Índice general

Capítulo 1

Utilizar JavaScript en un documento HTML

Este documento está hecho usando OzTEX. Los programas de **JavaScript** son scripts (literalmente, manuscritos). En general un script es un programa que para ejecutarse no necesita escribirse en código máquina, sino que basta escribirlo en un lenguaje común. El ordenador que tiene que ejecutar el script, debe tener un programa (intérprete) que lee el script y lo traduce línea a línea en intruciones que puede entender el ordenador. En el caso de **JavaScript**, es el navegador web el encargado de entender el script.

Todos los programas **JavaScript** se incluyen dentro de documentos HTML. En HTML, un script se incluye con la etiqueta <script language="JavaScript">.
Después de esto se añade el texto de lo que es el script, formado por una serie de intrucciones. Cuando se acaba de editar el script se cierra con < /script>.

Así un texto en **JavaScript** incluido en un documento HTML debe tener la siguiente apariencia:

```
<script language="JavaScript">
texto del script
</script>
```

JavaScript tiene poca sintaxis. Para marcar el final de una línea debe ponerse un punto y coma (;). Se pueden dejar espacios en blanco y pulsar Intro para que el texto sea más legible, pero las líneas de código no terminan hasta que se pone un punto y coma.

Como ejemplo, copia el siguiente texto en tu editor de HTML (bloc de notas o HTML pad) y guárdalo con el nombre **miprog01.html**:

```
<html>
```

```
<head>
<script language="JavaScript">
function HolaMundo(){alert("!Hola, mundo!");
}
</script>
</head>
<body>
<form>
<input type="button" NAME="Boton" VALUE="Pulsame"
onClick="HolaMundo()">
</form>
</body>
</html>
```

Ahora lee la página que has guardado con tu navegador. Como ves sólo tiene un botón. Si lo pulsas, aparece una ventana con la frase "!Hola, mundo!". Esta frase quizá sea la más famosa entre los programadores, ya que sacar este mensaje en pantalla es el primer ejercicio del libro de programación más leído: "El lenguaje de programación C", de Brian Kernighan y Dennis Ritchie, inventores del lenguaje C y, con Ken Thompson, creadores del sistema operativo Unix.

Ahora vamos a ver qué significa cada uno de los nuevos elementos que tiene el programa anterior:

```
<script language="JavaScript">
</script>
```

Estas dos etiquetas señalan el principio y fin del script. Dentro de estos elementos es donde se puede poner el código **JavaScript**. Estas etiquetas se pueden poner en cualquier lugar del documento, y tantas como se quiera. Se suelen poner en la cabecera para hacer más legible la parte HTML de la página.

```
function HolaMundo(){
alert("!Hola, mundo!");
}
```

ésta es nuestra primera definición de una función en **JavaScript**. Una función es un procedimiento para hacer algo. Para definir una función hay que poner la palabra **function** seguida del nombre que se quiera dar a la función, aquí es HolaMundo(), luego se colocan dos llaves para delimitar el texto de la función, y entre ellas se escribe las intrucciones que forman el procedimiento de la función. Al texto que hay entre las dos llaves se le llama cuerpo de la función.

El cuerpo de una función puede constar de muchas intrucciones, cada una de ellas termina en un punto y coma (;). El navegador, cuando tiene que ejecutar la función, empieza por ejecutar la primera instrucción, luego la segunda, la tercera, y así, una después de otra, hasta que llega a la última llave, con lo cual termina la función.

En el código de la función vemos la línea

```
alert("!Hola, mundo!");
```

que es una llamada al método alert(). Este método es el que se encarga de mostrar en pantalla el mensaje "!Hola, Mundo!". En seguida hablaremos de objetos y métodos, pero antes terminaremos de comentar el ejercicio.

En el código HTML

```
<form>
<input type="button" NAME="Boton" VALUE="Pulsame"
onClick="HolaMundo()">
</form>
```

vemos el parámetro onClick="HolaMundo()". Es un controlador de evento. Cuando el usuario hace clic sobre el botón, entonces se ejecuta la función que está entre las comillas. Siempre que se usa una función, ésta debe haberse definido antes. Por esto, es una buena costumbre meter el código del script en la cabecera de la página.

Las páginas que tienen **JavaScript** se refieren a objetos, eventos, métodos y propiedades. Esto es así porque la programación en **JavaScript** utiliza la técnica de programación conocida como Programación Orientada a Objetos (Object Oriented Programming, OOP).

La programción orientada a objetos es una técnica de programación diseñada para simplificar conceptos de programación complicados. En esencia, la OOP descansa sobre la idea de tener bloques de datos definidos por el usuario o por el sistema, y también tener medios controlados de acceder y manipular esos datos.

La OOP consta de objetos, métodos y propiedades. Un objeto es básicamente una caja negra que guarda alguna información. Puede haber una forma de leer es información, y una forma para escribir o cambiar esa información, y puede haber otras menos obvias de interactuar con esa información. A parte de esa información, se puede acceder directamente a la información que hay en las propiedades. A otra parte de la información de un objeto puede que sólo se pueda acceder a través de un método. Con los métodos no se sabe que se está haciendo, sólo los efectos que desencadenan en el objeto.

Una página web es un objeto. Cualquier tabla, formulario, botón, imagen o enlace también es un objeto. Cada objeto tiene propiedades (información sobre el objeto). Por ejemplo, el color de fondo de un documento (=document background color) es una propiedad del documento que se denomina **document.bgcolor**. Se puede poner el color rojo como color de fondo de la página dándole el valor red.ª esta propiedad, como ya es sabido.

Los objetos están incluidos unos dentro de otros. Para referirse a un objeto que está dentro de otro, hay que poner el nombre del objeto exterior, un punto, y el nombre del objeto interior. Para referirse a una propiedad hay que poner primero el nombre del objeto, un punto, y el nombre de la propiedad. Así, por ejemplo, el contenido (propiedad value) de una caja de texto llamada "password"que está en un formulario llamado "formulario"es:

```
document.formulario.password.value
```

Con los objetos se pueden hacer varias cosas. Las acciones que se realizan con los objetos se llaman métodos.

Algunos métodos actúan en todos los objetos, y otros son específicos de algún objeto. Por ejemplo, se abre un documento nuevo con el método **document.open()**, o se puede escribir "Hola, mundo"en un documento escribiendo **document.write("Hola, mundo")**. Así, **open** y **write** son métodos del objeto **document**.

Los eventos son la forma en que conseguimos que nuestras funciones actúen. El ejemplo más frecuente es el de un botón en cuya definición está el parámetro **onClick="mifuncion()"**

. Cuando ocurre el evento **onClick**, es decir, se hace clic sobre él, entonces se ejecuta la función **mifuncion()**.

Copia el siguiente texto en tu editor y guárdalo con el nombre **miprog02.html**:

```
<html>
<head>
<script language="JavaScript">
function MsgBox(cadena_de_texto){
  alert(cadena_de_texto)}
</script>
</head>
<body>
<form>
<input name="text1" TYPE=Text>
<input name="submit" TYPE=Button VALUE="Muestrame"
```

```
  onClick="MsgBox(form1.text1.value)">
</form>
</body>
</html>
```

La función **MsgBox()** se ha definido con el parámetro çadena-de-texto". Un parámetro es un nombre que hace el papel de las indeterminadas en las funciones matemáticas: está en la definición de la función, pero cuando se quiere calcular la función en un caso concreto, el parámetro y la indeterminada se reemplazan por ese valor concreto. Lee esta página en tu navegador. Te presenta un cuadro de texto donde puedes escribir, y un botón. Cuando pulsas el botón, el texto que has escrito reemplaza el parámetro de la función **MsgBox()**, y se muestra dentro de una ventana de alerta.

El siguiente programa presenta cuatro botones con los que cambiar el color de fondo de la página. Cópialo en tu editor y sálvalo con el nombre de **miprog03.html**, y luego lee la página y observa como funciona:

```
<html>
<head>
<script language="JavaScript">
function cambiacolor(codigo){
   document.bgColor=codigo;
}
</script>
</head>
<body>
<form>
<input type="button NAME="Boton1" VALUE="ROJO"
onClik="cambiacolor('red')">
<input type="button NAME="Boton2" VALUE="VERDE"
onClik="cambiacolor('green')">
<input type="button NAME="Boton3" VALUE="AZUL"
onClik="cambiacolor('blue')">
<input type="button NAME="Boton4" VALUE="BLANCO"
onClik="cambiacolor('white')">
</form>
</body>
</html>
```

Capítulo 2

Tipos de datos y expresiones en JavaScript

2.1. Variables

Si tenemos una cadena de caracteres larga, como "Se llama internautas a quienes son aficionados a usar los navegadores de Internet", que queremos manipular de alguna forma, o que vamos a escribir varias veces, podemos crear una variable, por ejemplo *variable1*, y asignarle esa cadena de caracteres como su valor. Esto se hace escribiendo la siguiente línea en nuestro script:

```
var variable1="Se llama internautas a quienes son aficionados
a usar los navegadores de Internet";
```

Las variables sirven para guardar un dato y hacerlo accesible a las funciones e instrucciones del script. Las variables forman parte del núcleo de cualquier lenguaje de programación.

En **JavaScript** las variables pueden tomar valores de los siguientes tipos:

1. Número: como 2, 34 o 3.14159. Los enteros y los reales son datos del mismo tipo. Los números reales se escriben con un punto separando la parte entera de la parte decimal.

2. Booleano: sólo *true* o *false* (verdadero o falso). Con la función newBoolean() lo que se ponga entre los paréntesis se convierte en *true* o *false*.

3. Cadena de caracteres: Como "Hola.º "Valencia es una ciudad del Mediterráneo". Siempre deben ponerse entre comillas o apóstrofos.

4. El tipo *null*, que representa un valor nulo.

Una misma variable puede tomar un valor de un tipo, y luego, en otra parte del mismo script, tomar un valor de otro tipo. En **JavaScript**, a diferencia de otros lenguajes de programación, no importa qué tipo de valor toma una variable.

Escribe el siguiente código y sálvalo con el nombre **miprog04.html**:

```
<html>
<head></head>
<body>
<center>
<script language="JavaScript">
var a=new Boolean("Falso");
var b=54;
var c=3.14159;
var d="Menudo rollo de clase";
document.write("a="+a+"<br>");
document.write("b="+b+"<br>");
document.write("c="+c+"<br>");
document.write("d="+d+"<br>");
</script>
</center>
</body>
</html>
```

Lee esta página en tu navegador. Observa que esta vez no hay que apretar botones. Sólo se han mostrado algunas cosas en la página, pero no se pueden modificar sin modificar la propia página. Se ha usado el método **document.write()** para lograr escribir en la página. Observa el valor de la varible a.

Pon los siguientes valores entre los paréntesis de la función **new Boolean()** con la que se asigna valor a la variable a, y observa cuáles dan true: null, true, false, 23.

Las cadenas de caracteres pueden tener caracteres especiales que indiquen el formato. Estos caracteres son:

Carácter	**Descripción**
\b	Espacio atrás
\f	Cambio de página
\n	Nueva línea
\r	Retorno de carro
\t	Tabulación

Como ejercicio escribe una página que se llamará **miprog05.html** que tenga un script consistente sólo con el método **document.write()**. Pon en este método una

cadena de caracteres de tu invención, que tenga una cierta longitud (al menos dos líneas del editor).

Intercala los caracteres especiales de la tabla anterior en la cadena que has puesto como argumento del método **document.write()**, y observa el efecto en la presentación leyendo la página en tu navegador.

2.2. Expresiones y operadores

Para manipular variables y valores necesitamos *operadores*. Ejemplos de operadores son +,-,* y =.

Los operadores permiten operar con valores, asignar valores a las variables, comparar valores y variables y escribir condiciones en las instrucciones (que trataremos en el siguiente apartado). Los valores y variables sobre los que actúa un operador se llaman *operandos*. Una combinación de un operador y sus operandos es una *expresión*. Por ejemplo, 3+4 es una expresión, + es el operador, 3 y 4 son los operandos. La propia expresión 3+4 tiene un valor, 7, y por tanto puede formar parte de otra expresión como operando. Al igual que se hace en matemáticas, se deben usar paréntesis para indicar el orden en que actúan los operadores en las expresiones que tienen más de un operador.

Un operador fundamental para programar es el operador *asignación* de un valor a una variable. En **JavaScript** se representa por el signo matemático de igual (=). Las expresiones que forma tienen la siguiente estructura:

```
variable=expresion;
```

Por ejemplo x=x+5; renta=capital*0.05;

Además del operador de asignación, hay operadores aritméticos, lógicos y relacionales.

La siguiente tabla muestra los operadores aritméticos que realizan las operaciones aritméticas habituales, y los operadores aritméticos que combinan una asignación y una operación aritmética:

Operador	**Propósito**	**Operador**	**Propósito**
+	Sumar 2+3 o unir cadenas	+=	Sumar y asignar
-	Restar 3-2	-=	Restar y asignar
*	Multiplicar 3*2	*=	Multiplicar y asignar
/	Dividir (4/2)	/=	Dividir y asignar
%	Módulo o resto de la división	%=	Asignar el módulo o resto
++	Incrementa en 1	- -	Disminuir en 1

Para ver todo lo anterior con ejemplos, escribe el siguiente código en un fichero llamado **miprog06.html**. Luego léelo en el navegador:

```
<html>
<head></head>
<body>
<center>
<script language="JavaScript">
var a=2+3;
var b=2*3;
var c=7/2;
var d=7%2;
document.write("a="+a+"<br>");
document.write("b="+b+"<br>");
document.write("c="+c+"<br>");
document.write("d="+d+"<br>");
</script>
</center>
</body>
</html>
```

Ahora añade al script las siguientes instrucciones y observa el efecto:

```
a+=2;
b*=2;
c/=2;
d++;
document.write("a="+a+"<br>");
document.write("b="+b+"<br>");
document.write("c="+c+"<br>");
document.write("d="+d+"<br>");
```

Observa el uso del operador + para formar el argumento de la función **document.write()** concatenando cadenas de caracteres (añadiéndolas unas después de las otras).

Los operadores lógicos tienen como operandos valores booleanos y devuelven los valores *true* o *false*. La siguiente tabla muestra los operadores lógicos.

Operador	**Significado**	**Resultado**
&&	Y lógico	*true* si los dos operandos son *true*, y *false* si algún operando es *false*
||	O lógico	*true* si algún operando es *true*, y *false* si los dos operandos son *false*
!	No	*false* si el operando es *true*, *true* si el operando es *false*

Los operadores de relación permiten comparar los operandos y devuelven el valor *true* o *false* según el resultado de la prueba. Cuando los operandos son cadenas de caracteres, la comparación de orden se basa en el orden lexicográfico (es decir, el orden de las palabras en el diccionario). La siguiente tabla describe los operadores de relación:

Operador	**Resultado**
==	*true* si los operandos son iguales
!=	*true* si los operandos no son iguales
>	*true* si el izquierdo es mayor que el derecho
>=	*true* si el izquierdo es mayor o igual que el derecho
<	*true* si el izquierdo es menor que el derecho
<=	*true* si el izquierdo es menor o igual que el derecho

Capítulo 3

Instrucciones

Un programa, o script, de **JavaScript** es una lista de instrucciones. Las instrucciones terminan en punto y coma (;). Una instrucción se puede definir en varias líneas de texto, pero no termina hasta que se pone el carácter punto y coma (;). En una línea de texto puede haber varias instrucciones separadas por punto y coma. A las instrucciones también se las llama líneas de código, que es como las introdujimos anteriormente.

El lenguaje **JavaScript** tiene unas pocas palabras reservadas para ser usadas en la escritura del esquema de las instrucciones que pueden hacerse en un programa. Estas palabras no pueden usarse como nombres de variables, ni como nombres de funciones. En los subapartados siguiente veremos cuáles son esas palabras.

Por el esquema de una instrucción entendemos el esquema sintáctico de la instrucción. Cada instrucción tiene su propia sintaxis, que si no se respeta no puede ser entendida por el navegador.

3.1. Declaración de variables

Cuando se quiere usar una variable, primeramente hay que crearla mediante una declaración, según el esquema:

```
var variable=valor;
```

En este esquema se puede suprimir la parte de asignación de valor (=valor). También pueden declararse varias variables en una sola instrucción, según el esquema:

```
var variable1=valor1, variable2=valor2,..;
```

donde las inicializaciones también son opcionales.

3.2. Comentarios

En un script se pueden insertar líneas que no representan instrucciones, sino que sólo sirven para hacer comentarios al código. Estos comentarios explican las partes del script y ayudan a entenderlo. Ya que no son instrucciones el navegador las ignora.

Una línea de texto que comienza con dos barras oblícuas hacia la derecha (//) es un comentario. Varias líneas de texto escritas entre los signos /* y */, también constituyen un comentario. Por ejemplo, son comentarios las siguientes líneas:

```
//Esta es una linea de comentario.
/*Aqui comienza un comentario
que ocupa dos lineas*/
```

3.3. Definición de funciones

Las funciones son conjuntos de instrucciones que pueden ser ejecutados en cualquier lugar de la página.

Para definir una función se usa la palabra **function**, seguida del nombre de la función y luego de sus argumentos, delimitados por paréntesis y separados por comas. Después de los argumentos se colocan las instrucciones que forman el cuerpo de la función, delimitadas por llaves.

El esquema para definir una función es el siguiente:

```
function nombre_de_la_funcion (arg1,arg2,...,argN){instrucciones};
```

Si se quiere que la función produzca un valor que pueda ser asignado a una variable, la última instrucción del cuerpo de la función debe ser de la forma:

```
return expresion
```

donde se usa la palabra reservada **return**, y *expresion* es la expresión cuyo valor queremos que sea el valor producido por la función.

Cuando se quiere usar una función se escribe el nombre de la función y, entre paréntesis, se pone la lista de valores de los argumentos de la función.

Como ejemplo de lo dicho hasta aquí sobre las funciones, copia el siguiente texto en tu editor y guárdalo con el nombre de **miprog07.html** y luego léelo en el navegador.

```
<html>
<head>
<script language="JavaScript">
```

```
function StatusWrite(status){
self.defaultStatus=status
}
</script>
</head>
<body>
<script language="JavaScript">
StatusWrite(">Vas a pasar el rato mirandome?");
</script>
</body>
</html>
```

Esta página tiene dos scripts. El primero está en la cabecera, y consiste en la definición de la función **StatusWrite()**. Esta función tiene un argumento. El segundo script está en el cuerpo de la página, y en él se usa la función definida en el primer script con una cadena de caracteres como valor del argumento.

Al cargar la página, ésta aparece vacía. Lo único que hace la función es escribir el texto que se pone de argumento de la función en la barra de estado del navegador. (**Netscape** requiere estar en modo conectado para que el mensaje se vea.)

Copia el siguiente ejemplo en un archivo denominado **miprog08.html** y luego léelo en el navegador. Fíjate en el uso reservado de la palabra **return** para conseguir que la función tenga un valor.

```
<html>
<head>
<script language="JavaScript">
function cuadrado(numero){
return numero*numero;
}
</script>
</head>
<body>
<script language="JavaScript">
document.write("La funcion cuadrado(5) toma
el valor:",cuadrado(5),".");
</script>
<P>Fin de la demo.
</body>
</html>
```

3.4. Las intrucciones condicionales if e if...else

Una instrucción condicional **if** es un conjunto de instrucciones que se ejecuta si (if) es cierta (tiene el valor *true*) una condición especificada. Tiene la siguiente sintaxis:

```
If(condicion){
Instrucciones a ser ejecutadas
}
```

donde condición es cualquier expresión **JavaScript** que toma el valor *true* o *false*. Las "Instrucciones a ser ejecutadas"puede ser cualquier conjunto de instrucciones **JavaScript**, incluso otras instrucciones condicionales.

La instrucción condicional **if...else** es como **if**, pero tiene un conjunto de instrucciones a ejecutar cuando la condición es falsa. Su sitanxis es:

```
If(condicion){
Instrucciones a ser ejecutadas
}
else{
Instrucciones a ejecutar si condicion es false;
}
```

Si en la parte **if** o en la parte **else** sólo hay que ejecutar una instrucción, entonces se pueden omitir las dos llaves correspondientes a esa parte.

Haz el siguiente ejercicio, guardándolo con el nombre **miprog09.html**. La función **checkData()** cambia a una página con un mensaje de felicitación si el número de caracteres en el objeto de texto **password** tiene más de siete caracteres, y si no es así, si tiene siete o menos, presenta una ventana de alerta:

```
<html>
<head>
<script language="JavaScript">
function checkData(){
if (document.form1.password.value.length>=08){
    document.write("!Muy bien, repita.","<br>");
    }else{
      alert("Escriba al menos 8 caracteres."+
      document.form1.password.value+" no es valido.");
      return false;
```

```
      }
}
</script>
</head>
<body>
<P>Escriba una palabra de mas de 7 letras<br>
<form NAME="form1">
<input TYPE="text" NAME="password" Value="">
<input TYPE="button" NAME="Boton" Value="Terminar"
onClick="checkData()">
</form>
</body>
</html>
```

3.5. El bucle for

Los bucles son instrucciones que permiten ejecutar reiteradamente un conjunto de mandatos.

El bucke **for** permite recorrer un conjunto de índices sucesivos, realizando cada vez un conjunto de mandatos, hasta terminar de recorrer el conjunto de índices.

Los bucles **for** tienen la sintaxis:

```
for (expresion_inicial;condicion_final;actualizacion){
   conjunto de instrucciones
}
```

Donde *expresion-inicial* define e inicializa una variable llamada contador que recorre el conjunto de índices. La *condicion-final* es una expresión que vale *true* mientras la variable contador toma un valor en el conjunto de índices, y que vale *false* cuando el contador sale del conjunto de índices. La *actualizacion* es una expresión que cambia el valor de la variable contador después de ejecutar el *conjunto de instrucciones* del bucle.

Por ejemplo

```
for (var i=0;i<9;i++){
     x+=1;
     una_funcion(x);
}
```

es un bucle donde se declara e inicializa a cero la variable **i**, se ejecutan dos instrucciones y se aumenta i en uno, y se sigue haciendo esto hasta que i deja de valer menos de nueve. El conjunto de índices del bucle es, por tanto el conjunto de números (0,1,2,...7,8), y el conjunto de instrucciones se ejecuta nueve veces.

Escribe el siguiente código en un archivo de nombre **miprog10.html** y léelo en el navegador

```
<html>
<head>
<script language="JavaScript">
function cuantas(selectObjeto){
var numeroSeleccionado=0;
for (var i=0;i<selectObjeto.options.length;i++){
    if (selectObjeto.options[i].selected==true)
    numeroSeleccionado++;
    }
    return numeroSeleccionado;
    }
</script>
</head>
<body>
<form name="selectForm">
<P><B>Elija las frutas que le gustan, y luego pulse el boton:</B>
<BR><select name="frutas" MULTIPLE>
<option selected>Naranjas
<option>Peras
<option>Manzanas
<option>Fresas
<option>Platanos
<option>Melon
</select>
<P><input type="button" VALUE=">Cuantas ha seleccionado?"
onClick="alert('Numero de opciones seleccionadas:'+
cuantas(document.selectForm.frutas))">
</form>
</body>
</html>
```

Este código presenta una lista desplegable de opciones donde se pueden hacer varias

selecciones, y un botón. Al hacer clic en el botón, se presenta una ventana de alerta con un mensaje donde dice cuántas opciones se han seleccionado.

3.6. El bucle while

El bucle **while** tiene la siguiente sintaxis:

```
while(condicion){
   conjunto de instrucciones
   }
```

Permite ejecutar repetidamente el *conjunto de instrucciones* mientras la expresión *condición* tenga el valor *true*. El siguiente código presenta una sucesión de números con un bucle while (guárdalo como **miprog11.html**)

```
<html>
<head>
<script language="JavaScript">
var i=0;
while(i<5){
   i+=1;
   document.write(i+" ");
};
</script>
</body>
</html>
```

Como ejercicio, cambia el bucle anterior para que la página presente los múltiplos de 3 menores que 100.

Ahora cambia el bucle para que presente los mismos números, pero en orden inverso.

Capítulo 4

Eventos y objetos

4.1. Eventos

Los eventos son el resultado de una acción del usuario. Por ejemplo, hacer clic sobre un botón o seleccionar un campo de un formulario son eventos. El interés de **JavaScript** radica en poder gestionar los eventos en el ordenador del usuario que lee la página, en vez de hacerlo en el ordenador del servidor.

La gestión de los eventos se efectúa asociándolos a los scripts cuando se produce un evento. Para asociar un script a un evento, se introducen parámetros *manejadores de evento* en las instrucciones HTML de la página. El esquema sintáctico para poner un parámetro de evento es el siguiente:

```
<ETIQUETA...onEvento="Codigo JavaScript"...>
```

donde ETIQUETA representa una directiva HTML que define el elemento de la página que puede tener el evento, y **on**Evento es un nombre compuesto que identifica el evento, siendo la parte Evento del nombre una palabra que describe la acción del usuario. En cualquier directiva HTML se puede colocar más de un parámetro manejador de evento, además de los parámetros necesarios ya habituales, que no son manejadores de evento.

Por ejemplo, se podrá ejecutar una función **JavaScript** ya definida, que tiene el nombre *calcula()*, para que procese el contenido de los campos de un formulario de nombre *form* cuando se pulsa un botón, si ponemos una línea como la siguiente en la definición de dicho formulario:

```
<input type="button" value="Calcular" onClick="calcula(this.form)">
```

La siguiente tabla muestra los manejadores de evento más frecuentes, su significado y las dirctivas HTML a las que se aplican:

onEvento	**Si el usuario...**	**Etiquetas HTML**
onClick	Hacer clic en formulario o enlace	A, FORM
onChange	Cambiar contenido	INPUT, SELECT
onFocus	Activa un campo de edición	INPUT
onLoad	Carga la página en el navegador	BODY
onMouseOver	Mover el ratón sobre un enlace	A
onSelect	Selecciona un campo de texto	INPUT
onSubmit	Envía un formulario	FORM
onUnload	Descarga la página del navegador	BODY

Escribe la siguiente página y dale el nombre **miprog12.html**

```
<html>
<head>
<script language="JavaScript">
function DisplayMsg(msg,evento){
   alert(msg+"\n"+"Evento:"+evento);
   }
</script>
<body onLoad="DisplayMsg('Se ha cargado la pagina','onLoad')"
onUnload="DisplayMsg('Se cierra la pagina','onUnload')">
<form method="POST">
<table>
<tr><td>Pulse sobre este boton
<td><input type="button"
onClick="DisplayMsg('Ha pulsado un boton','onClick')">
<tr><td>Escriba un nombre
<td><input type="text" size=15
onChange="DisplayMsg('Ha terminado de escribir el nombre '
+this.value,'onChange')">
<tr><td>Seleccione una opcion
<td><select Name=Seleccion
onChange="DisplayMsg('Ha cambiado de opcion;',
'onChange')">
<option value=Option1>Seleccion1
<option value=Option2 SELECTED>Seleccion2
<option value=Option2>Seleccion3
</select>
```

```
</table
</form>
</body>
</html>
```

Copia el siguiente código en un archivo llamado **miprog13.html** y cárgalo en el navegador. Al pasar el ratón por el enlace se invoca una función **JavaScript**

```
<html>
<head>
<script language="JavaScript">
function HolaMundo(){
  alert("!Hola, mundo!");
  }
</script>
</head>
<body>
<a href=""onMouseOver="HolaMundo()">Saluda</a>
</body>
</html>
```

El siguiente código permite escribir en la línea de estado del navegador al pasar por un enlace, y borrar lo escrito al cabo de 10 segundos. Cópialo en un archivo llamado **miprog14.html** y cárgalo en el navegador.

```
<html>
<head>
<script language="JavaScript">
function moveover(txt){
   window.status=txt;
   setTimeout("erase()",10000);
   }
   function erase(){
   window.status="";
   }
   </script>
</head>
<body>
<a href="" onMouseOver="moveover('!Este mensaje se
autodestruira en 10 segundos!');return true;">
```

```
Pasa por aqui</a>
</body>
</html>
```

Al pasar el ratón por el enlace se invoca la función **moreover()** que escribe en la línea de estado del navegador y luego borra el escrito. Para escribir lo que hace es cambiar el valor de la propiedad **status** del objeto **window** (que es la ventana del navegador) a una cadena de caracteres determinada, y para borrar, cambia la misma propiedad a la cadena nula con la función **erase()**. Esta función tiene por primer argumento **setTimeout**, que está predefinida en **JavaScript**. El segundo argumento de la función es **setTimeout** (que es 10.000 milisegundos, es decir, 10 segundos) es lo que esa función tarda en ejecutar la función que figura como su primer argumento.

4.2. Objetos

Los objetos tienen asociados propiedades y métodos. Las propiedades son variables **JavaScript**. Los métodos son funciones asociadas a los objetos. **JavaScript** tiene muchos objetos definidos para poder actuar sobre las páginas que se muestran en el navegador, y sobre las ventanas de éste.

A cada objeto se le asocian variables denominadas propiedades (que a su vez pueden ser también objetos). Puede accederse a ellas utilizando el siguiente esquema sintáctico:

```
Nombre_del_objeto.Nombre_de_la_propiedad
```

Los métodos de un objeto son funciones que se le asocian. Puede accederse a un método utilizando la sintaxis:

```
Nombre_del_objeto.Nombre_del_metodo
```

Cada elemento de una página web definido por una etiqueta HTML es un objeto desde el punto de vista de **JavaScript**, y se puede hacer referencia a él gracias al valor del parámetro NAME de dicha etiqueta, que actúa como nombre del objeto. Pero en el código que se escribe dentro de la propia etiqueta es más cómodo usar, en vez del nombre del objeto, la palabra reservada **this**, que siempre permite referenciar el objeto actual.

Por ejemplo, supongamos que tenemos la función:

```
function EdadValida(objeto, valor){
```

```
If(objeto,value<valor)
alert("No tienes suficiente edad");
}
```

con la que queremos comprobar si los lectores de nuestra página son mayores de edad. Se podrá entonces llamar a la función **EdadValida** dentro de un formulario con la palabra reservada this y un controlador de evento para saber cuándo se cambia el valor del dato, como en el siguiente ejemplo:

```
<form>
<center><input type="text" name="edad" size=3
onChange="EdadValida(this,18)"></center>
</form>
```

Haz un archivo html donde esté el código del ejemplo anterior. Llámalo **miprog15.html** y comprueba que funciona.

4.2.1. El objeto Math y las estructuras matemáticas

El objeto **Math** posee propiedades y métodos asociados que corresponden a las constantes y funciones matemáticas. Por ejemplo, la propiedad **PI** del objeto **Math** tiene el valor de π, y se accede a ella mediante **Math.PI**.

De manera parecida, las funciones matemáticas usuales son métodos del objeto **Math**. Así, por ejmplo, la función coseno es **Math.cos()**. Haz un archivo con el nombre de **miprog16.html** y copia el siguiente texto donde se usa el método correspondiente a la función trigonométrica seno, **Math.sin()**.

```
<html>
<head></head>
<body>
<center>
<table>
<tr><td>Escribe un numero:
<tr><td><form name="form1">
<center><input type="text" name="numero" size=15>
<input type="button" value="Funcion seno"
  onClick="alert('El seno de '+form1.numero.value+'
         es:\n'+Math.sin(form1.numero.value))">
   </center>
   </form>
```

```
</table>
</center>
</body>
</html>
```

Añade botones para las funciones coseno, tangente (**Math.tan()**), y raíz cuadrada (**Math.sqrt()**).

4.2.2. El objeto Date: la representación de las fechas

El lenguaje **JavaScript** no posee variables de tipo fecha, pero el objeto predefinido **Date** permite la manipulación de datos que representen fechas. Este objeto no tiene propiedades, pero posee métodos que permiten la obtención (métodos **get**) y actualización (métodos **set**) de fechas.

Se utiliza el siguiente esquema para crear un objeto de tipo **Date**:

```
variable=new Date(parametros)
```

Los prámetros pueden ser cualquiera de los siguientes:

1. Ninguno: se cre la hora y la fecha actual. Por ejemplo: today=new Date().

2. Una cadena de caracteres que representa una fecha en la siguiente forma:"Mes día, año hora:minutos:segundos". Por ejemplo Xmas95=new Date("December 25, 2003 13:30:00"). Si se omite hora, minutos o segundos, el valor será puesto a cero.

3. Un conjunto de valores enteros para año, més y día. Por ejemplo, Xmas65=new Date(95,11,25).

4. Un conjunto de valores enteros para año, mes, día, hora, minutos y segundos. Por ejemplo, Xmas95=new Date(95,11,25, 9,30,0).

Copia la siguiente página (**miprog17.html**) que muestra la hora y la fecha en que se lee la página:

```
<html>
<head></head>
<body>
<script language="JavaScript">
today=new Date()
document.write("La hora actual es:
```

```
  ",today.getHours(),":",today.getMinutes()+"<br>")
document.write("La fecha es:",today.getDate(),"del",
   today.getMonth()+1," de ", today.getYear());
</script>
</body>
</html>
```

4.2.3. El objeto string y las cadenas de caracteres

En cada asignación de una cadena de caracteres a una variable o a una propiedad, **JavaScript** crea un objeto de tipo **string**. Por ejmplo, mistring=”!Hola, mundo”, crea un objeto llamado **mistring**, de tipo **string**.

Un objeto de tipo **string** posee una sola propiedad, **length**, que contiene el tamaño de la cadena de caracteres representada por dicho objeto.

El método **substring()** extrae los caracteres comprendidos entre los índices pasados cmo argumentos. Por ejemplo, ”El nuevo año”, **substring(0,8)** devuelve ”El nuevo”.

Los métodos **toLowerCase()** y **toUperCase()** transforman la cadena de caracteres en minúsculas y mayúsculas respectivamente.

Copia el siguiente texto en un archivo llamado **miprog18.html** y cárgalo en tu navegador. La función **scroll()** que se define en él se encarga de escribir una cadena de texto en la barra de estado del navegador, borrarla, volverla a escribir en otra posición, etc., logrando así que el texto se mueva:

```
<html>
<head>
<script language="JavaScript">
var scrtxt="Este mensaje se mueve...";
var lentxt=scrtxt.length;
var width=100;
var pos=1-width;
function scroll(){
  pos++;
  var scroller="";
  if (pos==lentxt){
     pos=1-width;
     }
  if (pos<0){
```

```
    for (var i=1;i<=Math.abs(pos);i++){
      scroller=scroller+" ";}
      scroller=scroller+scrtxt.substring(0,width-i+1);
      }
  else{
    scroller =scroller+scrtxt.substring(pos,width+pos);
    }
    window.status=scroller;
    setTimeout("scroll()",150);
    }
    </script>
    </head>
    <body onLoad="scroll();return true;">
    Mira el borde inferior de la ventana
    </body>
    </html>
```

Observa el uso de la propiedad **length** y el método **substring()** con la variable de texto **scrtxt**.

4.2.4. El objeto document

El objeto **document** contiene la información respecto al documento actual. Se define por la directiva BODY. El contenido del documento se delimita por las etiquetas <BODY>...</BODY>, y se puede acceder por las propiedades del objeto **document**.

Los métodos del objeto **document** son los siguientes:

1. **clear()** Borra el contenido de la ventana
2. **write()** Permite mostrar en un documento un número cualquiera de cadenas de caracteres o instrucciones (que se evalúan antes de mostrarse).
3. **writeln()** Es como **write()**, pero inserta un retorno de carro.
4. **open()** Abre un búfer (una zona de memoria) para recoger los resultados de los métodos write y writeln.
5. **close()** Cierra los búferes abiertos mediante el método **document.open()**.

Como ejemplo vamos a ver como poner la fecha de la última modificación de la página. Dale el nombre **miprog19.html**

```
<html>
<head></head>
<body>
Ultima modificacion
<script language="JavaScript">
document.write(" "+document.lastModified+"<br>");
</script>
</body>
</html>
```

4.2.5. El objeto window

Este objeto representa la ventana en la cual se presenta la página, y por tanto contiene a todos los demás objetos. Por eso, sus métodos se usan sin referencia al propio objeto.

El método **alert()**, que hemos usado en varios ejercicios, crea una ventana con un mensaje de alerta. El método **open()**, que veremos en el siguiente ejercicio, crea una nueva ventana del navegador.

El método **confirm("mensaje")** crea un cuadro de confirmación (Aceptar/Cancelar), y el método **prompt("mensaje",predeterminado)** abre un cuadro de diálogo que presenta la cadena de caracteres "mensajeτ un campo de edición, con el valor inicial de *predeterminado*.

```
<html>
<head>
<script language="JavaScript">
function AbreVentana(){
  msg=open("","DisplayWindow","toolbar=no,directories=no,
    menubar=no");
  msg.document.write("<HEAD><TITLE>Ventana creada con
    JavaScript</TITLE></HEAD>");
  msg.document.write("<CENTER><h1><B>Aqui empieza el
  texto de la pagina</B></h1></CENTER>");
  }
  </script>
  </head>
  <body>
  <form>
```

```
<input type="button" NAME="Boton" VALUE="Pulsame"
   onClick="AbreVentana()">
</form>
</body>
</html>
```

Como ejercicio, modifica las opciones del método open() y cambia el texto que escribe la función en la nueva ventana.