Manual práctico de JavaScript

1.- Introducción

1.1.- Reseña histórica

Originalmente denominado LiveScript, fue desarrollado por Netscape Communications para crear aplicaciones de Internet en el cliente.

Mas o menos en la misma época, Sun Microsystems lanzó el lenguaje de programación JAVA (que originalmente fue llamado Oak), el que adquirió rápidamente popularidad así que por razones netamente comerciales se le cambia el nombre a JavaScript.

La versión 1.0 apareció con la version 2.0 del navegador de la misma empresa y mas tarde es incorporado en el iExplorer 3.0 de MicroSoft. Poco tiempo después aparece también el VBScript (Visual Basic Script) de Microsoft, un muy buen competidor.

1.2.- Que es un script ?

Un script es una secuencia de ordenes, en un determinado lenguaje, que puede ser ejecutado por un cliente Web desde su navegador y visualizar el documento en que está contenido.

Actuamente los dos lenguajes de script mas usados en paginas Web son JavaScript y VBScript. La inclusion de scripts en los documentos HTML hace que éstos sean más inteligentes. El contenido se genera en forma dinamica, mientras que los valores introducidos en los formularios pueden comprobar localmente, sin necesidad de contar con un servidor y emplear un cierto tiempo en ello. A pesar del nombre, JavaScript, este lenguaje tiene poco que ver con JAVA.

1.3.- Por qué JavaScript ?

Actualmente los navegadores Web mas usados son Netscape Navigator e Internet Explorer, ambos soportan JavaScript, pero solo el Intenet Explorer soporta VBScript.

2.- Conceptos básicos

JavaScript es un lenguaje interpretado en el cliente por el navegador al momento de cargarse la pagina, es multiplataforma, orientado a eventos con manejo de objetos, cuyo código se incluye directamente en el mismo documento HTML.

Hasta entonces ya se usaba HTML y JAVA, pero la aparición del JavaScript produjo una importante revolución, ya que dio al usuario la posibilidad de crear aplicaciones "on-line", es decir modificar páginas web en tiempo real.

2.1.- Caracteristicas

Es simple, no hace falta tener conocimientos de programación para poder hacer un programa en JavaScript.

Maneja objetos dentro de nuestra página Web y sobre ese objeto podemos definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la maquina del usuario, tales como formateo de unidades, modificar archivos etc.

Es dinámico, responde a eventos en tiempo real. Eventos como presionar un botón, pasar el puntero del ratón sobre un determinado texto o el simple hecho de cargar la página o caducar un tiempo. Con esto podemos cambiar totalmente el aspecto de nuestra página al gusto del usuario, evitándonos tener en el servidor un página para cada gusto, hacer cálculos en base a variables cuyo valor es determinado por el usuario, etc.

2.2.- Diferencia con JAVA

La principal diferencia es que JAVA es un lenguaje compilado, mientras que JavaScript es interpretado. JAVA al compilar crea programas independientes, llamados APPLETS que se invocan desde una pagina Web, mientras que el código de JavaScript va incluido en la pagina.

Esta orientado a objetos de forma limitada ya que no maneja los conceptos de clase ni herencia.

En JavaScript no es necesario declarar el tipo de variable, ni debe existir las referencias al objeto antes de ejecutarlo, por lo que se habla de una ligazón dinámica a diferencia de la ligazón estática del JAVA.

2.3.- Principales aplicaciones

Si bien hoy en día, JavaScript, es un lenguaje muy usado, sus principales aplicaciones son:

- Responder a eventos locales dentro de la página, como apretar un botón.
- La realización de cálculos en tiempo real.
- La validación de formularios dentro de una página.
- La personalización de la página por el usuario, que le permitirá tener una página web a su medida.
- La inclusión de datos del propio sistema, como son la hora y la fecha.

Aunque según pasa el tiempo sus aplicaciones se van incrementando.

2.4.- Dónde incluirlo

Antes siquiera de que conozcamos la sintaxis o una primera orden de JavaScript, debemos saber primero cómo se incluye un script dentro de un documento HTML.

El código JavaScript se inserta directamente en nuestra pagina HTML. Hay cuatro (4) maneras de hacerlo:

2.4.1..- En el cuerpo del documento

Es decir entre las etiquetas <BODY> y </BODY> usando el comando **<SCRIPT>**

```
<HTML>
<HEAD><TITLE>Titulo</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!-- escondemos el código>
document.write("Hola que tal");
// fin de esconder -->
</SCRIPT>
</BODY>
</HTML>
```

Este código se ejecuta inmediatamente al cargar la pagina y lo que produce es un texto sobre la pagina, para ello use el método **write** del objeto **document**, que representa al documento actual[1].

Observa que se emplea un comentario <!-- --> para ocultar el código a los navegadores que no soportan JavaScript.

2.4.2.- En archivo aparte

En este caso todo el código del script esta situado en otro archivo y se hace una llamada.

```
<HTML>
<HEAD><TITLE>Titulo</TITLE></HEAD>
<BODY>
<SCRIPT SRC=codigo.js>
</SCRIPT>
</BODY>
</HTML>
```

Observa que aquí no fue necesario esconder ningún código ya que los navegadores que no soporte el comando **SCRIPT** simplemente lo ignorarán.

2.4.3.- Usando manejadores de evento

[1] NOTA: Excepto en texto encomillado, JavaScript es sensible a mayúsculas y minúsculas, por lo que tendrás que tener el cuidado al usar por ejemplo document.write de escribirlo así, en minúsculas o no se interpretara. Cualquier error simplemente es ignorado. Se puede usar la comilla simple para los valores de atributos.

Los comandos de JavaScript son evaluados inmediatamente al cargarse la pagina. Las funciones son almacenadas, pero no ejecutadas, hasta cierto evento.

```
<HTML>
<HEAD><TITLE>Titulo</TITLE></HEAD>
<BODY>
<A HREF="" evento=método o llamada a función interna>algo</A>
</BODY>
</HTML>
```

Observa que ahora es un evento el que dispara.

2.4.4.- Haciendo una llamada a función

Dentro de la cabecera, después del titulo. Es decir, entre los comandos </TITLE> y </HEAD> y luego la llamada a la función en el cuerpo.

```
<HTML>
<HEAD>
<TITLE>Titulo</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!-- escondemos el codigo>
function Ver() {
  alert("Le dije que NO !");
}
// fin de esconder -->
</SCRIPT>
</HEAD>
<BODY>
No haga Clic <A HREF="Ver()">AQUI</A>
</BODY>
</HTML>
```

Observa que aquí se definió la función en la cabecera, pero recién se ejecuta al hacer clic en el enlace, que es el evento que llama a la función a la cual se le para un parámetro.

3.- VALORES, CONSTANTES Y EXPRESIONES

3.1.- Tipos de datos

JavaScript acepta diferentes tipos de datos:

Tipo	Descripción	Ejemplo
Números	Pueden ser números enteros o reales	473.1416
Cadenas	Cualquier texto que este encerrado entre comillas. Pueden ser simples o dobles	"Hola que tal..."
Lógicas	También llamados Booleanos, son las que toman uno de dos valores de verdadero o falso	true o false
Nulas	Es cuando la variable no toma ningún valor	null

3.2.- Valores y constantes

Tipo	Descripción	Ejemplo
Números		
Enteros Decimales (base 10)	Es una secuencia de dígitos (0-9) que no comiencen con 0	1999
Enteros Hexadecimales (base 16)	Una secuencia de dígitos (0-9) y letras (A-F) que comienza con 0x	0xE477
Enteros Octales (base 8)	Secuencia de dígitos (0-7) que comiencen con 0	0777
Punto flotante	Puede tener un entero decimal, un punto, una fracción (otro numero decimal), un exponente que consiste en la letra **e** seguida de un entero, el cual puede llevar un signo (+ o -).	3.14159, -2e4, 5e-12
Cadenas		
Cadenas de caracteres	Consta de uno o mas caracteres encerrados entre comillas simples o dobles	"Hola", '1999'
También pueden usar los siguientes caracteres	\f indica un avance de pagina (Form feed) \n Indica nueva línea (New Line) \r Indica un retorno de carro (Carruage return) \t Indica un tabulador (Tab) \" se puede incluir comillas. Ej: "José \"Chemo\" del Solar"	
Logicas		
Logicas	Verdadero o falso	true o false
Nulas		
Nulas	Es cuando la variable no toma ningún valor	null

3.3.- Expresiones

Es cualquier conjunto valido de constantes, variables y operadores que al evaluarse dan un único valor del tipo numérico, cadena o lógico.

3.4.- Variables

Las variables son elementos a los cuales les daremos un valor estático, o un valor totalmente dinámico. Para definir una variable lo podemos hacer de varias formas.

- La primera, es la siguiente: var mivariable; en la cual la sentencia var hace que JavaScript, cree una variable.

- Pero también podemos definir la variable justo antes de usarla:

```
...aquí sentencias JavaScript
var mensaje = "HOLA";
...aquí más sentencias
```

Probemos cómo se les pueden dar valores a las variables; podemos hacer que aparezca un cuadro en el cual podamos insertar un texto, la siguiente fuente es la usada para hacer el siguiente ejemplo:

```
<html>
<head>
<title>Probemos la función creada</title>
<script language="JavaScript">
<!--
function pregunta()
{
var nombre;
nombre = window.prompt("Escribe tu nombre:","");
window.alert("Hola "+nombre);
} //-->
</script>
</head>
<body>
<form>
<input type=button value=")-=ºPÚLSAMEº=-(" onclick="pregunta()">
</form>
</body>
</html>
```

3.5.- Variables tipo matriz

La sintaxis para crear la matriz es la siguiente: *nombre=new Array(elementos)*, posteriormente para usarla se usa *nombre[# de elemento]*.

3.6.-Operadores

Aritmeticos	
+	Adición
-	Sustracción
*	Multiplicación
/	División
%	Modulo
++	Incremento
--	Decremento
-	Negación
Cadena	
+	Concatenación.
Nota: Cuando se opera un valor de cadena con un numerico, el resultado es una cadena.	
Sobre BITs	
AND o &	Devuelve un 1 en cada bit para el cual ambos operandos sean 1 y 0 en el resto.
OR o |	Devuelve un 0 en cada bit para el cual ambos operandos sean 0 y 1 en el resto.
O-EX o ^	Devuelve un 1 en cada bit para el cual uno de los operandos sea 1 y el otro 0 y 0 en el resto.
NOT o ~	Devuelve un 1 en cada bit para el cual el operandos sean 0 y viceversa.
Lógicos	
&&	Devuelve verdadero si ambos operandos son verdaderos y falso en el resto.
||	Devuelve falso si ambos operandos son falsos y verdadero en el resto.
!	Devuelve falso si operando es verdadero y viceversa.
Comparación	
==	Devuelve verdadero si ambos operador son iguales.
!=	Devuelve verdadero si ambos operador son diferentes.
>	Devuelve verdadero si operador izquierdo es mayor al derecho.
>=	Devuelve verdadero si operador izquierdo es mayor o igual al derecho.
<:	Devuelve verdadero si operador izquierdo es menor al derecho.
<=	Devuelve verdadero si operador izquierdo es menor o igual al derecho.
Asignación:	
X=y	Asigna a x el valor de y
X+=y	Asigna a x el valor de x+y

x-=y	Asigna a x el valor de x-y
X*=y	Asigna a x el valor de x*y
x/=y	Asigna a x el valor de x/y
x%=y	Asigna a x el valor de x%y
x<<=y	Asigna a x el valor de x=x<x<y
x>>y	Asigna a x el valor de x=x>x>y
x>>>=y	Asigna a x el valor de x=x>x>x>y
x&=y	Asigna a x el valor de x=x&y
x^=y	Asigna a x el valor de x=x^y
X|=y	Asigna a x el valor de x=x|y

4.- Inicio al control del flujo del prgrama.

Existen dos grandes tipos de sentencias de control, las sentencias estáticas, y las dinámcas o blucles. Entre las estáticas, encontramos la sentencia if la cual se utiliza de la siguiente forma:

```
if ( sentencia de condición )
{
grupo de sentencias JavaScript que se ejecutarán
en caso de que la sentencia de condición sea "verdadera"
}
```

Un ejemplo claro de uso de la sentencia if sería este:

```
var esto = '25';
if ( esto < 100 )
{
window.alert('La variable esto es menor que 100');
}
else
{
window.alert('La variable no es menor que 100');
}
```

Por supuesto el ejemplo anterior, no tiene ningún sentido, pues la variable siempre será menor que cien; si piensan, esto nos puede ser de ayuda, quiero decir, si en lugar de usarse la variable esto de una forma estática, sino que su valor variase a través de usar un window.prompt o una entrada de formulario o de cualquier otra forma, si tendría alguna utilidad: veamos un ejemplo de utilización de las sentencias if y else, esta última abarca todas las sentencias que no abarquen los "if" que están por encima del mismo, en el ejemplo anterior serían todos los números mayores que 100 y el propio 100.

En este ejemplo demostraremos que si a una variable le damos el valor window.confirm este nos devolverá dos valores: 1 si el usuario pulsara el botón de *Aceptar*, y 2 si pulsase el de *Cancelar*:

```
var aquello = window.confirm("¿Cree Ud. que funcionará?");
if ( aquello )
{
window.alert("Botón Aceptar pulsado.");
}
else
{
```

```
window.alert("Botón Cancelar pulsado.");
}
```

Espero que se haya entendido el ejemplo anterior y las sentencias if y else pues la siguiente sentencia funciona de la misma manera, pero es un poco más ordenada y a su vez restringida; no permite hacer comparaciones múltiples; veamos pues la sentencia switch. Esta sentencia se usa de la siguiente forma:

```
switch ( variable )
{
case 1: sentencias break;
case 2: sentencias break;
case 3: sentencias break;
case 4: sentencias break;
case 5: sentencias break;
default: sentencias break;
}
```

La sentencia switch es muy sencilla de usar, después del case se escribe el valor numérico que debe cumplirse para que se ejecuten las sentencias, y después de estas, un break; que le dirá al intérprete de comandos que las sentencias han acabado; y por último la sentencia default: la cual hace la función del else, es decir, toma los valores no recogidos en el resto de sentencias del bloque.

5.-Control del flujo del programa (bucles)

Existen dos tipos: los bucles normales, y los bucles de retardo de tiempo; estos últimos los usaremos para interactuar con los elementos del reloj. El ejemplo siguiente, muestra cómo una variable es modificada y presentada a través del bucle for. El ejemplo sería el siguiente:

```
function Bucle()
{
var estoesinteresante = "";
   for (var i = 0; i<=10; i++ )
   {
   estoesinteresante += i+".\n";
   }
window.alert(estoesinteresante);
}
```

Expliquemos el ejemplo anterior: 1º declaramos la función y la variable y le damos valor en blanco; 2º declaramos el bucle, en el cual la primera zona declara e inicializa la variable, la segunda pone la condición a cumplirse para que se ejecute el bucle, y la tercera indica el cambio que se le hace a la variable; en este caso se llama *incremento* de i; 3º se invoca el objeto window.alert para que muestre el contenido de la variable.

Ahora veremos otro bucle: el while. Este bucle es mucho más sencillo que el bucle for, pues simplemente ejecuta algunas sentencias mientras la sentencia sea verdadera, por ejemplo:

```
function Bucle_While()
{
var variable;
variable = 1;
   while ( variable < 3 )
   {
   variable++;
   window.alert(variable);
   }
}
```

Esta función ejecuta la sentencia *incremento de variable* y muestra un cuadro de diálogo, mientras variable sea menor que 3.

Llegamos al final de las sentencias de control del programa; ya sólo nos quedan los tipos de bucle llamados de *retardo*, estos se usan con la sentencia setTimeout(a,b) (recordar que JavaScript entiende mayúsculas y minúsculas). Esta sentencia recibe dos valores, "a", que es la función a la cual queremos llamar, y "b", que es el tiempo en milisegundos que JavaScript esperará antes de llamar a la función "a". Ejemplo:

```
setTimeout("MiFuncion()",5000);
```

6.- Inicio a formularios

Una de las cualidades de JavaScript, es la interacción con los elementos de un formulario HTML, es decir, que podemos comprobar si las entradas de un formulario son como deberían de ser, y que no estén en blanco. Para ello, tendremos que darle un nombre al formulario, y un nombre a cada elemento al que queramos acceder. Después, simplemente, modificar a muestro antojo las propiedades del elemento del formulario, pero con JavaScript. Un ejemplo claro sería el siguiente:

```
<html><head><title>P&aacute;gina de prueba</title>
<script language="JavaScript" type="text/javascript">
<!--
function comprobar()
{
   if ( document.formu.texto.value == "" )
   {
   window.alert('Por favor, rellene todos los campos.');
   }
   else
   {
   window.alert('Formulario completamente relleno.');
   }
}
//-->
</script>
</head>
<body>
<form name=formu>
<input type=text name=texto>
<input type=button value="%$% ESTOY AQUÍ %$%" onclick="comprobar()">
</form>
</body>
</html>
```

7.- Formularios, barra de estado y apertura de nuevas ventanas

En el capítulo anterior aprendimos a validar entradas de formulario, por tanto en este capítulo enseñaremos cómo validar un campo de e-mail; aunque lo valide de una forma bastante arcaica es válida para lo que queremos aprender:

E-Mail:

El código sería el siguiente:

```
<html><head>
<script language="JavaScript" type="text/javascript">
<!--
function Comprueba()
{
if (document.formu.email.value.indexOf('@') != -1)
{
window.alert('El campo tiene al menos 1 @, y parece ser correcto.');
}
else
{
window.alert('Introduzca bien el E-Mail porfavor.');
}
}
//-->
</script>
</head><body>
<form name=formu>
E-Mail: <input type=text name=email><br>
<input type=button value="--<-<@ PULSA @>->--" onClick="Comprueba()">
</form>
</body><html>
```

Este código no debería tener mayor dificultad para ser entendido, pero si se fijan he incluido una nueva función que a partir de ahora empezaremos a usar cuando validemos entradas de formulario, esta es *a*.indexOf(*b*,*c*) donde *a* equivale a la variable de cadena de caractéres en la cual se buscará la subcadena *b*, empezando por un número determinado en *c*, esta función devualve el punto en el cual se encuentra la subcadena *b*, y devuelve -1 si no se encuentra la subcadena; de todas formas continúo con la tradición y les pido me escriban un e-mail en caso de que no lo entiendan.

Me falta por decir que cualquier atributo es modificable con JavaScript de forma dinámica, así que investiguen y si tienen dudas mándenma un e-mail.

La barra de estado está situada normalmente en la esquina izquierda del navegador, pero esto es variable, así que búsquenla en su explorador, y empecemos a trabajar con ella. Para ver un ejemplo de barra de estado pulse el botón, el código de la página abierta está debajo:

```
<html>
<head>
```

```
<title>Tutorial JavaScript -> BARRA DE ESTADO</title>
</head>
<body onLoad="window.defaultStatus='Hola, yo soy la barra de estado.';">
<center>
<h2>Mire en la barra de estado</h2>
<form>
<input type=button value=Cerrar onClick="window.close(self)">
</center>
</form>
</body>
</html>
```

Por supuesto podemos hacer scripts que al pulsar un botón se escriba algo en la barra de estado, o que se borre, o que al pasar el mouse se cambie... pero esto ya lo iremos viendo más adelante. Ahora explicaré los eventos más comunes que usará un programador JavaScript como Ud.:

	Estos eventos se activarán cuando...		
onMouseOver	...el mouse esté encima	onUnLoad	...se cierre la ventana
onMouseOut	...el mouse **no** esté encima	onAbort	...se pare la carga de una imágen
onClick	...se *clickcee* encima	onSelect	...se seleccione algo
onLoad	...se cargue la página	onCopy	...se copie algo

En este capítulo vamos a proponer unos ejercicios los cuales les recomiendo hagan, pues así practican y aprenden a descubrir técnicas y tácticas como programadores que son, así que *valor y al toro*: (en el siguiente capítulo pondré las soluciones)

1. Script que al pasar el mouse por encima de un vínculo se escriba en la bara de estado una explicación del vínculo.
2. Página con script que valide un formulario con campo de e-mail, nombre y apellidos. (de la forma explicada)
3. Página que pregunte por medio de un window.prompt el nombre, y que escriba en la barra de estado *Hola nombre que pase un buen rato*.

Ir a la página de soluciones.

Para abrir ventanas nuevas, para desplegar banner publicitarios, o algún anuncio importante, o como por ejemplo lo que hice para enseñarle la barra de estado, se usa el window.open(a,b,c), la cual recibe tres valores, *a* que es la URL de la página, (si no se pone nada, se crea una ventana en blanco), *b*, tipo de ventana, usaremos MsgWindow, y *c*, que son las propiedades de la ventana descritas a continuación, estas propiedades se ponen separadas por "," y después de la "," **no** hay que poner espacio. Las propidades son las siguientes:

status	yes o no	menubar	yes o no
width y height	valor	directories	yes o no
...	...	resizable	yes o no

De todas formas investiguen porque existen muchas más propiedades de las ventanas que no considero muy importantes.

Veamos un ejemplo de abrir ventanas mediante un botón con JavaScript:
El código sería el siguiente:

```
function Abrir_Ventana()
{
```

```
var propiedades = 'menubar=no,width=300,height=300,status=yes,resizable=no';
var ventana = window.open("","MsgWindow",propiedades);
var codigo = '<HTML><HEAD><TITLE>Página de prueba</TITLE></HEAD><BODY>';
codigo += '<H3>Esto es muy útil para desplegar publicidad</H3><br>';
codigo += '<H3>O para desplegar algún cartel de aviso</H3><br>';
codigo += '<H3>O para lo que se te ocurra</H3><br>';
codigo += '<form><center>';
codigo += '<input type=button value=Cerrar onClick="window.close(self)">';
codgio += '</center></form></BODY></HTML>';
ventana.document.write(codigo);
}
```

Note que lo único que hacemos es escribir el código de una página web, en una ventana que hemos abierto con JavaScript; esto es muy útil, pero mucho más útil, es controlar la ventana que hemos abierto mediante funciones JavaScript, como por ejemplo escribir algo en la barra de estado, cerrarla, mandarla a una URL mediante la sentencia *a*.window.location=*b*; donde *b* es la URL a la cual queremos mandar la ventana *a*, si a *a* no le damos valor tomará el de self, que se refiere a la ventana en la cual se ejecuta la función, es decir la que estás viendo. Esto quizá te parezca un poco raro, así que por favor si no entiendes algo mándame un e-mail. (No me gusta romper la tradición de pedirte un e-mail :-P)

capítulo vamos a profundizar un poco en los eventos posibles en JavaScript, lo primero que debemos saber, es que para los navegadores de Netscape 4.7 y anteriores, no todos los eventos son aplicables a todos los elementos, esto quiere decir, que el elemento <IMG>, por ejemplo, no admite los eventos onMouseOver ni onMouseOut, pero si onClick, y así pasará con muchos elementos, pero al contrario, los navegadores Netscape 6.0, Internet Explorer 4.0 y posteriores, y Opera 5.0 si lo entienden, por lo general con Internet Exploer, todos los elementos soportan todos los eventos, pero sin embargo el resto de los navegadores, no todos los elementos permiten todos los eventos. Sabiendo esto, podemos hacer dos cosas, *programar por lo bajo o por lo alto*, esto es, podemos pensar que que todos los usuarios tienen el N 3.0 Gold, o IE 2.0, con lo cual nuestros scripts serán un poco *arcaicos* y algunas veces con poca funcionalidad, o podemos pensar que el que no tenga N 6.0, Opera 5.0 o IE 5.5 como mínimo no sabe lo suficiente como para usar nuestros scripts, por supuesto estos dos tipos de pensamiento son completamente erróneos, siempre hay que intentar *programar por lo bajo*, pero sin pasarnos, pues no tiene sentido que podamos hacer muchas cosas, pero no las hagamos por si los visitantes no están lo suficientemente actualizados como para usarlos al 100%. En conclusión, debemos pensar que el estándar está en N 4.5+ o IE 4.0+, pero siempre encontrarás a alguien que tengan navegadores más antiguos, por tanto, JavaScript nos permite averiguar la versión y el navegador que el usuario está usando en el momento de visualizar la página, esto se hace con las siguientes funciones:

navigator.appName	...este devuelve el nombre del navegador
navigator.appVersion	... y este la versión, pero será más eficiente usar la de abajo...
navigator.appVersion.charAt(0)	...devuelve el primer caracter de la versión del navegador.

En la útima sentencia, aparece una función que aún no había explicado *a*.charAt(*b*), la cual devuelve el caracter de la cadena *a*, situado en la posición *b*, (cabe destacar que las cadenas empiezan en el caracter 0). Si no algo no ha quedado claro, ya sea tanto de la teoría de los eventos y los navegadores, como de la función *charAt*, esribidme un e-mail. (Me gusta seguir la costumbre del e-mail).

Vamos a profundizar un poco más con esto de los eventos, vamos a ver un ejemplo de una imágen que cambia al pasar el mouse por encima, esto es muy sencillo, pero si no se tiene el cuidado

suficiente, puede quedar un script bastante malo, para que salga bien es importante que se use la técnica de la precarga, que consiste en bajar las imágenes al cargar la página, para que al pasar el mouse por encima, el cambio sea automático, y no tarde mucho tiempo en cambiarse, el script sería así:

```
<html>
<head>
<title>Script de Cambio de imágenes con evento onMouseOver y técnica Precarga</title>
<script language="JavaScript" type="text/javascript">
<!--
var mousefuera = "";
var mousedentro = "";
function Precarga()
{
mousefuera = new Image(181,29);
mousedentro = new Image(181,29);
mousefuera.src = "imagen1.gif";
mousedentro.src = "imagen2.gif";
}
//-->
</script>
</head>
<body onLoad="Precarga()">
<center>
<a href="javascript:alert('El vínculo lo diriges donde quieras');"
onMouseOver="imagen.src = mousedentro.src"
onMouseOut="imagen.src = mousefuera.src">
<img src="imagen1.gif" name="imagen" border=0 width="181" height="29">
</a>
</center>
</body>
</html>
```

En el ejemplo anterior se incluye un nuevo objeto que es Image(*a*,*b*), donde *a* es el ancho de la imágen, y *b* es el alto, y después se le da el atributo .src para que la imágen se cargue, luego creamos la imágen y las etiquetas de hipervínculo, y a la imágen le damos un nombre y a las etiquetas de vínculo, les asociammos los eventos onMouseOver y onMouseOut, de tal forma que el script, creado haría algo así:

Nintendo 64

Ahora vamos a tratar el evento onLoad que se ejecuta al cargarse la ventana:
Dentro de etiquetas <Script></Script>.

```
function Abrir()
{
var publicidad = window.open("","msgWindow","menubar=no,directories=no,width=400,height=400");
publicidad.window.location = "banners.html"; // Poner la página que se quiera
}
```

Y en la etiqueta <BODY ...> añadir:

```
onLoad="Abrir()"
```

Les recomiendo intenten inventarse ejemplos, para desarrollar sus dotes como programadores JavaScript que son, aunque de todas formas en los anexos del tutorial encontrarán múltiples ejemplos

sobre Scripts típicos que puedan usar en su web, aunque saben que si en algún momento tienen alguna duda, pueden mandarme un e-mail.

Capítulo 6: Arrays

¿Qué es un *Array*? Se preguntarán... Pues un array, es un grupo de variables que se denominan con un mismo nombre y un número (para diferenciarlas unas de otras), esto lo entenderán muy bien cuando lean algún ejemplo:

```
var miarray = new Array();
miarray[0] = 'Esto es el primer elemento.';
miarray[1] = 'Esto es el segundo elemento.';
miarray[2] = 'Esto es el tercer elemento.';
...
```

Noten que el primer elemeno es el múmero **0**, el segundo es el **1**, y así sucesivamente, por supuesto podemos manejar muchos elementos en un array, no sólo 3 o 4, tendrá todos los que quieran. Imaginemos que queremos cargar las imágenes que cada una es un número, lo podríamos hacer así:

```
var imagenes = new Array();

function Precarga()
{
   for ( var i = 0 ; i <= 9 ; i++ )
   {
   imagenes[i] = new Image();
   imagenes[i].src = 'images/'+i+'.gif';
   }
}
```

Esto cargará 10 imágenes, 0.gif, 1.gif, 2.gif, etc... que se guardan en un array que se llama imagenes.

Y al fin, el ejemplo final... **El Buscador**. Este Script, es un buscador que guarda sus registros en un array, y que busca palabras concretas, no es muy eficiente, pero su potencia es grandísima, quiero decir, que si el número de registros es muy grande, será un buen buscador, y además usted investiga cómo hacerlo para que soporte más palabras clave, es un buscador muy potente, pero si usa el mismo que aquí le pongo, por supuesto tendrá usted un buscador OBSOLETO. Ahí va:

```
<HTML>
<HEAD>
<title>Buscador JavaScript</title>
<script language="JavaScript" type="text/javascript">
<!--
var resultado = '';
var codigo = '';
var Clave = new Array();
var Url = new Array();
var Descripcion = new Array();

Clave[0] = "juegos";
Url[0] = "http://www.upss.go.to/";
Descripcion[0] = "Sitio Web con múltiples cosas sobre consolas y Sobre HTML y JavaScript";

Clave[1] = "natacion";
Url[1] = "http://www.cnlasrozas.nav.to/";
Descripcion[1] = "Sitio con mucha información sobre natación.";

Clave[2] = "javascript"
Url[2] = "http://www.tusitio.com/";
Descripcion[2] = "Aquí puede ir tu sitio";
```

```
// Pon todas las que quieras...

function Busca( str )
{
   if ( str == '' || str == null )
   {
   window.alert('No tengo nada que buscar.');
   document.busqueda.cadena.focus();
   }
                else
                {
resultado = window.open('','msgWindow','status=no,menubar=no,width=400,height=400');
codigo += '<html><head><title>Resultados</title></head><BODY link=blue vlink=blue alink=blue>';
codigo += '<h3><center>De la cadena: <i>'+str+'</i>.</center></h3>';
codigo += '<hr width=50%>';
   for ( var i = 0 ; i <= Clave.length ; i++ )
   {
      if ( str == Clave[i] )
      {
      codigo += "<li><a href="+Url[i]+">Url</a><br>";
      codigo += Descripcion[i]+"</li>";
      }
   }
codigo += '<form><center>';
codigo += '<input type=button value=Cerrar onClick="window.close(self)">';
codigo += '</center></form></body></HTML>';
resultado.document.write(codigo);
                }
}
//-->
</script>
</HEAD>
<BODY>
<form name=busqueda>
<input type=text name=cadena><br>
<input type=button value=Buscar onClick="Busca(busqueda.cadena.value);"><br>
</form>
</BODY>
</HTML>
```

Es quizá un código un poco más largo de lo que estamos acostumbrados a programar, pero si se fijan e intentan entenderlo paso a paso, es un código bastante sencillo. Lo primero generamos la página con el formulario de búsqueda, y damos a un botón la propiedad de que al ser pulsado envíe a la función Busca (str), un parámetro que es la entrada del formulario. Esta función comprueba la "Base de Datos", que no es más que un conjunto de arrays que se relacionan estre si. Para comprobarla, usa un bucle for, al cual se le anida una sentencia if, si es correcto se le añaden las entradas a la variable codigo, y al final se abre una ventana a la cual se le escribe codigo. Si no lo entienden, les pido que me escriban un e-mail. El ejemplo creado, muestra una página tal que así:

Capítulo 7: Trabajando con Frames y operador Ternario (?:)

Por supuesto, JavaScript permite modifcar la Url de los frames de una página, el ancho, y muchas cosas, de todas formas no vamos a introducirnos mucho por este tema. Repasemos un poco la teoría sobre los frames que todos deberían saber antes de leer este capítulo. Existe una página *padre*, donde están declarados los frames, y desde la misma son llamados, a estos se les llama *hijos*, para JavaScript parent y al resto por sus nombres. Supongamos que tenemos una oágina padre con el siguiente código:

index.html

```
<html>
<head>
<title>Menu</title>
</head>
<frameset rows="40,*" frameborder=0 framespacing=0>
 <frame src=menu.html name="menu" scrolling="no">
 <frame src=index2.html name="cambiado">
</frameset>
</html>
```

index2.html

```
<html>
<head><title>Titulo</title></head>
<body>
<p><center>ESCOGE UNA OPCI&Oacute;N</center></p>
</body>
</html>
```

menu.html

```
<html>
<head>
<title></title>
<script src="cambio.js"></script>
</head>
<body>
<form name=menu>
<center>
<input type=button value="Buscador" onclick="Cambia(../lesson06','#busca')">
<input type=button value="Diseño Web" onclick="Cambia('webdes',')">
<input type=button value="Trucos" onclick="Cambia('tricks',')">
<input type=button value="JavaScript" onclick="Cambia('../index')">
<input type=button value="Otro" onclick="Cambia('unomas',')">
</center>
</form>
</body>
</html>
```

cambio.js

```
function Cambia(url,mas)
{
parent.cambiado.window.location=url+'.html'+mas;
}
```

Antes de continuar, les enseñaré una forma de no tener que incluir el código JavaScript en la página HTML, es con la etiqueta script a la cual se le dará el atributo src cuyo valor será la Url del archivo con extensión .js que contiene el código, es muy útil, pues despeja mucho la página, como podreis observar, he incluido esta sentencia en la cuarta línea del código de menu.html para que se vayan familiarizando con esta sentencia. Este ejemplo lo pueden ver pulsando aquí ------->

Un último concepto sobre frames, JavaScript, guardará todos los frames en un Array llamado *frames*, según el orden de declaración empezando por supuesto por el cero, por ejemplo en nuestra página el elemento document.frames[0]sería el frame del menu.

Vayamos al operador ternario (?:), este se usa con la siguiente sintaxis:

```
( expresion booleana ) ? resultado1 : resultado2;
```

Se ejecutará resultado1 si la expresión es verdadera, y de lo contrario se ejecutará la resultado2.Ejemplo:

```
var navegador = (document.all) ? explorer : netscape;
```

Se puede observar que el código anterior es equivalente a lo siguiente:

```
var navegador;
if (document.all) navegador = "explorer";
else navegador = "netscape";
```

Veamos un ejemplo de utilización que nos puede ahorrar muchos quebraderos de cabeza y múltiples líneas de código:

```
var isIE = (document.all) ? true : false;
var isNN = (document.layers) ? true : false;
```

El código anterior inicializa dos booleanos,isIE y isNN, el primero tendrá el valor true, en caso de que document.all sea cierto, es decir, si el navegador es Microsoft Internet Explorer, de cualquier otro modo, tomará el valor false, lo mismo ocurre con isNN, pero con la expresión document.layers, que será cierta cuando el navegador se trate de Netscape Navigator. Pero noten la potencia de este operador, podemos tener varias expresiones, de la siguiente forma:

```
var isIE4sup = (document.all && navigator.appVersion.charAt(0) >= 4) ? true : false;
var isNN3sup = (document.layers && navigator.appVersion.charAt(0) >= 3) ? true : false;
```

Se pueden hacer infinidad de cosas con este operador, pero como con todo, eso tiene que surgir del propio programador. Les enseñaré un ejemplo práctico para la detección del navegador:

```
function Detecta()
{
var isIE4sup = (document.all && navigator.appVersion.charAt(0) >= 4) ? true : false;
var isNN3sup = (document.layers && navigator.appVersion.charAt(0) >= 4) ? true : false;
if (isIE4sup) alert('Estás usando el navegador MSIE 4.0 o superior.');
else if (isNN3sup) alert('Estás usando el navagador Netscape 3.0 o superior.');
else alert('No usas ni MSIE ni Netscape.');
```

Capítulo 8: Iniciación a objetos, *split* y exámen.

Les explicaré de una forma muy simple cómo trabajar con objetos en JavaScript; como me parece que explicar la definición de objeto será engorroso, simplemente les diré que son un grupo de variables y métodos que trataremos de una forma determinada; sin más dilación, les enseño el primer ejemplo:

```
function Persona(nombre,email)
{
 this.nombre = nombre;
 this.email = email;
 return this;
}

var yo = new Persona('Ferdy','ferpereda@jazzfree.com');
window.alert("Nombre: "+yo.nombre+"\nE-Mail: "+yo.email);
```

Veamos cómo podemos potenciar el buscador de forma que guardemos todos los registros en un array que contenga objetos de un tipo, en lugar de tener tres o más array's:

```
<HTML>
<HEAD>
<TITLE>Buscador optimizado con objetos</TITLE>
<SCRIPT language="JavaScript" type="text/javascript">
<!--
var Registros = new Array();

function Registro ( clave , descripcion , url )
{
 this.clave = clave;
 this.descripcion = descripcion;
 this.url = url;
 return this;
}

Registros[0] = new Registro('javascript','Sitio con mucho sobre JavaScript','http://loquesea.com/');
// Introduce los que quieras ...
```

```
function Busca(str)
{
   if ( str == '' || str == null )
   {
   window.alert('No tengo nada que buscar.');
   document.busqueda.cadena.focus();
   }
              else
               {
resultado = open("" , "msgWindow" ,'status=no,menubar=no,width=400,height=400');
codigo += '<html><head><title>Resultados</title></head><BODY link=blue vlink=blue alink=blue>';
codigo += '<h3><center>De la cadena: <i>'+str+'</i>.</center></h3>';
codigo += '<hr width=50%>';
   for ( var i = 0 ; i <= Clave.length ; i++ )
   {
      if ( str == Registros[i].clave )
      {
      codigo += "<li><a href="+ Registros[i].url +">Url</a><br>";
      codigo += Registros[i].descripcion +"</li>";
      }
   }
codigo += '<form><center>';
codigo += '<input type=button value=Cerrar onClick="window.close(self)">';
codigo += '</center></form></body></HTML>';
resultado.document.write(codigo);
codigo = '';
               }
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM name=busqueda>
 <INPUT TYPE=text name=cadena>  
 <INPUT TYPE=button onClick="Busca(this.form.cadena.value)" value="Buscar">
</FORM>
</BODY>
</HTML>
```

Conozcamos la función split(), esta función se usa de la siguiente forma:

ObjCadena.split(*separador*);

y devuelve un array temporal en el cual cada uno de sus elementos son las subcadenas que se encuentran o desde el principio de la cadena hasta *separador*, o entre *separadores*, o entre *separador* y el final. Ejemplo:

```
var cadena = 'El Programador|El Novato|El Experimentado'.split('|');
window.alert(cadena[0]+'\n'+cadena[1]+'\n'+cadena[2]);
```

Se preguntarán si esto tiene alguna utilidad, pues pensarán que sería más fácil que declarasemos el array desde un principio, pues no, pues el array se quedaría en memoria todo el tiempo, y esto se haría de forma temporal.

Ahora saben lo suficiente JavaScript como para entender la función que los programadores JavaScript usamos en su día cuando los Arrays no estaban implementados:

```
function initArray()
{
 this.length = initArray.arguments.length;
 for (var i = 0 ; i < this.length ; i++)
```

```
 {
 this[i] = initArray.arguments[i];
 }
}
```

No es recomendable usarla, puesto que no tiene mucha funcionalidad.

Por último veremos una subrutina que ordena un Array aleatoriamente, mejor dicho, desordena:

```
for ( var i = 0; i < miArray.length; i++)
{
 var j = Math.floor(Math.random() * miArray.length);
 var temp = miArray[i];
 miArray[i] = miArray[j];
 miArray[j] = temp;
}
```

Pruebe el Script pulsando el botón de abajo:

Les enseñaré ahora a incluir métodos (funciones) a los objetos que creen. Simplemente tiene que darle al elemento del objeto el nombre de una función definida, como es costumbre, les enseñaré un ejemplo:

```
function Incremento()
{
 this.cont++;
 return this;
}

function Contador()
{
 this.cont = 0
 this.Incremento = Incremento;
 return this;
}

var num = new Contador;
num.Incremento();
window.alert(num.cont);
```

Este capítulo me contiene un ejercicio que os teneis que currar, ahí va:

1. Página que contenga un formulario, que valide que todas las entradas estén rellenadas correctamente, y que una vez relleno, abra una ventana en la cual escriba los datos introducidos por el usuario.

Solución

Bueno, les he preparado un pequeño exámen en JavaScript, consta de 15 preguntas, pero en realidad son más de 20, y saldrán en orden aleatorio, el exámen no tine NINGÚN VALOR LEGAL, simplemente es para que ponga a prueba lo que ha aprendido en estos 8 mortales capítulos, además recomiendo revisen los Anexos del tutorial donde se documenta todo lo nencesario para conseguir sacar adelante sus dotes como programador de JavaScript, el exámen está programado en JavaScript, y he de decir que es fácilmente *crackeable* quiero decir que es muy fácil saber cuál será la respuesta correcta simplemente con un editor de textos, pero no les diré como para que no hagan trampas. He de remarcar que el código del Exámen no es mío, quiero decir que lo saque de un buen libro de casa, así que dicho lo dicho, si quieres probar tus dotes como programador JavaScript, estás totalmente invitado a examinarte.

Anexo 1: Operadores.

Bueno, intentaré explicar de una forma sencilla qué son los operadores. Los operadores son comandos que usamos para: modificar, comparar, ... variables; entonces encontramos los **operadores matemáticos**:

+	Suma dos variables, si son cadenas, las concatena.
-	Resta dos variables numéricas
/	Divide dos variables numéricas
*	Multiplica dos variables numéricas
%	Resto de la división entera de dos variables numéricas
++	Incrementa la variable en 1
--	Decrementa la variable en 1

De los operadores anteriores, pondré un ejemplo de cada uno:

```
var y = 1;
var x = 289;
var res = x + y; // res vale 290

var y = 5;
var x = 34;
var res = x - y; // res vale 29

var y = 5;
var x = 4;
var res = x * y; // res vale 20

var y = 25;
var x = 5;
var res = y / x; // res vale 5

var y = 10;
var x = 4;
var res = y % x; // res vale 2

var x = 0;
x++; // x valdrá 1 cuando se ejecute la siguiente sentencia

var x = 45;
++x; // x valdrá 46 en este momento

var y = 34;
y--; // y valdrá 33 cuando se ejecute la siguiente sentencia

var y = 23;
--y; // y valdrá 22 en este momento
```

Pero estos operadores se pueden abreviar de la siguiente forma:

```
var x = 4;
var y = 34;
x += y; // esto es igual que x = x + y; con lo que x valdrá 38
// lo mismo nos ocurre con el resto de los operadores, -= *= /= %=
```

Existen los llamados **operadores de comparación**:

<	Menor que...	!=	Distinto que...
>	Mayor que...	==	Igual que...

>=	Mayor o igual que...	<=	Menor o igual que...

Ejamplos de los anteriores:

```
var x = 45;
var y = 34;
if ( x < y )  // la sentencia es falsa
if ( x > y )  // la sentencia es verdadera
if ( x == y ) // la sentencia es falsa
if ( x != y ) // la sentencia es verdadera
if ( x <= y ) // la sentencia es falsa
if ( x >= y ) // la sentencia es verdadera

// note que se podría usar cualquier sentencia que no fuera if.
```

Otro tipo de operadores son los llamados **operadores bit a bit**:

~	Operador NOT (bit a bit)	|	Operador OR (bit a bit)
^	Operador XOR (bit a bit)	&	Operador AND (bit a bit)
>>>	Unsigned Cambio a la derecha (bit a bit)	!	Operador NOR (bit a bit)
<<	Cambio a la izquierda (bit a bit)	>>	Cambio a la derecha (bit a bit)

Se usan para cifrados de datos, y manipulación de cadenas.

```
var cadena = "Hola a todos";
var clave = "Esto deberia ser secreto";
var cifrado = cadena << clave;
```

Pero no es muy común el uso de cifrado con JavaScript, pues la clave está en claro.

También podemos encontrar los **operadores lógicos**:

===	Operador de identidad
!==	Operador de desidentidad
||	Operador OR (lógico)
&&	Operador AND (lógico)
?:	Operador ternario

Los operadores de Identidad y Desidentidad son como los operadores de Igualdad y Desigualdad, a excepción de que no se realiza conversión de tipos de datos. (no son muy comunes).

```
// ( sentencia ) operador ( sentencia )
var x = 35;
var y = 5842;
var booleano = true;
(( x < y ) && ( booleano )) // sería lo mismo que...
( x < y && booleano )       // cualquier posibilidad es válida
```

Anexo 2: Funciones avanzadas de JavaScript y objeto Math.

En este anexo, me dedicaré a explicar algunas funciones que nos harán la vida más fácil, y nos ahorrarán varios quebraderos de cabeza. Las primeras que explicaré nos valdrán para convertir una cadena de caractéres en un número o comprobar si la cadena es numérica o no:

parseInt(*cadena* , *idx*): Devuelve un número sacado de *Cadena* en base *idx*. Por ejemplo:

```
parseInt("Hola");    // Devuelve NaN
parseInt("12hola");  // Devuelve 12
```

parseFloat(*cadena*): Devuelve un número con coma flotante sacado de *Cadena*. Por ejemplo:

```
parseFloat("Hola");     // Devuelve NaN
parseFloat("1.2hola");  // Devuelve 1.2
```

isNaN(*variable*): Devuelve true si *variable* es NaN, es decir, si no es un número, y false si *variable* es numérica. Por ejemplo:

```
isNaN("Hola");    // Devuelve true
isNaN("12");      // Devuelve false
```

Empezaremos a usar el objeto Math, este objeto, tiene muchas funciones muy potentes, desde la más sencilla función de devolver el valor absoluto de un número, hasta generación de números pseudoaleatorios, pasando por funciones trigonométricas, etc...aquí sólo explicaré unos pocos:

```
//-----------------------------------------------------------------
// Math.abs ---> devuelve el valor absoluto de un número:
Math.abs('-5'); // devuelve 5
Math.abs('5');   // devuelve 5
//-----------------------------------------------------------------
// Math.cos ---> devuelve el coseno de un ángulo
Math.cos('45') // devuelve 0.5253219888177297.
//-----------------------------------------------------------------
// Math.sin ---> devualve el seno de un ángulo
Math.sin('45'); // devuelve 0.8509035245341184
//-----------------------------------------------------------------
// Math.ceil ---> devuelve el número entero mayor o igual a el.
Math.ceil('0.03324'); // devuelve 1
//-----------------------------------------------------------------
// Math.floor ---> devuelve el número entero menor o igual a el.
Math.floor('5.2368'); // devuelve 5
//-----------------------------------------------------------------
// Math.random ---> devuelve un número pseudoaleatorio entre 0 y 1 inclusive.
Math.random() // por ejemplo 0.1438574885700009
//-----------------------------------------------------------------
//Math.pow ---> devuelve el resultado de elevar el primer argumento al segundo
Math.pow('10','3'); // devuelve 1000
//-----------------------------------------------------------------
//Math.sqrt ---> devuelve la raíz cuadrada de un número
Math.sqrt('400'); // devuelve 20
//-----------------------------------------------------------------
```

Ejemplo práctico: Utilización de el objeto Math, para desplegar banners publicitarios aleatoriamente, usando los métodos random y ceil o floor:

```
var banners = new Array();
banners[0] = 'images/banner1.gif';
banners[1] = 'images/banner2.gif';
banners[2] = 'images/banner3.gif';
banners[3] = 'images/banner4.gif';

function Selecciona_Banner()
{
 var num = Math.ceil( Math.random() * banners.length-1 );
 document.write('<a href=\"loquesea\"><img src='+banners[num]+' border=0></a>');
}
```