/21/2006#
  - #Jan 6, 2008#

Variables

Dim *variableName*

- Variables hold values that can change during program execution
- They function just like temporary storage locations
- Declaration of variables is optional, but recommended

Constants

Const constant*Name* **=** *value*

- Constant names hold values that **cannot** change during program execution
- Constants should be used in place of hard-coded values

Operations

- Operations are the many forms of computations, comparisons etc that can be performed in VB
- Most operations are binary in the form:

  operand1 operator operand2
- Other operations are unary in the form

  operator operand1
- When an operator has 2 symbols, you cannot separate them with a space:
  - Good: 2 <> 3
  - Bad: 2 < > 3

Arithmetic

- Used to perform calculations
- Both operands must be numeric values
- The result is a numeric value

+	Addition
-	Subtraction, Negation
*	Multiplication
/	Division
\	Integer Division
^	Exponent
Mod	Modulo

Comparison

- Used to compare the value of two items
- Both operands must be of the same data type
- The result is a Boolean value

=	Equality
<>	Inequality
<	Less than
>	Greater than
<=	Less than or equals
>=	Greater than or equals

Logical

- Used to reduce Boolean values into a single Boolean value
- Both operands must be Boolean values
- The result is a Boolean value

And	Conjunction
Or	Disjunction
Not	Negation
Xor	Exclusion

And Truth Table

- Conjunction
- Used when both conditions are required

Operand1	Operand2	Result
T	T	T
T	F	F
F	T	F
F	F	F

Or Truth Table

- Disjunction
- Used when either condition is sufficient

Operand1	Operand2	Result
T	T	T
T	F	T
F	T	T
F	F	F

Not Truth Table

- Unary Operand
- Used to change the value of a condition

Operand 1	Result
T	F
F	T

Xor Truth Table

- Exclusion
- Used when either condition should be different

Operand1	Operand2	Result
T	T	F
T	F	T
F	T	T
F	F	F

Assignment

- Changes the value of a variable
- =
- Uses the same symbol as an equality check operator.
- Assignment only occurs when used in any of the following forms:

variable = literal value

variable = expression

variable = variable

Concatenation

- Combines 2 data types for display as a string
- &

Branching

- Allows you to avoid running certain sections of your code
- Code is only executed when a condition (or conditions) evaluate to True
- Provides a single application the ability to react differently to different input values

If

If *condition* **Then**
statement(s)
End If

- Performs an operation only if the condition evaluates to True
- Used when an action is either performed or not performed.

If...Else

If *condition* **Then**
  statement(s)
Else
  statement(s)
End If

- Performs the If portion only if the condition evaluates to True and the Else portion otherwise.
- Used in an either or scenario when an operation is always performed.

If...ElseIf

If *condition1* **Then**
  statement(s)
ElseIf *condition2* **Then**
  statement(s)
Else
  statement(s)
End If

- Only one section can be executed even if many conditions evaluate to True.
- Used in a multiple choice scenario
- Inclusion of the last **Else** is optional

Select

Select Case *variable*
  statement(s)
Case *value1*
  statement(s)
Case *value2*
  statement(s)
Case Else
  statement(s)
End Select

- Used to choose between 1 of several discrete values for a variable.
- Inclusion of **Case Else** is optional
- Less powerful compared to the If statement

Loops

- Allows you to repeat running certain sections of your code
- Code executes when a condition (or conditions) evaluate to True
- Be careful with Loops. They can result in infinite processing.
- Forms
  - Entry Condition
    - Entry only when a initial condition is met
  - Iterated
    - Repeats for a specific number of times

Loops: Cautionary Items

1. Can this loop ever be entered
   - If no, then you don't need the loop
2. Can this loop ever be exited
   - If no, then you have an infinite loop

Loop: Parts

1. Where does the loop start?
    - Expressed as an assignment
2. When does the loop end?
    - Expressed as a condition
3. How is the loop variable altered?
    - Expressed as an arithmetic operation
    - Ensure that you are increasing/decreasing properly

While

While *condition*
statement(s)
Wend

- Entry Condition Loop
- Simplest form of the loop
- Requires manual modification of the loop condition

For

For *variable* **=** *start* **To** *finish* [**Step** *change*]
  statement(s)
Next

- Iterated Loop
- Favored because all the loop details are in the definition statement

Function

Function *functionName(parameter list)*
statement(s)
functionName = value
End Function

- Block of code used to perform an operation
- Both VBScript and QTP provide a large number of built-in functions
- You can add your own built-in functions to the list

Best Practices

- Comment liberally
- Use a naming convention
- Avoid mixing cases
- Indent your code