**VB script Control Flow statements**

Sometimes in our programs we want to execute some code depending upon the result of some condition.   
  
VBScript If statement is used to make decisions in our code depending upon the result of some condition.   
For example we want to greet the visitors of our website with appropriate messages; good morning, if it is morning, good evening if it is evening, etc.

Here is the syntax of the If statement:

If [condition] Then  
' VBScript code to execute if the condition is true   
End If

We write the keyword If followed by a condition. A condition is any VBScript expression that evaluates to true or false. A condition is usually written by using one of the several comparison operators available in VBScript. After the condition we write the keyword Then and then we write the code that we wish to be executed in case the condition turns out to be true. Finally we write the words **End If** to end the If code block.

Dim a  
Dim b  
a = 10  
b = 20  
If a < b Then

Msgbox “a is less than b”

End if

Sometimes we have two pieces of code and we wish to execute one if the condition is true and the other if the condition is false.   
  
In such situations we use the **If Else** statement. The If Else is an extension to the If statement that allows us to execute a piece of code if the condition is true, in case the condition is false a different block of code gets executed.   
  
Here is an example:

Dim a  
Dim b  
a = 30  
b = 20  
If a < b Then   
Msgbox "A is less than B”  
Else  
Msgbox "A is greater than B"  
End If

There is another variation of the If statement called Else if. We use Else if when we need to check multiple conditions.   
  
Here is an example:

Dim a  
Dim b  
a = 30  
b = 20  
If a < b Then   
Msgbox A is less than B"  
Else if a > b Then  
Msgbox “A is greater than B"  
Else  
Msgbox "A and B are equal"  
End If

**Select case**

There is another construct available in VBScript called Select Case, that is a more elegant way of doing the same thing.   
A Select Case statement allows us to perform multiple tests, with fewer lines of code.   
  
Here is an example:

Dim favFruit  
favFruit = "Mango"  
Select Case favFruit  
Case "Orange"  
Msgbox "My favorite fruit is Orange"  
Case "Mango"  
Msgbox "My favorite fruit is Mango"  
Case "Apple"  
Msgbox "My favorite fruit is Apple"  
Case Else  
Msgbox "My favorite fruit is not available!"  
End Select

**Important points about the Select Case Statement**  
The Select Case statement only allows us to check if a variable is equal to a value. We cannot make greater and / or less than type checks.   
We can only test against one variable. For testing multiple variables we need to use the If statement.

**While Loop**

Sometimes in our programs we have some code that we need to repeat for a specific number of times.   
  
VBScript offers us a number of looping constructs that we can use to repeat code in our programs. In this lesson we will see the use of While Loop.   
  
The While Loop is used to repeat a block of code while a given condition is true.

Here is an example:

Dim counter  
counter = 0  
While counter <= 10  
Msgbox "Loop Iteration No. " & counter   
counter = counter + 1  
Wend

Here is how the above code works:

* We start by declaring a variable named counter and then setting it to 0.
* Next we use the **While** keyword to start the While loop followed by the condition i.e. counter <= 10. The condition asks the VBScript interpreter to repeat the code contained in the While loop until the value of the variable named counter is less than or equal to 10.
* The next two lines are the actual code that gets executed on each iteration of the loop. We call every loop repetition an iteration of the loop. The first line outputs the loop iteration number and the second line increments the counter variable by one.
* At the end of each loop iteration the VbScript interpreter checks to see if the loop condition is still true, if it find it to be true the loop code block is executed otherwise the control transfers to the line immediately after the Wend keyword.

While loop keeps repeating itself until a given condition is true.

**FOR LOOP**

Another looping construct available in VBScript is the For Loop. Unlike the While loop, the For Loop does not take a condition, instead we provide a numeric variable to it and provide a minimum value to start the loop and a maximum value till we require the loop to keep iterating.

For counter = 1 to 10  
Msgbox Loop “ Iteration No. " & counter   
Next

Here is how the above code works:

* We start the loop by using the For keyword.
* Next we provide a variable name followed by the equal (=) sign, the start value and the end value for the loop. In the above code we provide 1 as the start value for the loop and 10 as the end value for the loop, so that the loop executes 10 times.
* Next we output some text and finally end our loop with the Next keyword.
* Note that the variable name we provide is used, internally, by the VBScript interpreter to control the count of the loop.
* After every iteration the loop control variable is increments by one, automatically.

**For Each Loop**The VBScript For Each Loop allows us to iterate over an array. This is a quick and easy way of accessing all elements within an array without needing to know how many elements are in it.

Here is an example:

Dim fruits(3)  
fruits(0) = "Orange"  
fruits(1) = "Apple"  
fruits(2) = "Mango"  
For Each fruit In fruits  
Msgbox (fruit)  
Next

Here is how the above code works:

* We define an array named fruits having 3 items and assign values to its every index.
* We begin the loop with the keywords For Each followed by the name of a variable followed by the keyword In and the name of the array.
* In the For Each loop body, on every iteration, we have a new item contained in the variable fruit, which we access and output to the browser window.
* The keyword Next marks the end of the For Each loop block.