

CSET 3250

Client-Side Scripting

VBScript : Operators, Control Structures,
Procedures and Functions

Operators

- VBScript has a full range of operators, including arithmetic operators, comparison operators, concatenation operators, and logical operators
- When several operations occur in an expression, each part is evaluated and resolved in a predetermined order. This is known as operator precedence
- Operations within parentheses are always performed before those outside

Operators

Arithmetic		Comparison		Logical	
Description	Symbol	Description	Symbol	Description	Symbol
Exponentiation	^	Equality	=	Logical negation	Not
Unary negation	-	Inequality	<>	Logical conjunction	And
Multiplication	*	Less than	<	Logical disjunction	Or
Division	/	Greater than	>	Logical exclusion	Xor
Integer division	\	Less than or equal to	<=	Logical equivalence	Eqv
Modulus arithmetic	Mod	Greater than or equal to	>=	Logical implication	Imp
Addition	+	Object equivalence	Is		
Subtraction	-				
String concatenation	&				

Conditional Loops

- The **If...Then...Else** statement is used to evaluate whether a condition is **True** or **False** and then to specify one or more statements to run, depending on the result
- If you want to run only one statement when a condition is **True**, you can use the single-line syntax of the **If...Then...Else** statement

```
Sub FixDate()  
Dim myDate  
myDate = #5/4/01#  
if myDate < Now Then myDate = Now  
End Sub
```

Conditional Loops

- If you want to run more than one line of code, you must use the multiple-line syntax. This syntax includes the **End If** statement, as shown in the following example:

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

Conditional Loops

- You can use an **If...Then...Else** statement to define two blocks of executable statements: one block to run if the condition is **True**, the other block to run if the condition is **False**

```
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
```

Conditional Loops

- Looping allows you to run a group of statements repeatedly.
- Some loops repeat statements until a condition is **False**; others repeat statements until a condition is **True**. There are also loops that repeat statements a specific number of times. The following looping statements are available in VBScript:
 - **Do...Loop**: Loops while or until a condition is **True**
 - **While...Wend**: Loops while a condition is **True**
 - **For...Next**: Uses a counter to run statements a specified number of times

Do..Loops

- The **While** keyword is used to check a condition in a **Do...Loop** statement.
- You can check the condition before you enter the loop or you can check it after the loop has run at least once.

```
Sub ChkFirstWhile()  
    Dim counter, myNum  
    counter = 0  
    myNum = 20  
    Do While myNum > 10  
        myNum = myNum - 1  
        counter = counter + 1  
    Loop  
    MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

```
Sub ChkLastWhile()  
    Dim counter, myNum  
    counter = 0 myNum = 9  
    Do myNum = myNum - 1  
        counter = counter + 1  
    Loop While myNum > 10  
    MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

Do..Loops

- **Repeating a Statement Until a Condition Becomes True**
- You can use the **Until** keyword to check a condition in a **Do...Loop** statement before you enter the loop or you can check it after the loop has run at least once.

```
Sub ChkFirstUntil()  
Dim counter,  
myNum counter = 0  
myNum = 20  
Do Until myNum = 10  
    myNum = myNum - 1  
    counter = counter + 1  
Loop MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

```
Sub ChkLastUntil()  
Dim counter,  
myNum counter = 0  
myNum = 1  
Do myNum = myNum + 1  
    counter = counter + 1  
Loop Until myNum = 10  
MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

Do..Loops

- You can exit a Do...Loop by using the Exit Do statement. Because you usually want to exit only in certain situations, such as to avoid an endless loop, you should use the Exit Do statement in the True statement block of an If...Then...Else statement

```
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 "The loop made " & counter & " repetitions."  
    End Sub
```

For...Next

- You can use For...Next statements to run a block of statements a specific number of times
- For loops, use a counter variable whose value is increased or decreased with each repetition of the loop
- For example, the following procedure causes a procedure called MyProc to execute 50 times
- The For statement specifies the counter variable x and its start and end values
- The Next statement increments the counter variable by 1

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

- Using the Step keyword, you can increase or decrease the counter variable by the value you specify
- To decrease the counter variable, you use a negative Step value

Procedures

- In VBScript there are two kinds of procedures; the **Sub** procedure and the **Function** procedure
- A procedure is defined by a Sub statement and ends with an End Sub statement
- A procedure is defined using the following structure:
 Sub ProcedureName(ArgumentList)
 VbScript Code
 End Sub

Example:

```
Sub DisplayName(strFirstName, strLastName)
    alert("Welcome " & strFirstName & " " & strLastName)
End Sub
```

Procedures

- In order to call a procedure, it is sufficient to simply use the name of the procedure:

```
DisplayName "Arman", "Danesh"
```

- Procedures can also be called using the Call statement:

```
Call DisplayName("Arman", "Danesh")
```

Functions

- A **Function** procedure is a series of VBScript statements enclosed by the **Function** and **End Function** statements
- A **Function** procedure is similar to a **Sub** procedure, but can also return a value
- A **Function** procedure can take arguments (constants, variables, or expressions that are passed to it by a calling procedure)
- If a **Function** procedure has no arguments, its **Function** statement must include an empty set of parentheses
- A **Function** returns a value by assigning a value to its name in one or more statements of the procedure
- The return type of a **Function** is always a **Variant**

Example Function

```
Sub ConvertTemp()  
    temp = InputBox("Please enter the  
    temperature in degrees F.", 1)  
    MsgBox "The temperature is " &  
    Celsius(temp) & " degrees C."  
End Sub  
  
Function Celsius(fDegrees)  
    Celsius = (fDegrees - 32) * 5 / 9  
End Function
```

Functions(contd)

- To use a **Function** in your code, it must always be used on the right side of a variable assignment or in an expression. For example:

```
Temp = Celsius(fDegrees)
```

```
or MsgBox "The Celsius temperature  
is " & Celsius(fDegrees) & "  
degrees."
```

Assignment

- Download the examples for this topic
 - Unzip them on your local machine
 - Load each into your browser in turn
 - After determining what each does, look at the source code to see how it is done
 - Modify any of the examples as needed to make them fully functional (correct any errors)
- FTP the examples to the server and run them from the server