Tutorial Básico de vbscript

Bueno, primero aclarar que este tutorial, pretende explicar de manera **básica** las distintas instrucciones aplicadas en visual basic script (vbs de aquí en más), para que así de este modo, todos aquellos interesados en este lenguaje de scripting, tengan la posibilidad de comenzar a desarrollar sus pequeños scripts.

Introducción:

Para empezar, debemos decir, que los vbs, tal y como lo dice su nombre, son "simples" scripts, que se ejecutan por medio de un intérprete en el sistema, con lo que bastará con crear el código en un simple bloc de notas (o similar), y guardarlo con extensión *.**vbs,** sin necesidad de realizar ninguna compilación (aunque existe la posibilidad).

En este caso el intérprete es, el **Windows Script Host** de Microsoft, por lo que estará presente (aunque en distintas versiones) de manera predeterminada desde Win 98 en adelante, así que podremos hacer uso de estos archivos en prácticamente, todos los Windows.

Se puede decir que los vbs son una mejora con respecto a los bat, ya que estos, permiten una mayor interacción con el sistema operativo, y decir además que este lenguaje, es un derivado de **Visual Basic,** por lo que desde un principio y aunque de manera mucho más limitada, se podrá familiarizar con dicho leguaje (la sintaxis es similar en todos los aspectos).

A tener en cuenta antes de comenzar:

- En vbs no importan las mayúsculas o minúsculas

  VBscript = vbscript

- Para indicar un final de línea, no se debe de hacer más que pasar a la siguiente (no es necesario terminar con por ejemplo " **;** " como en javascript)

- Los comentarios, van precedidos de una " **'** " (comilla simple)

  ' Esto es un comentario

- No es obligatorio declarar las variables, aunque es conveniente hacerlo para dejar más legible el código

  Se pueden declarar utilizando **Dim,** o bien, se les da valor directamente.

  Dim nuestravariable

  También puede ser utilizado **Option explicit** para así obligarse a declarar las variables (en los pequeños ejemplos lo haré así)

Nota: En la mayoría de los ejemplos utilizaré la función **msgbox**, la cual se encarga de mostrar un mensaje en pantalla.

En el caso de copiar y pegar uno de los ejemplos al notepad, revisar las comillas, ya que al copiar desde aquí a allí, no son las mismas, y puede que de errores.

Tipos de dato:

Existen varios tipos de datos que pueden manejarse en vbs, y los que más vamos a utilizar, son ;

Booleano, Byte, Fecha, Double, Entero, Entero largo, Objeto, Single, Cadena.
Todos estos representan valores verdadero/falso, fecha/hora, númerosentero positivo/negativo, cadena de texto (no creo necesite más explicación que esta).

Operadores:

Tendremos varios tipos de operadores a nuestra disposición.

1. Aritméticos: Suma(+), Resta(-), Multiplicación(*), División decimal(/), División entero(\), Potencia(^), Resto división(mod)
2. Comparación: Igual(=), Distinto(<>), Mayor(>), Menor(<), Menos o igual(<=), Mayor o igual(>=)
3. Lógicos: Y(and), O(or), Xor, No(not)
4. De cadena: Concatenación(&)

Estos han sido a grandes rasgos, los operadores a utilizar en vbs, por lo que después, y haciendo uso de nuestro próximo tema (estructuras de control), se mostrarán algunos ejemplos.

Estructuras de control:

1. **IF (condicional)**

   Esta se utiliza para evaluar 2 o más posibles resultados, en virtud del cual, se tomarán diferentes acciones.

```
Dim valor
Valor = 8
If  valor < 10 then
    Msgbox "El valor es MENOR a diez "
Else
    Msgbox "El valor es MAYOR a diez "
End if
```

   Como se ve en este ejemplo se comienza declarando la variable **“valor”,** luego de esto, se le asigna el valor **8,** y posterior a esto, se evalúa dicho valor (haciendo uso de los operadores de comparación vistos anteriormente), por lo que si el valor es menor a **10,** se mostrará un mensaje en pantalla indicando que es MENOR, de lo contrario, el mensaje indicará MAYOR (en este caso le habíamos asignado el valor 8, por lo que será menor).

2. Case (condicional):

En el caso del IF, si bien no se ha mostrado, existe la posibilidad de anidar dos o más, para así realizar varias evaluaciones, pero el caso es que cuando se realizan muchas, el código puede volverse algo engorroso. Es en estos casos, donde recurrimos al **select case.**

```
Dim numero
numero = 8
Select case numero
    case 6
        msgbox "El valor es seis "
    case 7
        msgbox "El valor es siete "
    case 8
        msgbox "El valor es ocho "
End select
```

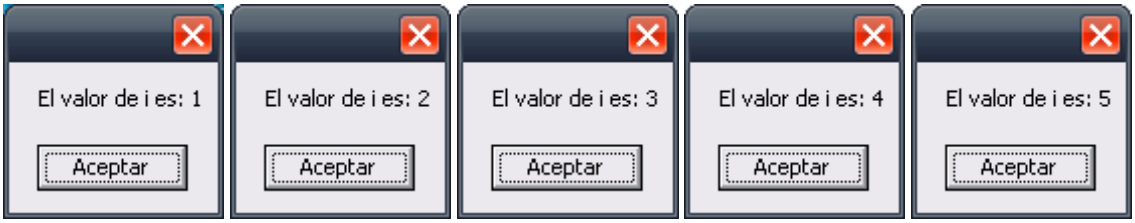
Al igual que en el anterior, en este ejemplo, se comienza definiendo la variable “numero”, y se le asigna **8** como valor. Luego con **select case** se indica que es lo que se va a revisar, en este caso, cuanto es el valor de “numero”, luego, se indican las opciones con un **case** por cada una de ellas, en este caso, “case 6”, en el caso de que el valor sea 6, “case 7”, y “case 8”, para el caso de que el valor sea 7 y 8 respectivamente (como sabemos es 8), es decir, “si es 6”, mensaje “El valor es seis”, “si es 7”, mensaje “El valor es siete”, y “si es 8”, mensaje “El valor es ocho”.

3. FOR (bucle)

El for es utilizado cuando queremos repetir una determinada acción un cierto número de veces.

```
Dim i
For i=1 to 5 step 1
    Msgbox "El valor de i es: " & i
Next
```

Comenzamos definiendo la variable “i”, luego de esto, inicializamos el bucle for, es decir, decimos que; desde “i” igual **1**, a “i” igual **5**, incrementando de a **1,** mostraremos un mensaje con la frase “El valor de i es: “ y luego de esto, el valor real de “i”, por lo que en cada “paso” que de nuestro bucle, mostrará algo más o menos así…



Como ven, 5 mensajes “casi” iguales, ya que la secuencia se repite 5 veces, y lo único que cambia es el valor de “i” (la hemos concatenado al mensaje con el carácter &), ya que va tomando distintos valores en cada uno de los “pasos”.

4. FOR EACH(bucle)

El for each, tiene la particularidad de que es un for que recorre todos los elementos de una colección o vector. Antes de continuar, intentaré explicar lo que es un “array”, ya que será lo que utilicemos en el ejemplo.

Un **array o matriz,** es una estructura de datos en forma de variable, que permite almacenar más de un único valor, dentro de una única variable. Para acceder a cada uno de estos valores, será necesario hacer uso de índices.

```
Dim nombres(2)

nombres(0)= "Martín"

nombres(1)= "Pablo "

nombres(2)= "Carlos "

Msgbox nombres(1)
```

En este ejemplo, se define la variable “nombres” que tendrá espacio para 3 valores (tener en cuenta que comienza por 0, por eso que se ponga 2), y luego, se definen cada uno de los valores de “nombres”. Por último utilizamos un mensaje, para mostrar el valor del índice 1 del array, con el siguiente resultado…

Como se ve, el mensaje muestra el nombre “Pablo” porque al llamar al array, hemos puesto el índice **1,** y como se ve cuando lo declaramos, le habíamos asignado valor “Pablo”, en el caso de indicar 2 en lugar del 1, el mensaje mostraría “Carlos”.

Espero esto haya quedado lo suficientemente claro.

Ahora, volviendo al For each (espero ya no te hayas olvidado), este podría ser utilizado en el caso del array (hay otros casos).

```
Dim nombres

Dim n

nombres = array("Martín","Pablo","Carlos")

For each n in nombres

    Msgbox n

next
```

A pesar de que no lo parezca, es muy similar al anterior, y algunos estarán diciendo, "que tenía que ver el array con esto", pero si se fijan, solamente he mostrado, una segunda manera de declarar el array.

En este caso, en lugar de ir metiendo cada uno de los valores con sus respectivos índices, declaro el array en una sola línea y va quedando organizado según el orden en el que meta los valores, por eso, "Martín" quedará en el lugar 0, "Pablo" en el lugar 1, y "Carlos", en el lugar 2, al **igual** que en el caso anterior. Esto con respecto al array, ahora a lo que íbamos, el for each.

En líneas siguientes comenzamos con el for, y decimos, **por cada** "n" (la letra, en este caso "n" la elegimos nosotros) **en "**nombres" (es un array que contiene tres nombres), mensaje con el "n", o lo que es igual, por cada nombre, en el "contenedor" **"nombres"**, mensaje con el nombre, y el resultado es el siguiente…

Tanto en **for** simple, como en el **for each**, podemos hacer uso de **exit for**, para parar la ejecución del for completa. Para decirlo de otra manera, con esto escapamos al for.

WHILE (bucle)

El bucle while, es utilizado cuando queremos que una determinada acción se repita **mientras** una determinada condición de cumpla.

```
Dim numero

Numero = 1

While numero <= 10

    Msgbox numero

    numero = numero + 1

  Wend
```

Comenzamos el ejemplo declarando la variable numero, y le asignamos valor 10, luego ejecutamos bucle, **mientras** el valor de "numero" sea menor o igual a 10, mostramos mensaje con el valor de numero, y además le sumamos 1 al propio valor de numero (de este modo lo utilizamos a modo de contador), luego de 10 veces de mostrar mensaje, el valor de "numero" será igual a **11** y al verificarse la condición esta no se cumplirá, por lo que terminará.

5. DO (bucle)

El bucle do tiene varias opciones, pero básicamente **hará** algo, "mientras", o "hasta que" ocurra algo.

```
Do

Msgbox "Esto es un mensaje"

Loop
```

Este bucle, se repetirá al infinito, por lo que mostrará un mensaje, interminables veces.

Ahora bien, este do, puede ser modificado, para tener algo más de control sobre él, y es justamente haciendo uso de esas condiciones que comentaba antes, "mientras", o "hasta que"

```
Dim a = 1

Do until a=10

Msgbox "Esto es un mensaje"

Loop
```

Este código se resume en; mensaje de "Esto es un mensaje" hasta que "a" sea igual a 10.

En lugar de **until** podría utilizarse **while**, para cambiar la condición a mensaje de “Esto es un mensaje” mientras “a” sea distinta de 10.

```
Dim a = 1

Do while a<>10

Msgbox “Esto es un mensaje”

Loop
```

Y también puede utilizarse **until** y **while**, antes o después de ejecutarse el bucle

```
Do until/while condición (es distinta según sea until o while)

Msgbox “Esto es un mensaje”

Loop until/while
```

Como se ve, este bucle nos permite darle varios usos según nos convenga, por lo que termina siendo muy versátil y útil.

Al igual que para los **for**, en el caso de querer salir de un bucle **do**, también contamos con una función de escape, en este caso, **exit do**

Así termina está pequeña introducción a las rutinas básicas de visual basic script, como comentario agrego además, que este lenguaje posee muchisimas funciones, las cuales no detallaré, ya que sería interminable, además de que me centraré más adelante, en explicar funciones expecificas de objetos.

En el caso de querer más documentación al respecto, se pueden descargar la documentación sobre windows script host, donde tendrán la referencia a todas las funciones de las que podemos hacer uso. Windows Script 5.6 Documentation