Trabajo para la asignatura
Programación Orientada a Objetos
junio, 2004

VISUAL BASIC SCRIPT

Enrique Martín Martín

Universidad de Salamanca

Tabla de Contenidos

1. INTRODUCCIÓN

Antes de comenzar a hablar sobre Visual Basic Script (más conocido por la abreviatura de VBScript*)* hay que hablar del lenguaje del cual está derivado, que es Visual Basic.

Hace unos 13 años, el proceso de construir una simple aplicación basada en Microsoft Windows se habría podido describir como complicado, difícil y largo. Construir estas aplicaciones ricas en gráficos no era un proceso trivial antes de la introducción de Visual Basic 1.0 en mayo de 1991. Con Visual Basic, los programadores podían, por primera vez, implementar aplicaciones de Windows en un ambiente intuitivo y gráfico, simplemente arrastrando controles sobre un formulario. Haciendo posible a los programadores profesionales y a los ocasionales maximizar su productividad, Visual Basic conllevó un renacimiento del desarrollo de aplicaciones basadas en Windows[1].

Después de ésta pequeña reseña histórica sobre Visual Basic hay que decir Visual Basic Script es un lenguaje (cuyo nombre indica) de script, es decir que es un lenguaje recortado de otro lenguaje como es el visto anteriormente. Estas versiones se usan para su integración en páginas web. Un código escrito en un lenguaje de script se incorpora directamente dentro de un código HTML y se ejecuta interpretado, no compilado.

Para incorporar un fragmento de código script en una página HTML se introduce el script entre una serie de etiquetas (*tags*) que veremos posteriormente.

Si se está comenzado a programar y Visual Script es tú primer lenguaje se recomienda, dentro de toda la bibliografía existente en el mercado, el libro [3] por su facilidad al explicar los conceptos de VBScript y su funcionamiento.

Este texto trata el lenguaje VBScript desde el principio; sin embargo presuponemos por parte del lector los necesarios conocimientos de HTML y del entorno web.

2. OPCIONES PARA CREAR PÁGINAS WEB

HTML es el lenguaje por excelencia de creación de páginas y documentos web. A diferencia de los lenguajes convencionales, HTML utiliza una serie de etiquetas ASCII especiales intercaladas en un documento ASCII. Dichas etiquetas serán posteriormente interpretadas por los exploradores encargados de visualizar la página con el fin de establecer el formato.

El problema que tiene HTML es que crea páginas estáticas, es decir, que puede visualizar documentos, sonidos, imágenes y otros elementos multimedia, pero el resultado no se actualiza mientras se visualiza.

Para la creación de páginas web dinámicas fueron varias las opciones que se fueron creando (CGI[4], ISAPI[4]) pero al final la opción que se implantó fue la secuencia de órdenes (*script*). Las dos variantes que hay de secuencia de órdenes es JavaScript (basada en Java) y la que estamos estudiando (VBScript).

Estas secuencias de órdenes pueden ser ejecutadas tanto en el servidor como en el cliente. Si quien ejecuta las órdenes es el cliente, el servidor envía a la máquina cliente tanto el código HTML como la secuencia de órdenes después se encarga el cliente en ejecutar e interpretar las dos cosas. La segunda opción (las órdenes son ejecutas en el servidor) fue creada por Microsoft que llamó a este tipo de páginas ASP (*Active Server Page* o Página Activa del Servidor).

Una página ASP dicho de forma sencilla es un fichero con extensión .asp que puede contener: texto, código HTML, secuencia de órdenes y componentes ActiveX. Cuando un

cliente solicita una página de estilo ASP, el servidor se encarga de ejecutar las rutinas *script* y genera como resultado una página en HTML estándar que envía las cliente [7].

Este texto trata de explicar el funcionamiento de VBScript ejecutado en el cliente por lo que no confundir con ASP o lo que es lo mismo páginas activas del servidor donde es el servidor quien ejecuta las rutinas VBScript y el que manda código HTML estándar. A pesar de esto se incluirá un tema sobre como funciona VBScript en una página asp.

3. PRIMEROS PASOS CON VBSCRIPT

Antes de nada hay que ver la estructura de una página HTML estándar:

```
<HTML>
<HEAD>
<TITLE> </TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Como se puede ver una página HTML se compone de cuatro etiquetas básicas junto con las cuatro etiquetas que las cierran:

`<HTML> </HTML>`	Encierra la página web completa
`<HEAD> </HEAD>`	Cabecera de la página web
`<TITLE> </TITLE>`	El título de la página que está en la cabecera
`<BODY> </BODY>`	`Todo lo que es el contenido de la página`

Después de haber visto esta pequeña introducción sobre la estructura de HTML el texto se centra otro vez sobre VBScript. Los lenguajes de Script, como VBScript, se insertan directamente en el listado de una página HTML y realiza ciertas operaciones que el lenguaje HTML, por si sólo, no es capaz de llevar a cabo. Un código VBScript en una página web tiene el siguiente el siguiente aspecto general:

```
<SCRIPT LANGUAGE = "VBScript">
    código Visual Basic Script
</SCRIPT>
```

Este formato es el que se usa para escribir código VBScript ejecutable en el lado del cliente, no del servidor. El código VBScript se puede teclear dentro de la cabecera o del cuerpo de la página web. Lo normal es incluir en la cabecera el código que debe estar en memoria antes de la ejecución de la página y en el cuerpo el que debe ejecutarse con la página. Por supuesto puede haber código VBScript dentro de la cabecera y dentro del cuerpo, simultáneamente, si el diseño de nuestra página lo requiere.

Un primer ejemplo de una página HTML con código VBScript sería el siguiente:

```
<HTML>
<HEAD>
<TITLE> Primer ejemplo en HTML </TITLE>
</HEAD>
<BODY>
```

```
<SCRIPT LANGUAGE = "VBScript">
        ‘ Comentario de ejemplo
        REM Otro comentario de ejemplo
        Document.Write("Esto ha sido escrito por VBScript")
        Variable = 5
</SCRIPT>
</BODY>
</HTML>
```

En este primer ejemplo podemos ver varias cosas interesantes:

El código VBScript se ha escrito en el cuerpo aunque también se podía haber escrito en la cabecera.

En `LANGUAGE = “VBScript”` se podía haber puesto “Javascript” para poder trabajar con Java Script pero se está usando Visual Basic Script se usa “VBScript”.

Para introducir comentarios en VBScript se usa tanto ‘ como REM.

No hace falta definir las variables para utilizarlas (`Variable = 5`).

Una peculiaridad que tienen Visual Basic y que por supuesto tiene también VBScript es que no diferencia entre mayúsculas y minúsculas. Así dondo pone `Variable = 5` se podría haber puesto `variable = 5` o `VaRiAbLe = 5` que el explorador lo hubiera entendido igual.

VBScript utiliza objetos ya definidos, como `Document` que hace referencia a la página del explorador donde se ejecutando dicha ordén. Por lo tanto con el objeto `Document` que está utilizando el método `Write` lo que se está haciendo es escribir en la página el texto `Esto ha sido escrito por VBScript. Hay mas objetos genéricos (con sus métodos) ya creados VBScript pero se verá más adelante.`

4. DEFICIONE DE VARIABLES Y MATRICES

Al contrario que en otros lenguajes de programación, en VB solo existe un tipo general de datos que se conoce con el nombre de `Variant`. En otros lenguajes existen datos de tipo String (Cadena) para almacenar contenidos alfanuméricos, distintos tipos de datos numéricos enteros y en coma flotante, datos booleanos, etc. Esta característica es muy útil, ya que permite reasignar un valor de un tipo a una variable de otro tipo. En la actualidad es el único lenguaje de alto nivel que implementa esta característica.

Las variables se clasifican en subtipos en función del contenido en un momento dado. Así se logra toda la funcionalidad de gestión de datos en lenguajes de alto nivel, pero con una mayor flexibilidad. Para cambiar una variable de un subtipo a otro, es suficiente con asignarle un dato de diferente tipo.

En la siguiente tabla aparecerán los subtipos de datos que hay en *Vbscript*:

String	Datos de tipo cadena (también llamado alfanuméricos).
Byte	Números enteros del 0 al 255.
Integer	Números enteros del -32.768 al 32.767.
Long	Números enteros del -2.147.483.648 al 2.147.483.647.

Single	Números en coma flotante de simple precisión.
Double	Números en coma flotante de doble precisión.
Currency	Números en coma flotante del -922.337.203.685.477,5808 al 922.337.203.685.477,5808.
Boolean	Datos lógicos verdadero o falso (true o false).
Null	Un dato Variant sin definir contenido de ningún subtipo.
Date	Un valor de Fecha / Hora.
Object	Contiene la representación de un objeto.
Error	Identifica los errores mediante un número.

TABLA 1 :Subtipos de datos de Variant

Hay que recalcar que con esta característica nos quitamos de problemas al cambiar el tipo a una variable ya que VBScript lo hace automáticamente y de forma totalmente transparente al usuario.

No es necesario la definición de una variable en VBScript para poder utilizarla pero es una buena práctica de programación ya que así se facilita la búsqueda de errores. Además no es necesario que la variable se defina al principio por que se puede definir en cualquier parte de nuestro código de HTML.

La forma de definir una variable es, primero utilizando una de las siguientes palabras reservadas: `Dim, Public y Private,` y el nombre de la variable, así tendríamos:

```
Dim variable1
Public variable2
Private variable3
```

La diferencia entre ellas es el ámbito en el que las variables declaradas serán válidas. Usando Public la variable será accesible desde cualquier parte del código. Usando Dim será accesible sólo en la función o procedimiento en que se declara; pero si se declara fuera de ellos será accesible a todo el mundo. Utilizando Private, en cambio, la variable sólo será accesible para el nivel en el que está. Ningún procedimiento podrá acceder a una variable privada declarada en fuera de él. Vamos a verlo con un ejemplo:

```
Dim     Variable1      'Accesible fuera y dentro de Pruebas
Public  Variable2      'Accesible fuera y dentro de Pruebas
Private Variable3      'Accesible sólo fuera de Pruebas
Sub Pruebas
        Dim     Variable4      'Accesible sólo dentro de Pruebas
        Public  Variable5      'Accesible fuera y dentro de Pruebas
        Private Variable6      'Accesible sólo dentro de Pruebas
End Sub
```

Se pueden declarar varias variables en una misma sentencia separándolos por comas:

```
Dim Yo, Tu, El, Ella, Nosotros, Vosotros, Ellos
```

Como se ha dicho anteriormente no es necesario la definición de las variables pero si quisiéramos que solo se pudiesen utilizar las variables que se definieran al principio, utilizaríamos la orden `OPTION EXPLICIT.`

Además de poder crearse una variable también se pueden crear matrices, que son un conjunto de elemento del mismo tipo con un mismo nombre que para hacer referencia a cada elemento de la matriz se utiliza uno o varios índice, pudiendo tener matrices de una, dos, tres ... dimensiones. La forma de declararla sería la siguiente:

```
DIM matriz(n)
PRIVATE matriz(n)
PUBLIC matriz(n)
```

Los ejemplos puestos anteriormente son matrices de una solo dimensión y con n elementos, si quisiéramos una matriz de d dimensiones sería:

```
DIM matriz(n)(n2)(n3)...(nd)
PRIVATE matriz(n) (n2)(n3)...(nd)
PUBLIC matriz(n) (n2)(n3)...(nd)
```

Donde tendríamos n * n2 * n3 * ... * nd elementos, hay que tener en cuenta que las matrices comienzan en 0 y así que si creamos una matriz de e elementos tendremos en realidad una de e+1 elementos.

Para hacer referencia a un elemento de la matriz, ya sea de una o de varias dimensiones, sería escribiendo el nombre de la matriz, luego entre paréntesis el elemento que sea y al final darle el valor que queramos:

```
matriz(n)(n2)(n3)...(nd) = valor
```

Un ejemplo de todo lo visto en este apartado sería:

```
<HTML>
<HEAD>
<TITLE> Ejemplo con Variables </TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE = "VBScript">
        OPTION EXPLICIT
        Dim Matriz(7)(6)
        Public Variable = FALSE
        Variable = 5
        Matriz (0)(4) = 8
        Variable = “Hola Mundo”
        Matriz (5)(3) = “HOLA” ‘Aunque se puede tener diferentes
        ‘tipos de datos en una matriz no se suele hacer
        ‘porque va en contra de lo que es una matriz,
        ‘un conjunto de elementos del mismo tipo
</SCRIPT>
</BODY>
```

```
</HTML>
```

5. ESTRUCTURAS DE CONTROL: CONDICIONALES

Las estructuras de control condicionales se usan para que la ejecución del código no sea lineal (de arriba abajo) y además para dar mayor flexibilidad a la página en cuanto a las alternativas que puede tener cada cliente en su navegador web, un ejemplo de esto sería un cliente que tuviera como navegador el mozilla y la página esté optimizada para el explorer de Microsoft.

Hay dos instrucciones condicionales disponibles en el VBScript y son:

```
If...Then...Else

Select Case
```

Se utilizará uno u otro según sea el tipo de condición y el número de alternativas a esa condición que queremos tener.

IF…THEN…ELSE

Esta instrucción condicional se utiliza para evaluar si una condición es `True` o *`False`* y, según el resultado, especificar una o más instrucciones para ejecutar.

La estructura de esta instrucción es la siguiente:

```
If condición Then
      ‘Acciones si condición es TRUE
Else
      ‘Acciones si condición es FALSE
End If
```

En donde se pone condición es la condición que se debe de cumplir para que se ejecuten las instrucciones que van después del `Then`, si no se cumple la condición se ejecutará las instrucciones que van después del `Else`. Además de todo esto hay que aclarar que esta instrucciones se puede definir sin necesidad del `Else` y así no se ejecutaría ningún código si no se cumpliera la condición.

Dentro de la condición se pueden usa una serie de operadores[5], éstos operadores se pueden usar para realizar operaciones aritméticas(suma, resta, multiplicación), también se pueden realizar para operaciones lógicas(negación, disyunción, conjunción) y además se pueden usar para las operación condicionales.

Se pueden usar varios operadores dentro de la condición:

```
If  (  (a+b*c) >= d ) AND (d<>e) ) Then
      ‘Acciones
```

En este ejemplo vemos que para que se realizan las acciones b por c mas a debe ser mayor o igual que d y además d debe ser diferente de e.

Si se quiere expandir la funcionalidad de If...Then...Else se puede usar la cláusula ElseIf para añadir nuevas condiciones y así poder tener diferente posibilidad. Siguiendo el ejemplo anterior se podría ampliar de la siguiente forma:

```
If  (  (a+b*c) >= d ) AND (d<>e) ) Then
      'Acciones
ElseIf (a<b)
      'Acciones2
ElseIf (b=e)
      'Acciones3
End If
```

En este segundo ejemplo del apartado vemos que si no se cumple la primera condición y se cumple que a es menor que b se ejecutan las acciones2, si esta segunda opción no se cumple y b es igual que e se ejecutan las acciones3.

SELECT CASE

La estructura Select Case proporciona una alternativa a If...Then...Elseif para ejecutar de forma selectiva un bloque de instrucciones de entre múltiples bloques. Una instrucción Select Case proporciona capacidad similar a la de la instrucción If...Then...Else, pero hace que el código sea más eficaz y legible.

Una estructura Select Case funciona con una expresión de prueba simple que se evalúa una vez, en la parte superior de la estructura. El resultado de la expresión se compara entonces con los valores para cada instrucción Case en la estructura. Si existe una coincidencia, se ejecuta el bloque de instrucciones asociadas con dicho Case:

```
Select Case A
      Case 1
      'Acciones1
      Case 2
      'Acciones2
      Case 3
      'Acciones3
      Case 4
      'Acciones4
      Case Else
      'Acciones alternativa
End Select
```

La estructura `Select Case` evalúa una expresión una vez en la parte superior de la estructura. En contraposición, la estructura `If...Then...ElseIf` puede evaluar una expresión diferente para cada instrucción `ElseIf`. Puede reemplazar una estructura `If...Then...ElseIf` con una estructura `Select Case` sólo si cada instrucción `ElseIf` evalúa la misma expresión.

6. ESTRUCTURA DE CONTROL: BUCLES

Los bucles se utilizan para ejecutar varias veces una o varias sentencias, estas sentencias se ejecutarán el número de veces que sea necesario hasta que una condición sea falsa o hasta que se halla recorrido completamente una estructura de datos, como puede ser una matriz unidimensional. La condición, según sea el tipo de bucle, puede verificarse al principio o al final y cada vez que se ejecuten las instrucciones se verificará la condición.

Los tipos de bucles que hay son: `FOR, FOR EACH, WHILE WEND` y `DO LOOP.`

FOR

La sentencia `FOR` se utiliza para los bucles, cuando sabemos el número de veces que debemos ejecutar el bucle. Su sintaxis es la siguiente:

```
FOR (inicialización) TO (termino del bucle) STEP (paso)
        ‘sentencias
NEXT
```

La sentencia realiza una repetición desde la `inicialización` hasta el `término del bucle`. Para llevar la cuenta se utiliza una variable, ya se verá en el ejemplo cómo se utiliza esta variable. Con cada ejecución del bucle se ejecutan las sentencias. `NEXT` sirve para delimitar el final del bucle, cuando se encuentra con el `NEXT` se vuelve otra vez al principio del `FOR`, así hasta realizar el número de ejecuciones determinado.

Existe un valor que sirve para indicar lo grandes que se desean realizar los saltos entre ejecución y ejecución, es el valor `STEP`. Un `STEP 2` determinaría que entre ejecución y ejecución la variable se ha de incrementar en 2 unidades. En el caso de no indicar nada se realizan pasos de 1 en 1. También podemos realizar pasos en valores negativos.

Un ejemplo de estos patos sería el siguiente:

```
FOR i=0 TO 6 STEP 2
        matriz(i)=0
NEXT
```

Este ejemplo inicializa un vector matriz a 0 pero solo sus elementos 0, 2, 4 y 6, para inicializarlos se utiliza la variable `i` que va cambiando de valor con cada ejecución del bucle.

FOR EACH

La estructura de control `FOR EACH` sirve para moverse por los elementos de una estructura de datos, como podría ser un vector, y realizar acciones para cada una de los elementos.

Veamos con un ejemplo esta estructura de control:

```
dim tor(20)
for i=0 to 20
    tor(i) = i
next
for each i in tor
    document.write (tor(i))
next
```

En el ejemplo primero creamos un vector y rellenamos con números cada una de sus casillas, con un bucle `FOR`. Más tarde utilizamos el bucle `FOR EACH` para acceder a cada una de las posiciones de este vector de números y escribir en la página cada una de estos números.

En el segundo bucle, se indica que para cada `i` (`i` es el índice con el que podemos movernos en el bucle y en la estructura) dentro de `tor` (que es la estructura, en este caso un vector) haga un `document.write(tor(i))`. Con `tor(i)` accedemos al elemento del vector y document.write() sirve para escribir algo en la página web. Combinadas lo que se escribe es lo que hay en la posición actual del vector.

WHILE WEND

El bucle WHILE...WEND sirve para realizar un tipo de bucle muy utilizado en programación que es el bucle mientras, que se ejecuta mientras que se cumpla una condición. A diferencia del bucle `FOR`, éste se utiliza cuando no conocemos el número de iteraciones que tenemos que realizar.

El bucle funciona de la siguiente manera. Cuando se va a ejecutar, evalúa una expresión y comprueba que esta da resultados positivos. Si es así, ejecuta el cuerpo del bucle (las sentencias que siguen hasta el WEND), en caso contrario se sale. Podemos ver la sintaxis a continuación:

```
WHILE (condición)
          ‘sentencias
WEND
```

Ahora vamos a ver un ejemplo sobre este bucle:

```
option explicit
dim a
a = 0
WHILE (a < 13)
   a = a + 1
   a = inputbox("Dame un valor entero, por favor","Petición de
```

```
número",a,200,100)
WEND
```

El ejemplo realiza una cuenta número a número hasta llegar al 13. En cada iteración del bucle muestra en una ventanita el número actual y ofrece la posibilidad de cambiarlo, ya que la ventanita es una ventana `Input`, que ofrece la oportunidad de cambiar el valor y devuelve ese valor, cambiado o no. Como decíamos, si dejamos el ejemplo sin tocar nada, cuenta hasta 13, pero si introducimos un número en el `inputbox` continúa la cuenta por el número introducido. Si el número introducido es mayor que 13 también se sale del bucle.

DO LOOP

El bucle DO...LOOP es muy versátil. Con el se pueden crear gran variedad de bucles distintos, bucles que comprueben una condición antes de ejecutar el bucle una vez, después de la primera ejecución y con combinaciones con mientras (WHILE) que se cumple una condición o hasta (UNTIL) que esa condición se cumpla. la sintaxis de esta estructura es la siguiente:

```
DO [WHILE | UNTIL (condicion)]
    Sentencias
    .....
LOOP [WHILE | UNTIL (condicion)]
```

Lo que siempre tendremos en estos bucles es el DO y el LOOP, entre estos dos colocaremos las sentencias que queremos ejecutar en cada iteración del bucle. Los bucles tienen que evaluar entre cada iteración si se siguen ejecutando o no, para ello evalúan una condición. Lo versátil de este bucle es que la condición se puede expresar de muchas maneras distintas.

Condición expresada al lado del DO: en este caso la condición se evalúa antes de empezar a ejecutarse el bucle.

Condición expresada al lado del LOOP: en este caso la condición se evalúa después de ejecutarse el bucle. Tiene como diferencia principal frente al otro método que en este caso el bucle se ejecutará por lo menos una vez.

Además de poder expresar la condición en estos dos sitios también se puede construir la condición con un enunciado mientras (WHILE) o un enunciado hasta (UNTIL). Las diferencias semánticas de estas dos posibilidades se trasladan también a su manera de funcionar.

A continuación se muestra un ejemplo donde se podrá ver mejor lo dicho anteriormente:

```
Dim entrada
entrada = ""
DO WHILE (entrada <> "migue")
   entrada = inputbox ("Dime el nombre del
autor","seguridad","migue",2,3)
   if (entrada = "out") then
     msgbox "salgo por la puerta de atras"
      exit do
```

```
   end if
LOOP
```

El ejemplo pide constantemente el nombre del autor de la página y no para hasta que el nombre sea "migue". También tiene el usuario la posibilidad de escribir "out", en ese caso, comprobado con un enunciado IF, se sale del bucle rompiéndolo con la sentencia EXIT DO, utilizada para romper bucles.

En el siguiente ejemplo se podrá observar otra forma de usar DO..LOOP:

```
option explicit
dim cont
dim respuesta
cont = 0
DO
   cont = cont +1
   respuesta = msgbox (cont,69,"Variable del bucle, con valor 6 se sale")
   if (respuesta = 2) then
       msgbox "Cuenta Cancelada",16,"Cancelaste!"
       exit do
   end if

LOOP UNTIL (cont = 6)
```

Realiza una cuenta numérica, que entre cuenta y cuenta se muestra el valor de la cuenta actual en una ventana donde sale un botón de Reintentar y otro de Cancelar. Si se pulsa reintentar se sigue ejecutando el bucle y si se pulsa Cancelar se sale por la puerta de atrás, de manera similar a como se salía en el ejemplo anterior, con EXIT DO.

7. CREACION DE PROCEDIMIENTOS

En VBScript existen dos tipos de procedimientos, el procedimiento Sub y el procedimiento Function.

Un procedimiento Sub es una serie de instrucciones de VBScript que están delimitadas por instrucciones Sub y End Sub, que realizan acciones pero no devuelven ningún valor. Un procedimiento Sub puede tomar argumentos (constantes, variables o expresiones que han pasado por un procedimiento de llamada). Si un procedimiento Sub no tiene argumentos, su instrucción Sub debe incluir un conjunto de paréntesis vacío ().

A continuación viene un ejemplo de cómo es la estructura de un procedimiento Sub:

```
   Sub ConvertTemp()
       temp = InputBox("Escriba la temperatura en grados Fahrenheit",
1)
       MsgBox "La temperatura es " & Celsius(temp) & " grados
Celsius."
    End Sub
```

El anterior procedimiento Sub utiliza dos funciones de VBScript intrínsecas o incorporadas, MsgBox e InputBox, para pedir al usuario alguna información. Después, muestra los resultados del cálculo efectuado a partir de esa información. El cálculo se realiza en un procedimiento Function creado con VBScript.

Un procedimiento Function es una serie de instrucciones de VBScript delimitadas por las instrucciones Function y End Function. Un procedimiento Function es similar a un procedimiento Sub, pero puede devolver un valor. Un procedimiento Function puede tomar argumentos (constantes, variables o expresiones que ha pasado un procedimiento de llamada). Si un procedimiento Function no tiene ningún argumento, su instrucción Function debe incluir un conjunto de paréntesis vacío. Function devuelve un valor asignando un valor a su nombre en una o más instrucciones del procedimiento. El tipo de retorno de un procedimiento Function es siempre Variant.

Vamos a ver un ejemplo de la utilización de Function junto con Sub:

```
Sub ConvertTemp()
    temp = InputBox("Por favor, introduzca la temperatura en
grados F.", 1)
    MsgBox "La temperatura es"   & Celsius(temp) & " grados
Celsius."
End Sub

Function Celsius(fDegrees)
    Celsius = (fDegrees - 32) * 5 / 9
End Function
```

La función Celsius calcula grados Celsius a partir de grados Fahrenheit. Cuando se llama a la función desde el procedimiento ConvertTemp Sub, se pasa a la función una variable que contiene el valor del argumento. El resultado del cálculo se devuelve al procedimiento de llamada y se muestra en un cuadro de diálogo.

Cada dato se pasa a los procedimientos mediante un argumento. Los argumentos sirven como marcadores de posición para los datos que desea pasar a su procedimiento. Puede darles a sus argumentos cualquier nombre que sea válido como nombre de variable. Cuando crea un procedimiento con la instrucción Sub o la instrucción Function, se deben incluir paréntesis después del nombre del procedimiento. Cualquier argumento se coloca dentro de estos paréntesis, separados por comas. Así, en el siguiente ejemplo, fDegrees es un marcador de posición para el valor que se va a pasar a la función Celsius para su conversión:

```
Function Celsius(fDegrees)
  Celsius = (fDegrees - 32) * 5 / 9
End Function
```

Para obtener datos de un procedimiento, debe utilizar un procedimiento Function. Recuerde, este tipo de procedimiento puede devolver un valor mientras que un procedimiento Sub no.

Utilizar procedimientos Sub y Function en el código

Un procedimiento Function en su código siempre debe utilizarse en la parte derecha de una asignación de variable o en un expresión. Por ejemplo:

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

o

```
 MsgBox "La temperatura en grados Celsius es " & Celsius(fDegrees)
& " grados."
```

Para llamar a un procedimiento Sub desde otro procedimiento, puede simplemente escribir el nombre del procedimiento junto con los valores para cualquier argumento necesario, cada uno separado por una coma. La instrucción Call no es necesaria, pero si la utiliza, debe delimitar cualquier argumento entre paréntesis.

El siguiente ejemplo muestra dos llamadas al procedimiento MyProc. Uno utiliza la instrucción Call en el código y el otro no. Los dos realizan exactamente la misma tarea.

Call MyProc(firstarg, secondarg)

MyProc firstarg, secondarg

Tenga en cuenta que los paréntesis se omiten en la llamada cuando no se utiliza la instrucción Call .

8. OBJETOS

Un objeto es un conjunto de variables y subprogramas que modifican dichas variables. Un ejemplo de objeto es la ventana de nuestro navegador. Entre sus variables puede estar una que tenga el texto que contiene la ventana y entre sus subprogramas algunos que permitan cambiar, añadir o borrar dicho texto.

VBScript tiene una serie de objetos ya definidos que podemos usar, estos objetos forman una jerarquía en donde el de mas alto nivel es el objeto WINDOW, que representa a la ventana activa. Todos los demás, son objetos derivados de éste. Veamos cuales son las propiedades y métodos que acepta cada uno, así como los eventos que les afectan.

OBJETO WINDOW

Representa la ventana activa del navegador y es el mas alto de la jerarquía.

Propiedades:

defaultStatus	se refiere al mensaje que aparecerá por defecto en la barra de estado.
document	representa al documento HTML en ejecución en ese momento.
frames []	es una matriz que contiene los frames de la ventana.

history	representa un registro histórico de las páginas visitadas en la actual sesión de uso de Internet.
length	contiene el número total de frames de la ventana.
location	representa a la dirección (URL) actual de Internet.
name	contiene el nombre de la ventana activa.
navigator	representa al navegador que estamos utilizando.
Self	se refiere a la propia ventana. Es el mismo objeto window.
status	es el mensaje que aparece en la barra de estado en un momento determinado.
window	representa a la ventana activa o a otra ventana o sub-ventana de navegación.

TABLA 2: Propiedades de Window

Métodos:

close ()	permite cerrar la ventana activa. Su sintaxis es self.close()
Open ()	permite abrir una nueva ventana, como sub ventana de la actual. Su sintaxis es: `nueva_ventana = window.open ("URL", "Target", "Opciones")`

TABLA 3: Métodos de Window

En la sintaxis expresada, nueva_ventana es el nombre que queremos darle a la sub-ventana. URL es la dirección de la página que queremos que se cargue en la sub-ventana. Target es la dirección del frame donde queremos que se abra la nueva ventana (caso de existir frames) como hacemos en HTML con los hiperenlaces. Opciones son las propiedades de la nueva ventana, de acuerdo a la siguiente tabla.

propiedad	**Tipo de dato**	**Explicación**
toolbar	booleano	Ventana con barra de herramientas.
location	booleano	Ventana con barra de direcciones.
directories	booleano	Ventana con directorios.
Status	booleano	Ventana con barra de estado
menubar	booleano	Ventana con barra de menús.

scrollbars	booleano	Ventana con barras de desplazamiento.
resizable	booleano	Ventana de tamaño redefinible por el usuario.
Width	píxeles	Anchura de la ventana
height	píxeles	Altura de la ventana.
top	píxeles	Posición Y de la ventana
left	píxeles	Posición x de la ventana

TABLA 4: Propiedades nueva_ventana

Eventos:

onLoad	Se ejecuta cuando se carga la página.
onUnload	Se ejecuta cuando se descarga (se cierra) la página.

TABLA 5: Eventos de nueva_ventana

OBJETO DOCUMENT

Representa el documento activo.

Propiedades:

alinkColor	Representa el color de los enlaces activos.
bgColor	Representa el color de fondo del documento.
fgColor	Representa el color del texto
lastModified	Representa la fecha de la última modificación.
linkColor	Representa el color de los enlaces.
location	Representa la URL del documento.
title	Representa el título del documento.

vlinkColor	Representa el color de los enlaces visitados.

TABLA 6:Propiedades de document

Métodos:

Write ()	Escribe un texto.
Writeln()	Escribe una línea de texto.

TABLA 7: Métodos de document

Eventos:

Ninguno

OBJETO FORM

Este objeto se refiere a un formulario empleado en el documento. Se accede a cada formulario mediante un índice: `document.form [índice]`

El número de índice se corresponde con el orden de creación de formulario en el documento.

Propiedades:

action	Representa la URL donde está el programa encargado de procesar un formulario (al que se llama al activar el botón Submit).
length	Es el número de elementos del formulario.
method	Es el método de envío (GET o POST).

TABLA 8: Propiedades de form

Métodos:

submit	Se usa para forzar el envío.

TABLA 9: Métodos de form

Eventos:

onSubmit	Se produce cuando se pulsa el botón Submit del formulario.

TABLA 10: Eventos de form

OBJETO LOCATION

Contiene la URL de la página actual

Propiedades:

href	Representa la propia URL.
pathname	Representa la ruta del disco del servidor donde se aloja la página.

TABLA 11: Propiedades de location

Métodos:

Ninguno

Eventos:

Ninguno

OBJETO NAVIGATOR

Representa al navegador actual.

Propiedades:

appName	Es el nombre del navegador.
appVersion	Se refiere a la versión del navegador.

TABLA 12: Propiedades de navigator

Métodos:

Ninguno

Eventos:

Ninguno

OBJETO HISTORY

Representa el historial de las páginas visitadas en la sesión actual de uso de Internet

Propiedades:

length	Representa la cantidad total de páginas visitadas.

TABLA 13: Propiedades de history

Métodos:

back ()	navega a la página anterior.
forward()	navega a la página siguiente.
go (n)	navega n páginas hacia delante (o hacia atrás, si n es negativo).

TABLA 14: Métodos de history

Eventos:

Ninguno

TIPOS DE EVENTOS

Aquí se listan los principales eventos que se pueden asociar a una imagen, hipervínculo, cadena de texto, etc. A continuación aparecen agrupados según donde se originen (ratón teclado, etc). Estos eventos son los mas usados.

Eventos de ratón

ONCLICK	Se activa con un botón del ratón.
ONDBLCLICK	Se activa si se hace un doble click.
ONMOUSEDOWN	Se activa si se pulsa el botón izquierdo del mouse.
ONMOUSEMOVE	Se activa si se mueve el mouse.
ONMOUSEOVER	Se activa cuando el puntero se sitúa sobre el objeto que incluye al evento.
ONMOUSEOUT	Se activa cuando el puntero sale del objeto que incluye al

	evento.
ONMOUSEUP	Se activa si se suelta un botón pulsado en el mouse (es contrario a ONCLICK).
ONDRAGSTART	Se activa cuando se inicia un arrastre.
ONSELECTSTART	Se activa cuando se inicia una selección con el ratón.
ONSELECT	Se activa cuando se ha realizado una selección con el ratón.

TABLA 15:Eventos del ratón

Eventos de teclado

ONKEYDOWN	Se activa si se pulsa una tecla cualquiera.
ONKEYPRESS	Se activa si se pulsa y suelta una tecla.
ONKEYUP	Se activa cuando se suelta una tecla pulsada.
ONHELP	Se activa si se pulsa la tecla de ayuda (normalmente F1).

Tabla 16:Eventos del teclado

Eventos de enfoque

ONFOCUS	Se activa cuando se entra en el ámbito de un elemento al que está asociado el evento.
ONBLUR	Se activa al abandonar el ámbito del elemento al que está asociado.

TABLA 17:Eventos de enfoque

Eventos de formulario

ONRESET	Se activa al pulsar un botón de reset de un formulario.
ONSUBMIT	Se activa al enviar un formulario.

TABLA 18: Eventos de formulario

Eventos de carga de página

ONABORT	Se activa cuando se aborta la carga de la página.
ONERROR	Se activa cuando se produce un error inesperado durante la carga de la página.
ONLOAD	Se activa cuando se carga la página.
ONUNLOAD	Se activa cuando el usuario descarga la página (es decir, carga otra o pretende salir del navegador).
ONAFTERUPDATE	Se activa si se actualiza o recarga la página.

TABLA 19: Eventos de carga de página

9. FUNCIONES

En VBScript existe una librería de funciones predefinidas que el usuario puede emplear si lo considera conveniente. Al estar predefinidas no es necesario hacer nada para cargarlas en memoria. Se cargan, automáticamente, con el intérprete. Solo es necesario invocarlas donde se necesiten. Estas funciones se clasifican en grupos en base al tipo de datos con el que se emplean. Como son muchas funciones y en cualquier manual (como por ejemplo en [5]) viene como se usa solo se especificará su funcionamiento a las más relevantes encontradas en :

ARRAY (Elemento1, Elemento2, ... , Elemento N)

Devuelve una matriz con los elementos que recibe, separados por comas, como argumento. Si no se le pasan argumentos, devuelve una matriz de cero elementos.

ASC (carácter)

Devuelve el valor ASCII del carácter que recibe como argumento. Si recibe una cadena, devuelve el código ASCII del primer carácter de la misma. Si recibe un dato Null, devuelve un valor Null.

DATE (): No recibe ningún argumento y devuelve la fecha del sistema.

DAY (fecha): Esta función recibe como argumento una fecha y devuelve un número que corresponde al día del mes de la fecha indicada.

EXP (número): Recibe como argumento un número (o variable que lo contiene) y devuelve el número e elevado a la potencia indicada.

HEX (número): Recibe como argumento un número decimal (o variable que lo contiene) y lo convierte en hexadecimal.

INSTR (comienzo, cadena 1, cadena 2): Busca la primera aparición de la cadena 2 dentro de la cadena 1. Los parámetros comienzo y comparación son opcionales. El parámetro comienzo indica a partir de que carácter de la cadena 1 se empieza a buscar la cadena 2. Si se omite, la búsqueda se inicia desde el primer carácter.

INT (número): Recibe un número (o variable que lo contiene). Devuelve la parte entera de un número, truncando los decimales. Si el argumento es un número negativo, esta función devuelve el primer negativo igual o menor que encuentre.

LCASE (cadena): Recibe como argumento una cadena (o una variable que contiene una cadena) y la convierte a minúsculas.

LEN (cadena): Recibe como argumento una cadena (o variable que la contiene) y devuelve la cantidad de caracteres que componen dicha cadena.

OCT (número): Recibe como argumento un número decimal (o variable que lo contiene) y lo convierte en octal.

RND(): Recibe un argumento vacío y devuelve un número aleatorio. Para que funcione correctamente, es necesario incluir en el código VBScript una línea con la instrucción RANDOMIZE.

ROUND (número, decimales): Esta función redondea el número (o variable que lo contiene) y lo devuelve con el número de decimales expresado en decimales.

SCRIPTENGINEBUILDVERSION (): Esta función no recibe ningún argumento y devuelve el número de versión del motor de Script que se está utilizando.

SIN (número): Recibe un número (o variable que lo contiene) que representa a un ángulo en radianes y devuelve el seno de ese ángulo.

TIME (): Esta función no recibe ningún argumento y devuelve una expresión que representa la hora del sistema en formato hh:mm:ss.

UCASE (cadena): Recibe como argumento una cadena (o una variable que contiene una cadena) y la convierte a mayúsculas.

10. ASP

Como ya se ha visto en el apartado “opciones para crear páginas web” una página ASP (Active Server Pages) es un tipo de página con extensión .asp que puede contener texto, código HTML, lenguaje script y componentes ActiveX, que son componentes de programas como puede ser el windows media player (reproductor) o el Microsoft word (procesador de texto) que podemos utilizar en nuestro página. Como ejemplo si en nuestro página ASP introducimos, por ejemplo, el componente ActiveX de Microsoft Word tendremos en la página un procesador de texto ya implementado.

La facilidad para conectar con una Base de datos y extraer datos de la misma dinámicamente visualizándolos en el navegador es la utilidad más practicada de las páginas ASP.

La diferencia en lo referente al VBScript es que el código de éste es ejecutado en el servidor y no en el cliente, lo que se envía al cliente es una página HTML normal. Las dos ventajas más significativas que supone es: la actualización ya que si se actualiza la información en el servidor todas los clientes automáticamente se actualizarán, además el cliente no podrá ver el código escrito en script creado en el servidor y por lo tanto nos quitamos problemas de seguridad.

Otra diferencia importante del VBScript en el servidor es que sus objetos, funciones, en general su lenguaje se ve bastante ampliado en comparación con la versión en el cliente. Un ejemplo de esto es que

En lo referente al código HTML solo hay que cambiar las etiqueta <SCRIPT LANGUAGES = “VBScript”> y </SCRIPT> por <% y %>.

Si se profundizar más en todo lo referente a ASP, la página [6] es muy completa, va desde lo más básico hasta los últimos artículos sobre ASP.

11. CONCLUSIONES

Para la creación de Visual Basic Microsoft se basó mucho en dos lenguajes creados por ellos, como son C y C++. Esto se puede ver en sus estructuras de control que son muy similares, incluido las declaraciones de las variables, la mayoría de diferencias entre éstos lenguajes es por facilitar al programador de VB la programación.

Visual Basic Script es un lenguaje intuitivo y fácil de manejar, y por eso es uno de los primeros lenguajes o sublenguajes que se empiezan a aprender por parte de los usuarios interesados en Internet y las páginas web (el primero sería el código HTML).

Dentro de lo que es el lenguaje hay varias ventajas como es la declaración y asignación de las variables porque no es necesario declararlas y por que el motor de VBScript asigna de forma totalmente transparente un dato al tipo de variable que le corresponde.

En los inconvenientes se puede decir que la no utilización de punteros es una muy importante desventaja además de la mezcla entre lenguaje estructurado y orientado a objetos.

Si se ejecuta en el cliente el código VBScript puede presentar problemas de seguridad, es uno de los motivos de la implantación de la tecnología ASP ya que al ejecutarse en el servidor es mucho mas seguro para las base de datos que no pueden ser manipuladas por el usuario.

En el futuro VBScript se utilizará en un ámbito muy exclusivo debido a las nuevas tecnologías que se están comenzando a implementar, como es .NET donde en ASP.NET ya no es necesario utilizar lenguajes script si no que ya se pueden utilizar lenguajes como VB.NET (última versión de Visual Basic), J# y C#.

Ese ámbito específico que se va a utilizar, que se dijo anteriormente es en el mundo del hacker donde se está usando para explotar (exploits) fallos de seguridad que tiene los navegadores y así poder acceder al ordenador del cliente, también se usan para crear gusanos que se van reproduciendo por la red, uno de los más conocidos es el **I LOVE YOU**.

Para ver más sobre VBScript y la utilización de muchas funciones en ejemplos fáciles de entender hay que ver [2].

12. DOCUMENTOS Y ENLACES RELEVANTES

VISUAL BASIC DEVELOPER CENTER http://msdn.microsoft.com/vbasic/

Página Oficial de Microsoft del Visual Basic.

MSDN HOME PAGE http://msdn.microsoft.com/

Ayuda de Microsoft acerca de cualquiera de sus productos.

SCRIPTING http://msdn.microsoft.com/Scripting/

Página oficial de Microsoft de todo lo relacionado sobre los lenguajes scripts.

ASP.NET Web: Home Page http://www.asp.net/

Página oficial de Microsoft de su nuevo producto ASP.NET .

MANUAL DE ASP. TUTORIAL DE ASP. WEBESTILO. http://www.webestilo.com/asp/

Ya se hizo referencia a está página en el texto, y se sigue recomendando como una página web de ASP en español de gran calidad.

13. REFERENCIAS

[1] **Fc. Javier Cevallos Sierra**. "Visual Basic: Curso de Programación. 2ª edición". Editorial ra-ma . Mayo 1999.

[2] **Paul Lomax, Matt Childs, Ron Petrusha.** "VBScript in a Nutshell". O'Reilly. May 2000

[3] **John Walkenbach**. "VBScript For Dummies". IDG Books. 1996.

[4] **Programación Fácil**. Página Web: http://www.programacionfacil.com/basicasp/indice.htm

[5]**Daniel Rodríguez Herrera.** Pagina Web: http://www.programacion.com/asp/tutorial/vbscript/

[6] **ASPtutor Todo sobre Active Server Pages.** Página Web: http://www.asptutor.com/

[7] **Manual de ASP. Tutorial de ASP. WebEstilo.** Página Web: http://www.webestilo.com/asp/