



LENGUAJE JAVASCRIPT



ÍNDICE

LENGUAJE JAVACRIPT

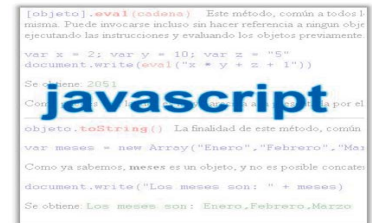
1. Los lenguajes de Script	3
2. Sintaxis de JavaScript	6
3. Funciones del lenguaje	14
4. Eventos	18
5. Objetos	20
6. Resumen	44



1. Los lenguajes de Script

Objetivos

1. Conocer la sintaxis de JavaScript
2. Identificar funciones y eventos de JavaScript
3. Desarrollar sentencias de control y objetos del lenguaje JavaScript



Los lenguajes Script

La combinación del lenguaje XHTML y el estándar de estilo CSS constituye una potente herramienta a la hora de presentar datos en la Web, ofreciendo numerosas posibilidades en el formato de los mismos.



Sin embargo, las páginas creadas con estos estándares, debido a su naturaleza estática, ofrecen poco nivel de interacción con el usuario, resultando a veces poco amigable su utilización.

En este contexto, surgieron los lenguajes de script como medio para añadir dinamismo a una página Web.

La utilización de script en páginas Web es la base de una técnica de programación muy popular en la actualidad llamada AJAX, mediante la que se consigue mejorar de forma espectacular la experiencia de usuario con las páginas Web.

Los lenguajes de script permiten incluir cierta lógica de programación dentro de la página Web, capaz de ser ejecutada por el navegador durante la carga de la misma o como respuesta a alguna acción del usuario. Estas instrucciones, que son capaces incluso de modificar dinámicamente el aspecto de las páginas, proporcionan así cierto grado de interacción con el usuario.



Características del lenguaje Script

Hay dos características de los lenguajes de script que los diferencian de los lenguajes de programación estándar:

- Poseen un reducido conjunto de instrucciones. Esto permite que los intérpretes de estos lenguajes incorporados en los navegadores Web puedan ser muy ligeros.
- Son lenguajes interpretados. El código de script no tiene que ser traducido para su posterior ejecución, el navegador (el intérprete que lleva incorporado) traduce las instrucciones en el momento de su ejecución.

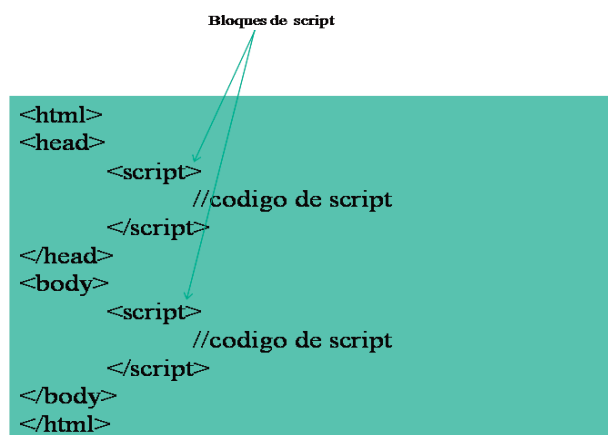
Aunque existen distintos lenguajes de Script, el más popular es sin duda JavaScript, siendo soportado por prácticamente todo los navegadores.

Analizaremos las características sintácticas de este lenguaje y los principales objetos que puede manejar.

Inclusión de un Script en un documento

Como se ha indicado, las instrucciones de script son incluidas directamente dentro de la página Web, aunque se separan del resto del contenido del documento (código XHTML/CSS) utilizando la etiqueta `<script>` de HTML.

Todo bloque de script debe estar delimitado por la etiqueta `<script>` y `</script>`



Como se puede ver en la imagen de la figura, los bloques de script pueden ser incluidos tanto dentro de la cabecera del documento como en el cuerpo. Habitualmente, la cabecera suele reservarse para la definición de scripts con funciones de respuesta a eventos.

LENGUAJE JAVASCRIPT

Fundamentos de JavaScript

Según hemos indicado, JavaScript es el lenguaje de script universalmente utilizado. Su reducido juego de instrucciones y modelo de objetos soportado nos ofrecen infinitas posibilidades a la hora de crear páginas Web dinámicas.

Fundamentalmente, se incluirá código JavaScript en una página Web para:

validación de datos

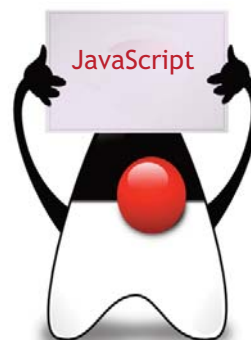
Antes de enviar al servidor los datos recogidos por un formulario, es necesario realizar una serie de comprobaciones básicas, como verificar que se han completado ciertos campos, que los valores introducidos son de un determinado tipo, etc. Todas estas operaciones deben ser realizadas desde la capa cliente (navegador)

Aplicaciones AJAX

AJAX es una técnica de programación que se basa en el intercambio de datos entre la página Web y el servidor de forma asíncrona. La implementación de estas operaciones es realizada mediante JavaScript.

Modificación dinámica de páginas

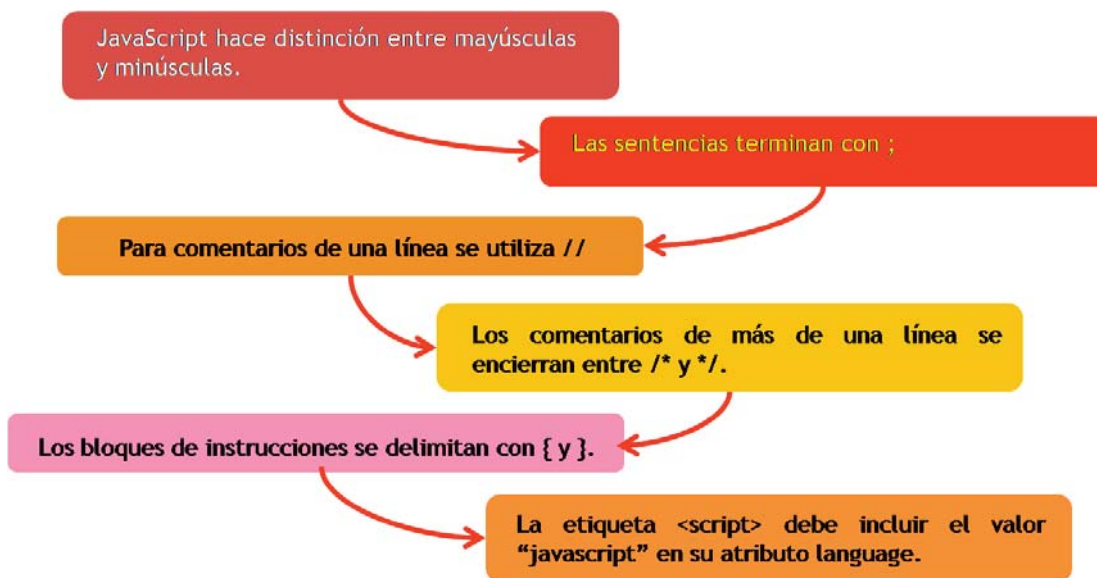
De cara a hacer más atractivo su uso por parte del usuario, ciertas páginas reaccionan con un cambio de aspecto o modificación de su contenido ante ciertas acciones del usuario, como pasar el ratón por encima de algún componente.



2. Sintaxis de JavaScript

Sintaxis básica

Antes de analizar los elementos sintácticos de JavaScript, tengamos en cuenta una serie de consideraciones básicas sobre el lenguaje.



Tipos de datos y variables

Una de las características de todos los lenguajes de script es que son débilmente tipados, lo que significa que el juego de datos que manejan es bastante reducido.

En el caso concreto de JavaScript, se distinguen cuatro tipos de datos:



Texto

El tipo texto representa una cadena de caracteres delimitada por comillas dobles (“ y ”).

Numérico

Incluye todo tipo de número, tanto enteros como decimales.

Boolean

Representa valores lógicos (true y false).

Objeto

En JavaScript se manejan diferentes tipos de objetos que son tratados como un dato más.

Los datos en un programa JavaScript son manejados a través de variables. Dichas variables, aunque no es obligatorio hacerlo, deben ser declaradas antes de ser utilizadas.

La declaración de una variable JavaScript se realiza a través de la palabra reservada var.

```
var nombre_variable;
```

Como vemos, en la declaración de una variable JavaScript no se puede indicar el tipo de datos que va a almacenar, lo que significa que una variable puede contener cualquier tipo de dato.

De cara a asignar una valor a una variable se utilizará el signo “=”

```
var mivariable;  
mivariable = 100; //asignación de un valor numérico
```

Si una variable debe contener un valor inicial, puede establecerse éste en la misma línea de declaración:

```
var p = “hola”; //variable inicializada con un texto
```

Si se utiliza una variable en alguna operación del programa sin haber establecido un valor inicial, lo normal es que se produzca un error de ejecución, puesto que toda variable no inicializada tiene asignado por defecto el valor undefined. Por ejemplo, la siguiente instrucción provocaría un error, puesto que no se puede realizar una operación aritmética sin haber asignado explícitamente un valor a la variable:

```
var k;  
k++; //error
```

En cuanto al ámbito de una variable, es decir, desde dónde puede accederse a ella, puede ser de dos tipos:

Global

La variable se declara fuera de cualquier función, siendo accesible desde cualquier línea de código que exista en la página.

Local

La variable se declara dentro de un bloque (función o instrucción de control), sólo es accesible en el interior de ese bloque.

Hot pot botón ejemplo, aparece lo que se muestra en la siguiente pantalla.

Ejemplo

Este ejemplo nos muestra una variable global y dos locales, una con ámbito de función y otra accesible únicamente dentro de la instrucción for.

```
<script language="javascript">
var m=10;
function ejemplo(){
  var p=10;
  for(var i=0;i<p;i++){
    m+=i;
  }
}
// ->
</script>
```

Variables global

Variables locales

Operadores

Los operadores permiten realizar operaciones con los datos dentro de un programa. JavaScript cuenta con los cinco tipos de operadores indicados en la siguiente tabla.

Tipo	Operador
Aritméticos	+, -, *, /, %, ++, --
Lógicos	&&, , !
Comparación	==, !=, >, <, >=, <=
Condicionales	(condicion) ? valor1 : valor2
Asignación	=, +=, -=, *=, /=

Ejemplo

Por ejemplo, el siguiente bloque de instrucciones incrementa en una unidad el contenido de una variable y, al cargar la página el navegador muestra su valor en un cuadro de diálogo (utilizamos la función `alert()` que estudiaremos más adelante):

```
<html>
  <head><title>Primera página</title></head>
  <body>
    <script language = "javascript">
      var num = 5;
      num++;
      alert(num); //muestra el valor 6
    </script>
  </body>
</html>
```

Además de los anteriores, JavaScript dispone de los siguientes operadores especiales:

new

Se utiliza para crear objetos a partir de su clase.

typeof

Permite conocer si el contenido de una variable es de un determinado tipo.

Sentencias de control

Un programa está formado por un conjunto de instrucciones que se ejecutan una tras otra con un orden preestablecido, pero raras veces se sigue una secuencia lineal de ejecución.

Existen unas instrucciones que nos permiten alterar el flujo de ejecución de un programa dependiendo de que se cumplan o no ciertas condiciones.

A este tipo de instrucciones se les denomina sentencias de control de flujo y pueden ser de dos tipos:

Alternativas

En función de una determinada condición o expresión encamina el flujo de ejecución del programa por uno u otro camino.

Repetitivas o bucles.

Como su nombre indica, permiten repetir la ejecución de un conjunto de instrucciones un número determinado de veces.

Sentencias de control: if y switch

A continuación, revisaremos el juego de instrucciones de control proporcionado por JavaScript. En esta pantalla veremos la instrucción if y la sentencia switch, y en las próximas estudiaremos la sentencia for y while.

If

La instrucción if comprueba una condición, si ésta es cierta (true), se ejecutan las sentencias que aparecen entre llaves (las llaves no son obligatorias si sólo hay una sentencia), si la condición no se cumple se ejecutan las sentencias indicadas en else, cuya utilización es opcional.

Tipo	Formato	Ejemplo
if	<pre>if(condicion){ Instrucciones } else { Instrucciones }</pre>	<pre>if(c==5){ alert ("Número correcto"); } else{ alert ("Incorrecto"); }</pre>

switch

El valor de la expresión se compara con cada una de las constantes de la sentencia case, si coincide alguno, se ejecuta el código indicado a continuación, finalizando la ejecución al encontrar la sentencia break.

En caso de que el resultado de la expresión no coincida con ningún case se ejecutará el bloque default (opcional).

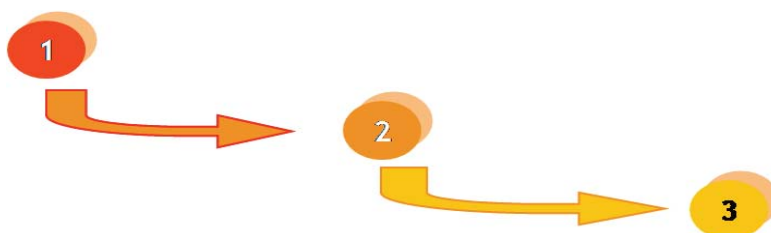
Tipo	Formato	Ejemplo
switch	<pre>switch (expresión){ case valor1: sentencia(s); break; case valor2: sentencia(s); break; [default:] sentencia(s); }</pre>	<pre>switch(tipo) { case "planta": alert("Eres un vegetal"); break; case "animal": alert("Eres del reino Animal"); break; default: alert("Especie Desconocida"); break; }</pre>

Sentencias de control: for

La sentencia for se trata de una sentencia de tipo repetitivo que permite ejecutar instrucciones un número definido de veces. Este número está determinado por la propia instrucción.

Tipo	Formato	Ejemplo
for	<pre>for(inicialización;comparación;incremento) { Instrucciones }</pre>	<pre>for(var i=0;i<10;i++) { alert(i); }</pre>

Los pasos que sigue la ejecución de un bucle for son los siguientes:



1. Se ejecuta la instrucción de inicialización, que como su nombre indica, inicia todos los valores necesarios para el control del bucle.
2. Se realiza la comparación, si la condición es cierta se ejecutan las sentencias del bucle, en caso contrario se sale del bucle y se realiza la primera instrucción que aparezca detrás del bucle.
3. Se ejecuta incremento y se vuelve al paso 2, a realizar otra vez la comparación.

Sentencia while

Al igual que for, while es una sentencia de tipo repetitivo, sólo que el bloque de instrucciones se ejecuta mientras la condición definida sea verdadera.

Tipo	Formato	Ejemplo
while	<pre>while(condicion) { //Instrucciones }</pre>	<pre>while(p!=0) { p--; }</pre>

Al ejecutarse por vez primera la condición, si es cierta se ejecuta el bloque de instrucciones, volviéndose después a comprobar la condición. Cuando la condición es falsa, deja de ejecutarse el bloque de instrucciones.

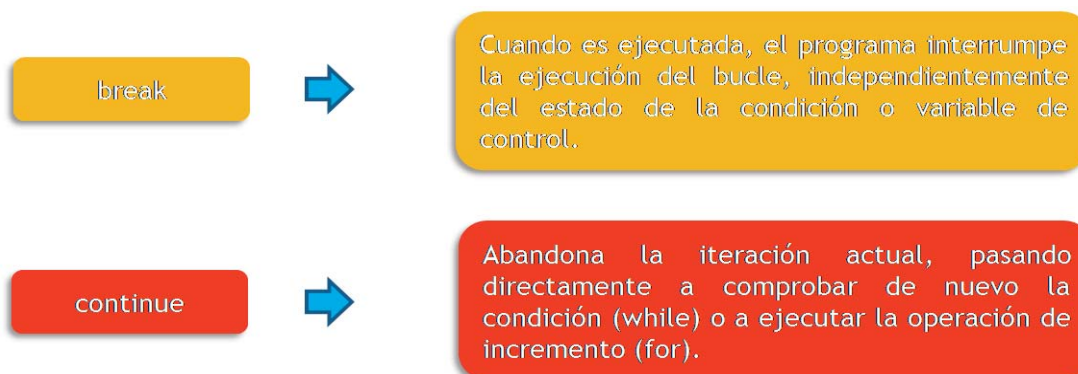
Existe una variante de esta instrucción que es do...while, donde primero se ejecutan las instrucciones y después se comprueba la condición. En este caso, está garantizada la ejecución del bloque de instrucciones por lo menos una vez.

Tipo	Formato	Ejemplo
do while	<pre>do { Instrucciones }while(condicion)</pre>	<pre>do { p--; }while(p!=0)</pre>

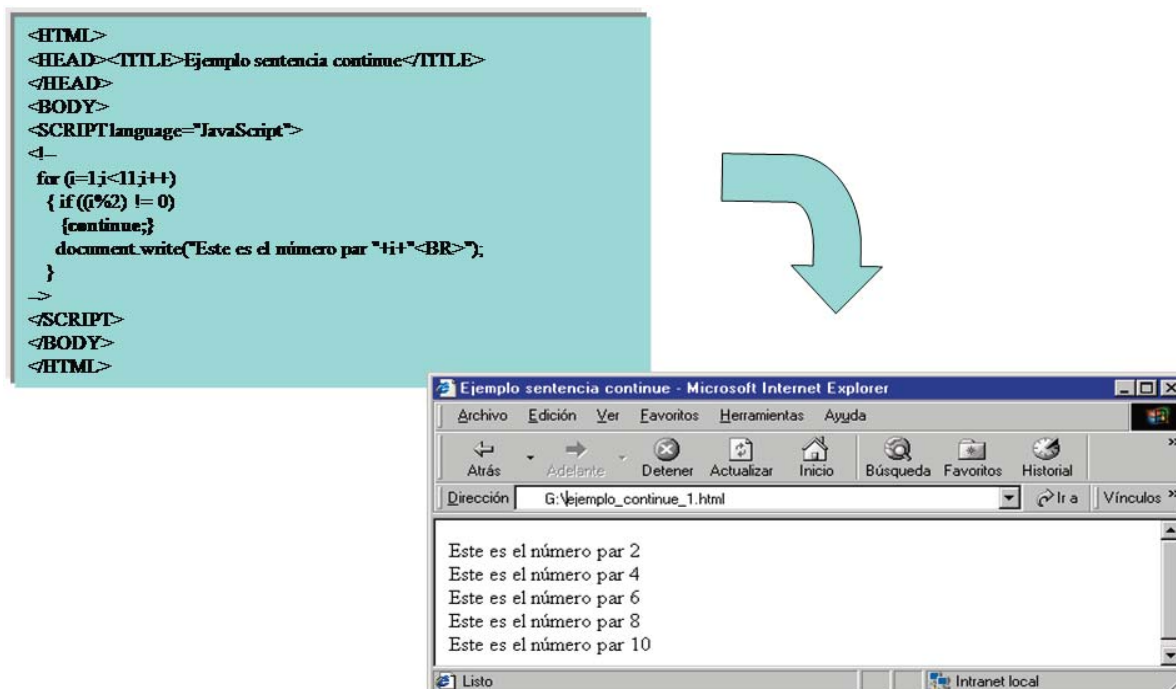


Sentencias de interrupción de un bucle

Existen dos instrucciones que permiten abandonar la ejecución de un bucle for y while. Éstas son:



Veamos un ejemplo de utilización de la sentencia continue. En él, se emplea dicha sentencia para permitir la ejecución de la instrucción de escritura, únicamente cuando la variable de control es par.



The example shows the source code for a file named 'Ejemplo sentencia continue' and its execution result in Microsoft Internet Explorer.

Source Code:

```
<HTML>
<HEAD><TITLE>Ejemplo sentencia continue</TITLE>
</HEAD>
<BODY>
<SCRIPT language="JavaScript">
<!--
for (i=1;i<11;i++)
{ if ((i%2) != 0)
{ continue; }
document.write("Este es el número par "+i+"<BR>");
}
-->
</SCRIPT>
</BODY>
</HTML>
```

Browser Output:

```
Este es el número par 2
Este es el número par 4
Este es el número par 6
Este es el número par 8
Este es el número par 10
```

3. Funciones del lenguaje

Funciones del lenguaje

JavaScript incluye un juego de funciones predefinidas que pueden utilizarse en cualquier parte del código para realizar tareas habituales en un programa.

Para utilizar estas funciones, únicamente hay que invocarlas pasándole los parámetros adecuados. En la siguiente tabla aparece algunas de las funciones más utilizadas de JavaScript.

Nombre	Acción
<code>eval (cadena)</code>	Ejecuta la expresión o sentencia contenida en la cadena que recibe como parámetro.
<code>escape (carácter)</code>	Devuelve el código ascii correspondiente al carácter.
<code>unescape (cadena)</code>	Devuelve el carácter cuya codificación incide en la cadena.
<code>parseInt (cadena)</code>	Convierte la cadena de caracteres numéricos a un entero.
<code>parseFloat (cadena)</code>	Convierte la cadena de caracteres numéricos a un real.
<code>isNaN (valor)</code>	Devuelve true si el valor evaluado es non numérico.

Existen otro tipo de funciones especiales de JavaScript cuya misión es la de generar cuadros de diálogo para interaccionar con el usuario de la página.

Alert (mensaje)

Genera un cuadro de diálogo que muestra un mensaje de aviso suministrado como parámetro.

Confirm (pregunta)

Genera un cuadro de diálogo con un texto que consiste en una pregunta al usuario y dos botones que cierran la ventana, “Aceptar”, cuya pulsación devuelve el valor true y “Cancelar”, cuya pulsación devuelve false.

Prompt (mensaje)

Genera un cuadro de diálogo con un mensaje que solicita un dato al usuario. La pulsación de “Aceptar” cierra la ventana y devuelve el dato introducido.



LENGUAJE JAVASCRIPT

```
var result;
if(!isNaN(texto1.value))
    result=2*parseFloat(texto1.value);
```

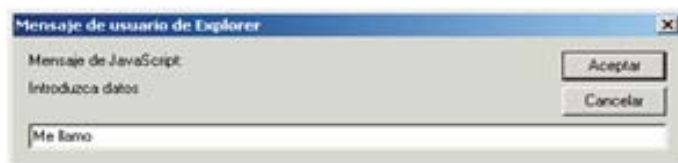
Opera con el contenido de una caja de texto si es numérico

```
var cad=test1.value;
//almacena el código ASCII del espacio como una cadena %20
var esp=escape(" ");
//obtiene el caracter espacio a partir de su código
var res="Buenos"+unescape(esp)+" días";
```

Obtiene un carácter a partir de su código

```
//a, b y c son variables con contenido numérico
var oper="a*b+c";
eval(oper);
```

Ejecuta el código de JavaScript que recibe como parámetro en forma de cadena



Definición de funciones de JavaScript

JavaScript permite al programador estructurar el código en funciones, de esta forma se consigue un doble objetivo:

- Organizar de forma adecuada el código y
- Reutilizar ciertas instrucciones que deben ser ejecutadas en diferentes partes del programa.

```
function nombre_funcion (argumento1, argumento2, ...)
{
    sentencias
    [return valor;]
}
```

Definición de una función

```
nombre_funcion(parametro1, parametro2, ...);

variable=nombre_funcion(parametro1, parametro2, ...);

<elemento-HTML evento="nombre_funcion(valor1,valor2,valor3);" >
```

Llamadas a una función

JavaScript es un lenguaje orientado a eventos, definiéndose las instrucciones manejadoras de dichos eventos en funciones. En esta imagen se ilustra la forma en la que se debe definir una instrucción en JavaScript y cómo ésta debe ser invocada desde otra parte del código. Como vemos, la función se define mediante la palabra reservada `function`, indicándose a continuación del nombre de la función los parámetros que serán recibidos en la llamada.

En caso de que la función devuelva un resultado, se deberá indicar este en una sentencia `return`. Por su parte, la llamada a una función puede realizarse directamente desde una instrucción de JavaScript o desde algún atributo de una etiqueta HTML, tal y como se indica en el ejemplo. Esta última opción es la utilizada para hacer una llamada a una función que debe ser ejecutada como respuesta a un evento.

Ejemplo

El siguiente ejemplo incluye una función que devuelve el resultado de dividir el mayor entre el menor de dos números:

```
<html>
<head>
<script language="javascript">
function division(x,y)
{
    if(x>y)
        return x/y;
    else
        return y/x;
}
</script>
</head>
<body>
<script language="javascript">
var n1=25,n2=10,resultado;
//obtiene el resultado de la división llamando
//a la función definida
resultado=division(n1,n2);
alert("La división vale:\n" + resultado);
</script>
</body>
</html>
```

Suele ser habitual, especialmente cuando se trata de funciones manejadoras de eventos, que las funciones se encuentren agrupadas en la cabecera del documento.



4. Eventos

Definición de eventos

Por evento entendemos una acción que el usuario puede realizar sobre la página, tales como hacer un click en un botón, arrastrar el ratón por encima de una imagen, seleccionar un elemento en una lista, etc.

Aunque la mayoría de los eventos se producen por acciones del usuario, también existen otros eventos que son generados por el propio navegador, como por ejemplo, el evento onload que es lanzado por el navegador al realizar la carga en memoria de la página.

En la siguiente se indican los principales eventos que pueden producirse en una página XHTML y son susceptibles de ser manejados mediante código JavaScript.



Evento	Suceso que lo produce
OnClick	Hacer un click sobre el objeto
onMousueOver	Situar el puntero del ratón encima del objeto
onMouseOut	El puntero del ratón sale del objeto
onMouseDown	Presionar un botón del ratón cuando el puntero está sobre el objeto
onMouseUp	Liberar un botón del ratón cuando el puntero está sobre el objeto
onFocus	El objeto recibe el foco
onBlur	El objeto pierde el foco
onKeyPress	Presionar una tecla cuando el foco esta sobre el objeto
onChange	Se modifica el valor de un select, text o textarea
OnLoad	Se produce la carga del documento
OnUnload	El documento se descarga de la memoria

Definición de funciones de JavaScript

Cuando el usuario realiza alguna acción sobre la página, el evento se produce en el objeto de la página donde el usuario ha ejecutado la acción, capturándose el mismo desde la etiqueta correspondiente a dicho objeto.



Por ejemplo un click en el documento provoca el evento onclick sobre la etiqueta <body>, mientras que el movimiento del ratón sobre una imagen provoca el evento onmouseover sobre la etiqueta correspondiente.

Cada evento tiene un manejador de evento que se encarga de responder automáticamente cuando ocurre un evento. Estos manejadores están representados por atributos de la etiqueta y su nombre coincide con el del evento.

En estos atributos manejadores se debe especificar las acciones a ejecutar como respuesta al evento, aunque, habitualmente, dichas acciones se agrupan en una función que es llamada desde el manejador.

Ejemplo

Por ejemplo, la siguiente página incluye una función, llamada mensaje(), que se ejecutará como respuesta al evento onclick sobre el documento. La función simplemente nos mostrará un mensaje de aviso en un cuadro de diálogo.

```
<html>
<head>
<title>Prueba de eventos</title>
<script language = "javascript">
function mensaje()
{
    window.alert("Ha hecho click en el documento");
}
</script>

</head>
<body onclick="mensaje();">

</body>
</html>
```

5. Objetos

Como la mayoría de los lenguajes modernos, JavaScript es un lenguaje basado en objetos. Esto significa que dentro de un script podemos hacer uso de objetos para aumentar la potencia y flexibilidad de nuestras aplicaciones.



Los objetos representan un tipo de estructura de programación que expone a quien lo va a utilizar a una serie de operaciones (métodos) que permiten resolver distintos problemas que se plantean en los programas.

Algunos de esos objetos, como los botones, formularios o ventanas son elementos que ofrecen un aspecto visual en la página, pero otros, como los arrays o las cadenas de caracteres, no tienen aspecto físico pero proporcionan una gran funcionalidad a las aplicaciones.

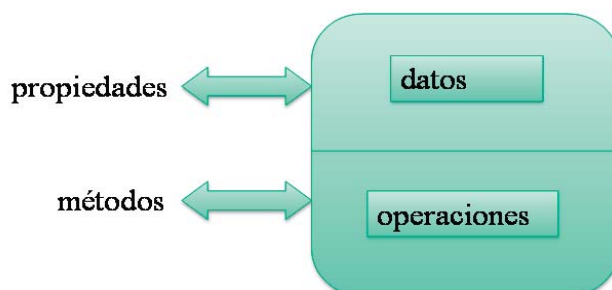
En cualquier caso, desde el punto de vista de la programación, un objeto es una “caja negra” que encapsula una serie de operaciones y de datos. Estas operaciones y datos son expuestas al exterior a través de propiedades y métodos

Propiedades

Representan características del objeto (por ejemplo, el título de una ventana). Se les puede asignar un valor y leer su contenido. Las propiedades pueden modificar a los objetos y la misma propiedad se puede aplicar a objetos completamente diferentes. El acceso a las propiedades desde el exterior del objeto se realiza utilizando la notación `nombre_objeto.propiedad`.

Métodos

Son acciones asociadas al objeto, es decir son las cosas que pueden hacer los objetos (por ejemplo, el método `open()` para abrir una ventana). Estas acciones están implementadas como funciones dentro del objeto. El acceso a los métodos desde el exterior del objeto se realiza utilizando la notación `nombre_objeto.metodo(parámetros)`.



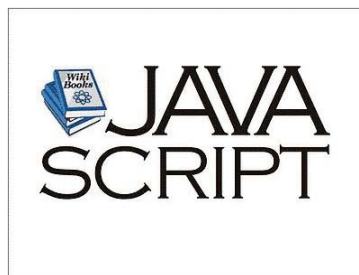
`document.location="http://www.google.es";` ← Utilización de una propiedad

`window.alert("hola");` ← Llamada a un método

Referencia a un objeto

Para hacer referencia a un objeto desde código unas veces se utilizará el nombre del objeto y otras una variable que lo referencie, todo dependerá del tipo del objeto de que se trate.

En ocasiones, a partir del nombre del tipo de objeto (clase), habrá que crear el objeto y guardar una referencia al mismo en una variable con la que poder después acceder a sus propiedades y métodos.

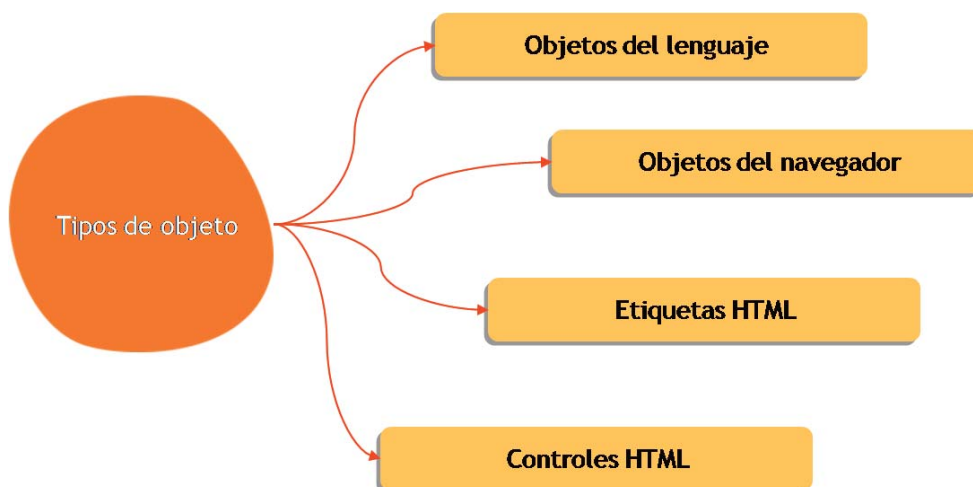


La creación de objetos a partir de una clase se realiza a través del operador new, utilizando la siguiente sintaxis: `var variable = new Clase_objeto();`

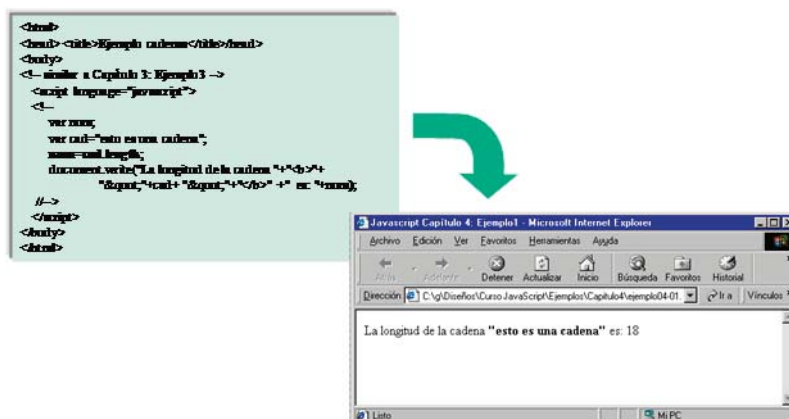
La instrucción anterior crea el objeto y almacena la referencia en la variable.

Tipos de objeto

De cara a facilitar su estudio, el conjunto de objetos que podemos utilizar desde un programa javascript podemos dividirlo en cuatro tipos:



En cuanto a los métodos de este tipo de objeto, la siguiente tabla nos muestra los más importantes.



Métodos	Acción
charAt(n)	Devuelve el carácter situado en la posición 'n' (la primera posición es la 0)
indexOf(fragmento [,posición])	Devuelve la posición de un fragmento de la cadena a partir de una posición determinada.
lastIndexOf(fragmento [,posición])	Devuelve la posición de la última ocurrencia de un fragmento de la cadena a partir de una posición determinada, hacia atrás.
link(URL)	Convierte la cadena en un vínculo
substring(x,y)	Muestra un fragmento de la cadena que empieza en la posición indicada por el menor de los valores
toLowerCase()	Devuelve la cadena en minúsculas.
toUpperCase()	Devuelve la cadena en mayúsculas.

Objetos del lenguaje: Array

Un array representa un conjunto de elementos que pueden ser recorridos mediante un índice.

En JavaScript los array se manejan como objetos de tipo Array, cuyos elementos pueden contener cualquier tipo básico, incluso pueden mezclarse tipos diferentes en un mismo array.

Los arrays en JavaScript no tienen prefijado el número de elementos en la declaración, siendo 0 el índice del primer elemento.

Para acceder a cada elemento individual usaremos un índice, entre corchetes, a continuación del nombre de la variable que apunta al array, teniendo en cuenta, como hemos indicado antes, que el primer elemento tiene índice 0.

```
var datos = new Array();
```

Para poder utilizar Array es necesario crear explícitamente el objeto utilizando el operador new, tal y como explicamos anteriormente

“índice 0”

```
datos[0] = 25;  
datos[1] = 15;  
datos[2] = 100;  
datos[3] = 50;
```

Objetos del lenguaje: Array bucle for

La ventaja de los arrays frente a las variables normales es que se pueden recorrer con un bucle for fácilmente.

Dado que su tamaño no es fijo, los arrays disponen de una propiedad llamada length que nos indica en cada momento el número de elementos que hay en el array.

```
1 <html>  
2 <head>  
3 <title>RTSHJS test</title>  
4 <script>alert(1);</script>  
5 <style>  
6     h1 {color:red;}  
7 </style>  
8 </head>  
9 <body bgcolor="black" margin="1">  
10
```

También se puede inicializar el array a la vez que se crea. Por ejemplo, la siguiente instrucción crearía un array con tres nombres.

```
var nombres = new Array("Juan", "Ana", "Beatriz");
```


Métodos	Acción
join(separador)	Devuelve un string formado por el contenido del array, separando cada elemento del siguiente por 'separador' (por defecto por comas).
reverse()	Invierte el orden de los elementos de un array.
sort()	Ordena los elementos de un array alfabéticamente.
push(elem, elem,...)	Añade nuevos elementos al array, devolviendo la nueva longitud
unshift(elem, elem,...)	Añade nuevos elementos al principio del array del mismo (v 1.3) o el último elemento añadido (v 1.2)
pop()	Elimina el último elemento del array, devolviendo dicho elemento
shift()	Elimina el primer elemento del array, devolviendo dicho elemento

The screenshot displays two parts: a code editor on the left and a web browser window on the right.

Code Editor:

```
<HTML>  
<HEAD><TITLE> Javascript Capitulo 4: Ejemplo3</TITLE></HEAD>  
<BODY>  
  <SCRIPT LANGUAGE=javascript>  
    <!--  
    MisSemana= new Array();  
    MisSemana[0]= "Lunes";  
    MisSemana[1]= "Martes";  
    MisSemana[2]= "Miércoles";  
    MisSemana[3]= "Jueves";  
    MisSemana[4]= "Viernes";  
    MisSemana[5]= "Sábado";  
    MisSemana[6]= "Domingo";  
    document.write("Tengo "+MisSemana.length+" elementos en mi tabla de  
document.write("Aquí estan todos relacionados "+ "<BR>" + "Abday,Abday,  
+MisSemana.join()+"<BR>");  
document.write("Aquí estan ordenados al revés "+ "<BR>" + "Abday,Abday,  
+MisSemana.reverse().join()+"<BR>");  
document.write("Aquí estan ordenados alfabéticamente "+ "<BR>" +  
"Abday,Abday,Abday," + MisSemana.sort().join()+"<BR>");  
document.write("Aquí estan ordenados alfabéticamente al revés "+ "<BR>" +  
"Abday,Abday,Abday," + MisSemana.sort().reverse().join()+"<BR>";  
    //-->  
  </SCRIPT>  
</BODY>  
</HTML>
```

Browser Window:

Title bar: Javascript Capitulo 4: Ejemplo3: objeto Array() - Microsoft Int...

Menu bar: Archivo, Edición, Ver, Herramientas, Ayuda

Toolbar: Abrir, Guardar, Detener, Actualizar, Inicio, Búsqueda

Status bar: Dirección C:\Program Files\Curso JavaScript\Ejemplos\Cap... | Vindic...

Content area:

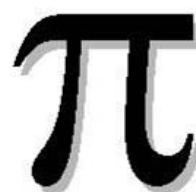
Tengo 7 elementos en mi tabla de dias
Aquí estan todos relacionados
Lunes,Martes,Miercoles,Jueves,Viernes,Sabado,Domingo
Aquí estan ordenados al revés
Domingo,Sabado,Viernes,Jueves,Miercoles,Martes,Lunes
Aquí estan ordenados alfabéticamente
Domingo,Jueves,Lunes,Martes,Miercoles,Sabado,Viernes
Aquí estan ordenados alfabéticamente al revés
Viernes,Sabado,Miercoles,Martes,Lunes,Jueves,Domingo

Objetos del lenguaje: Math

El objeto Math nos proporciona un conjunto de funciones matemáticas de uso general en cualquier programa.

Para hacer uso de estos métodos no necesitamos crear ningún objeto especial, pudiéndose utilizar el nombre genérico del objeto.

```
Math.metodo();
Math.propiedad;
```



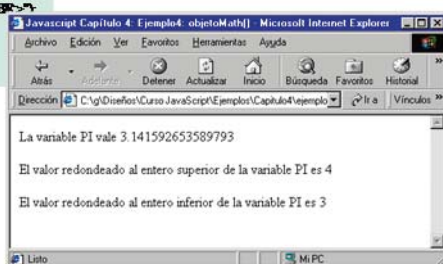
Como propiedad más importante de este tipo de objeto está la propiedad PI que contiene al número pi (3.1416).

En cuanto a los métodos, la siguiente tabla nos muestra los más importantes.

En la siguiente imagen se muestra un ejemplo de uso de los métodos de este objeto.

Métodos	Acción
abs(n)	Devuelve el valor absoluto de un número
ceil(n)	Redondea un número hacia su inmediato superior
floor(n)	Redondea un número hacia el inferior.
max(x,y), min(x,y)	Devuelve el valor mayor (o menor) de sus dos argumentos ('x' e 'y').
pow(base,exponente)	Calcula la potencia del número indicado en 'base' elevado a 'exponente'.
random()	Genera un número aleatorio entre 0 y 1.
round(n)	Redondea un número hacia el entero más cercano.

```
<HTML>
<HEAD><TITLE>Javascript Capítulo 4: Ejemplo4:
objetoMath</TITLE></HEAD>
<BODY>
<SCRIPTLANGUAGE=javascript>
<!--
var valor_PI=Math.PI;
var valor_redondeado_superior=Math.ceil(valor_PI);
var valor_redondeado_inferior=Math.floor(valor_PI);
document.write("La variable PI vale " + valor_PI + "<BR>");
document.write("El valor redondeado al entero superior de la variable
PI es " + valor_redondeado_superior + "<BR>");
document.write("El valor redondeado al entero inferior de la variable
PI es " + valor_redondeado_inferior + "<BR>");
//-->
</SCRIPT>
</BODY>
</HTML>
```



Objetos del lenguaje: Date

El objeto Date permite manejar fechas y horas. Como en el caso de Array, es necesario crear explícitamente una instancia del objeto utilizando el operador new.

```
var fecha_actual = new Date();
```

En el caso anterior, el objeto Date creado representa la fecha y hora actuales del sistema. Si quisiéramos crear un objeto Date con una fecha y hora concretas, podríamos utilizar cualquiera de las siguientes formas.

```
var fecha_otra=new Date(var_año, var_mes, var_dia);
var fecha_otra=new Date(v_anno, v_mes, v_dia, v_hora, v_minuto, v_segundo);
```

Hay que tener en cuenta que JavaScript maneja las fechas en milisegundos. Los meses empiezan a contar en 0 y acaban en 11, y los días van desde el 0 (Domingo) al 6 (Sábado).

January	February	March
April	May	June
July	August	September
October	November	December

Los métodos indicados en la siguiente tabla nos permiten obtener información sobre los distintos parámetros de la fecha.

Métodos	Acción
getDate()	Devuelve el día actual del mes como un entero entre 1 y 31
getDay()	Devuelve el día actual de la semana como un entero entre 0 y 6
getHours()	Devuelve la hora actual (comprendida entre 0 y 23)
getMinutes()	Devuelve los minutos de la hora actual (comprendidos entre 0 y 59)
getMonth()	Devuelve el mes actual (entre 0 y11)
getSeconds()	Devuelve los segundos del minuto actual (comprendida entre 0 y 59)
getYear()	Devuelve el año actual.
getFullYear()	Devuelve el año actual.

Por su parte, los métodos indicados en esta tabla, nos permiten manipular los diferentes datos de la fecha.

Métodos	Acción
<code>setDate(día_mes)</code>	Pone el día del mes actual en el objeto usado
<code>setDay(día_semana)</code>	Pone el día de la semana actual al objeto usado
<code>setHours(horas)</code>	Pone la hora actual al objeto usado
<code>setMinutes(minutos)</code>	Pone los minutos de la hora actual en el objeto usado.
<code>setMonth(mes)</code>	Pone el mes actual en el objeto usado.
<code>setSeconds(segundos)</code>	Pone los segundos del minuto actual en el objeto usado.
<code>setTime(milisegundos)</code>	Pone la fecha que dista los milisegundos indicados desde el 1 de enero de 1970
<code>setYear(año)</code>	Pone el año actual en el objeto usado
<code>toGMTString</code>	Devuelve el valor del objeto actual de la zona horaria GMT como una cadena.
<code>toLocaleString</code>	Devuelve el valor del objeto actual como una cadena

Objetos del navegador

Los objetos del lenguaje nos ofrecen muchas posibilidades a la hora de crear aplicaciones del lado cliente, sin embargo, estas posibilidades se ven multiplicadas con el uso de los objetos del navegador.

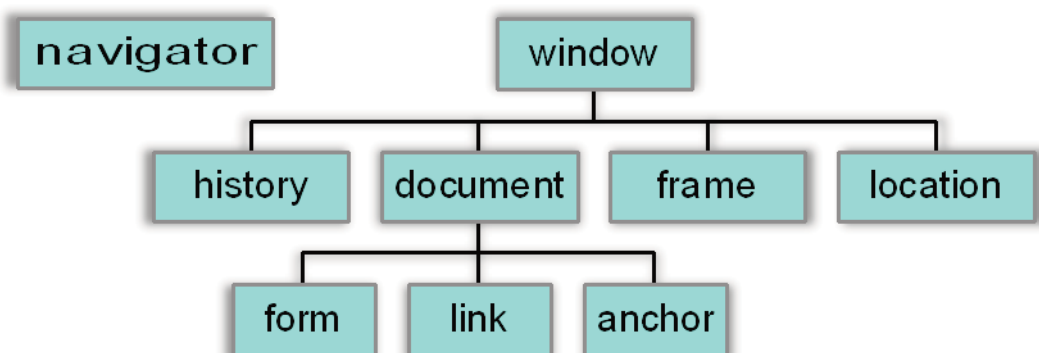
Estos objetos disponen de una serie de propiedades y métodos que pueden ser utilizados en cualquier script del cliente sin tener que instanciar el objeto, simplemente se utilizará el nombre del objeto y el operador "." para acceder a la propiedad o método:

`window.status`



Cuando se carga una página en un navegador se crean una serie de objetos característicos de éste, llamados objetos del navegador.

Como puedes ver en la siguiente imagen, los objetos del navegador están organizados jerárquicamente, lo que significa que para acceder a uno de ellos hay que indicar su referencia completa.



`window.document.anchor`

Objetos del navegador: Window

Se trata del objeto principal de la jerarquía, representa la ventana del navegador y cada una de las áreas o marcos definidos. Es el objeto que más propiedades y métodos tiene, con ellos se pueden enviar mensajes al usuario, solicitar datos, definir temporizadores, etc.



De hecho, los métodos para la generación de cuadros de diálogo como `alert()`, `prompt()` o `confirm()`, no son realmente funciones del lenguaje, sino métodos del objeto `window`.

Mediante el objeto `window` se accede además al resto de objetos de la jerarquía.

`window.document.forms`
o simplemente
`document.forms`

`window.alert("texto")` equivale a `alert("texto")`

LENGUAJE JAVASCRIPT

Dado que es el objeto predeterminado del navegador, se puede acceder a sus propiedades y métodos escribiendo únicamente el nombre de éstos, sin tener que indicar la palabra window delante

La siguiente tabla contiene algunas de las propiedades más importantes de este objeto. Como hemos mencionado anteriormente, este objeto es el que más propiedades y métodos tiene de todos los objetos del navegador. Veamos también, sus métodos más importantes.

Propiedades	Significado
name	Nombre del objeto Window
parent	Referencia a la ventana que contiene <FRAMESET>
self	Referencia a la propia ventana
top	Referencia a la ventana superior
status	Mensaje que aparece en la barra de estado del navegador
screenLeft	Distancia horizontal del área cliente
screenTop	Distancia vertical del área cliente.

Métodos	Significado
alert(<i>mensaje</i>)	Muestra un mensaje en un cuadro de diálogo con un botón <i>Aceptar</i>
confirm(<i>mensaje</i>)	Igual que el anterior pero con botones de <i>Aceptar</i> y <i>Cancelar</i> , devuelve true o false
prompt(<i>mensaje</i>)	Presenta un mensaje con una caja de texto
setTimeout(<i>expresion</i> , <i>tiempo</i>)	Evalúa la expresión cuando transcurre el intervalo de tiempo
setInterval(<i>expresion</i> , <i>tiempo</i>)	Evalúa la expresión periódicamente, cada cierto intervalo de tiempo
open(<i>url</i> , <i>nombre</i> , <i>características</i>)	Abre la url en la ventana llamada nombre, con las características indicadas
clearTimeout(identificador)	Elimina el temporizador especificado
clearInterval(identificador)	Elimina el temporizador especificado
close()	Cierra el objeto window
moveTo(posx, posy)	Desplaza la ventana a la posición especificada
resizeTo(w,h)	Establece un nuevo tamaño para la ventana
navigate(url)	Carga en la ventana el documento especificado en la url

Objetos del navegador: Temporizadores

Un temporizador es un objeto que realiza la ejecución de una tarea transcurrido un determinado periodo de tiempo. El objeto window dispone de dos métodos para generar temporizadores:

setTimeout

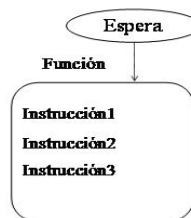
(funcion, tiempo): genera un temporizador que realiza una llamada a la función cada vez que transcurre el intervalo de tiempo definido en el parámetro tiempo

setInterval

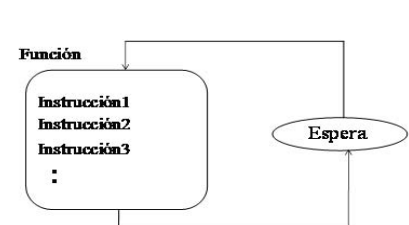
(funcion, tiempo): genera un temporizador que, transcurrido un determinado periodo de tiempo, hace una llamada a la función. En este caso, a diferencia de setTimeout, el temporizador no se reactiva de nuevo, funcionando por tanto como un retardador. Si quisiéramos que la función se ejecutase repetidamente, sería necesario incluir una nueva llamada a setTimeout dentro de la función:

```
function ejemplo()
{
  //código función
  :
  setTimeout("funcion();",1000);
}
```

setTimeout



setInterval

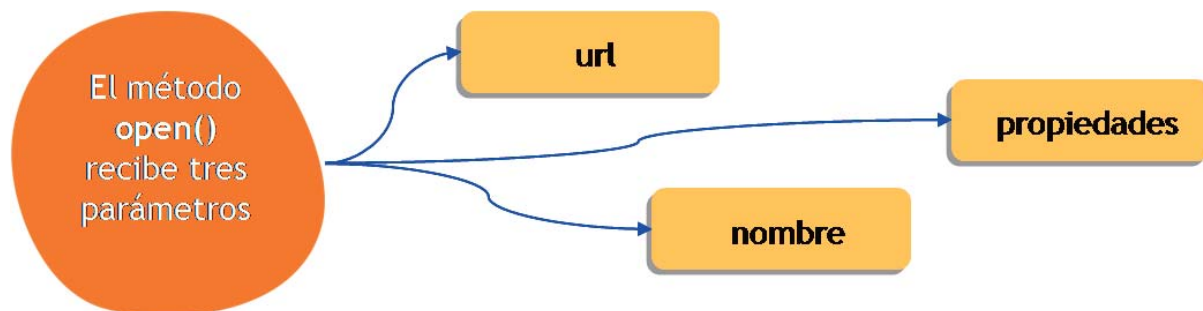


Si queremos desactivar un temporizador creado con setInterval, debemos recurrir al método clearInterval, al que hay que pasarle como parámetro el identificador (objeto devuelto en la llamada al método) del temporizador creado.

```
//activa el temporizador
var tempo = window.setInterval("repite();", 500);
:
//desactiva el temporizador
window.clearInterval(tempo);
```

Objetos del navegador: Apertura de una ventana

Otra de las funcionalidades que nos ofrece el objeto window es la apertura de nuevas ventanas del navegador desde código. Para realizar esta operación recurriremos al método `open()`.



url

Url del documento que se va a mostrar en la ventana. Cuando se quiere generar dinámicamente la página desde código este parámetro será cadena vacía.

nombre

Nombre asociado a la ventana. Puede utilizarse como valor del atributo `target` de un enlace, cuando se quiera cargar en la ventana el documento asociado al enlace.

propiedades

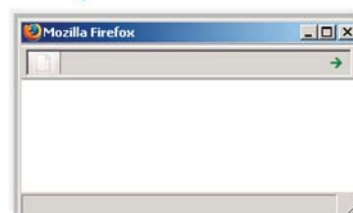
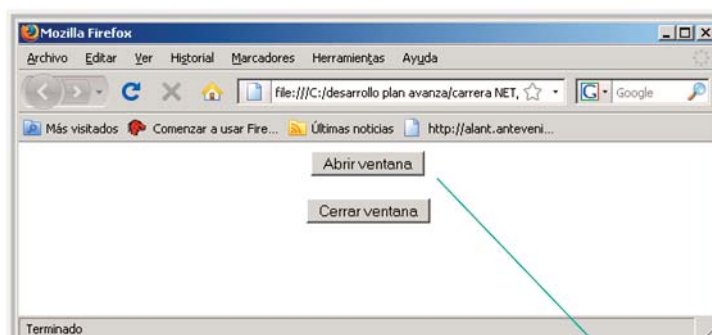
Permite definir distintas características de la ventana, como tamaño, posición, etc.

La llamada al método `open()` devuelve un objeto window que representa la nueva ventana creada.

Además de `open()`, window dispone de otro método llamado `close()` que permite cerrar el objeto window sobre el que se aplica.

Ejemplo

```
<html>
<head>
<script language="javascript">
  var vent;
  function nuevaVentana(){
    vent=window.open("", "Ventana",
      "height=100,width=300,top=200,left=400");
  }
  function cerrarVentana(){
    vent.close();
  }
</script></head>
<body>
<center>
<input type="button" value="Abrir ventana"
  onclick="nuevaVentana();" /><br/><br/>
<input type="button" value="Cerrar ventana"
  onclick="cerrarVentana();" />
</center>
</body>
</html>
```



Objetos del navegador: Document

El objeto document representa el documento que se está visualizando en el objeto window, equivale a la etiqueta <body> de HTML, por lo que puede acceder a las propiedades y métodos de este objeto utilizando document o el valor del atributo id de la etiqueta. La siguiente tabla contiene las propiedades más importantes de este objeto.



Propiedades	Significado
bgColor	Color de fondo del documento
fgColor	Color de primer plano
location	Cadena que contiene la URL del documento actual
title	Título del documento

Los métodos más importantes del objeto document son:

write ()

getElementById ()

write

write(texto_HTML)

Mediante este método podemos escribir cualquier texto con formato HTML en el interior del documento. Este método resulta útil, por ejemplo, para generar dinámicamente el contenido de una ventana abierta desde código. El siguiente código corresponde a la página de ejemplo anterior encargada de abrir una ventana. En este caso, se incluyen instrucciones que generan un texto en el interior de la misma, así como un botón para proceder a su cierre:

```
<html>
<head>
<script language="javascript">
  function nuevaVentana(){
    var vent=window.open("", "Ventana",
      "height=100,width=300,top=200,left=400");
    generarContenido(vent);
  }
  function generarContenido(ventana){
    ventana.document.write("<html><body>");
    ventana.document.write("<h1>Texto generado desde otra
      ventana</h1>");
    ventana.document.write("<br/><br/>");
    ventana.document.write("<input type='button'
      value='cerrar' onclick='self.close();'/>");
    ventana.document.write("</body></html>");
  }
</script></head>
<body>
<center>
<input type="button" value="Abrir ventana"
  onclick="nuevaVentana();" /><br/><br/>
</center>
</body>
</html>
```

getElementById ()

Otro método importante del objeto document es getElementById () el cual nos permite obtener una referencia a un determinado objeto etiqueta existente en la página a partir de su identificador. El uso de este método lo veremos más adelante cuando analicemos los objetos HTML. Como ejemplo, si tenemos una etiqueta con el identificador asociado “objeto” <p id = “objeto”>

Para obtener una referencia al mismo sería:

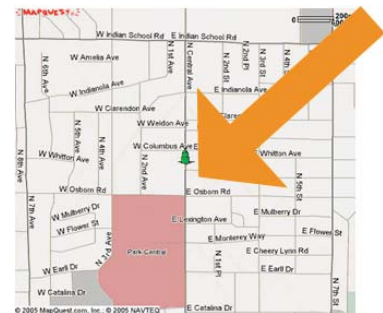
```
var obj = document.getElementById("objeto");
```

Objetos del navegador: Location

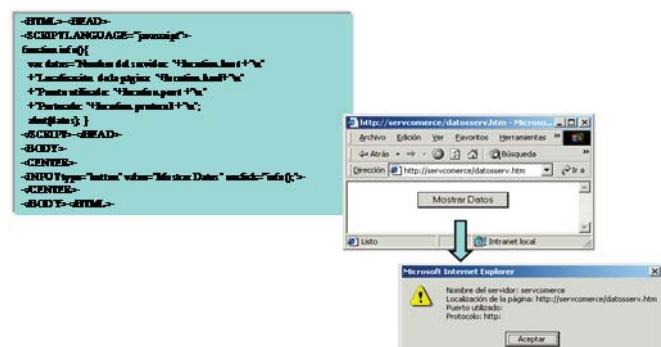
El objeto location proporciona información sobre la ubicación del documento, como su dirección, nombre del servidor, etc.

Puede utilizarse en aquellas aplicaciones en las que se quiera determinar el origen del documento. La siguiente tabla nos muestra las propiedades más importantes de este objeto.

Veamos un ejemplo de utilización de este objeto consistente en una página con un botón de pulsación que al ser activado nos muestra una serie de datos sobre el servidor y la página mostrada. Pulsa sobre la imagen.

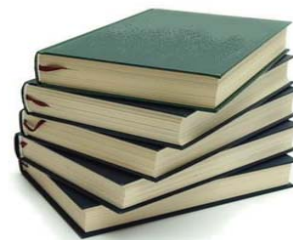


Propiedades	Significado
protocol	Protocolo utilizado
host	Nombre del servidor y n° de puerto
href	Dirección completa del documento
port	N° de puerto



Objetos del navegador: History

El objeto history representa el historial de documentos visitados. A través de sus métodos y propiedades podemos desplazarnos por el historial del navegador.



Las siguientes tablas nos muestran las propiedades y métodos más importantes de este objeto.

Propiedades	Significado
current	URL de la entrada actual en el historial
next	URL de la siguiente entrada en el historial
previous	URL de la entrada anterior en el historial
length	Nº de entradas en el historial

Métodos	Significado
back()	Carga el documento anterior al actual
forward()	Carga el siguiente documento
go(<i>posicion</i>)	Carga el documento cuya posición en el historial coincida con <i>posicion</i>

Objetos del navegador: Navigator

El objeto Navigator representa al propio navegador por lo que proporciona información del mismo, como nombre de la aplicación y versión.

La principal utilidad de este objeto es la identificación del tipo de navegador que se ha descargado la página y las características admitidas por éste.

Esto suele ser de gran utilidad cuando se tiene previsto ejecutar diferentes instrucciones dependiendo del tipo y versión de navegador.

La tabla nos muestra las propiedades más importantes de este objeto.

Propiedades	Significado
appName	Nombre del navegador cliente
appVersion	Versión del navegador cliente
userAgent	Cabecera completa enviada por el navegador en una petición HTTP
cookieEnabled	Indica si el cliente admite o no cookies.

Objetos etiquetas HTML

Toda etiqueta HTML representa un objeto. Como cualquier otro objeto, expone una serie de propiedades y métodos que pueden ser utilizados desde código JavaScript.

A través de las propiedades y métodos de las etiquetas podemos modificar dinámicamente el aspecto de la página, alterando la posición y formato de los elementos o incluso el contenido de los mismos.

Para poder acceder a las propiedades y métodos de las etiquetas es necesario que éstas expongan el atributo `id`, cuyo valor representa el identificador del objeto para ser manipulado desde código.

A partir de este identificador, puede utilizarse el método `getElementById()` del objeto `document` que estudiamos anteriormente para obtener una referencia al objeto etiqueta. Esta referencia se utilizará directamente con el operador `."` para acceder a las propiedades del objeto.

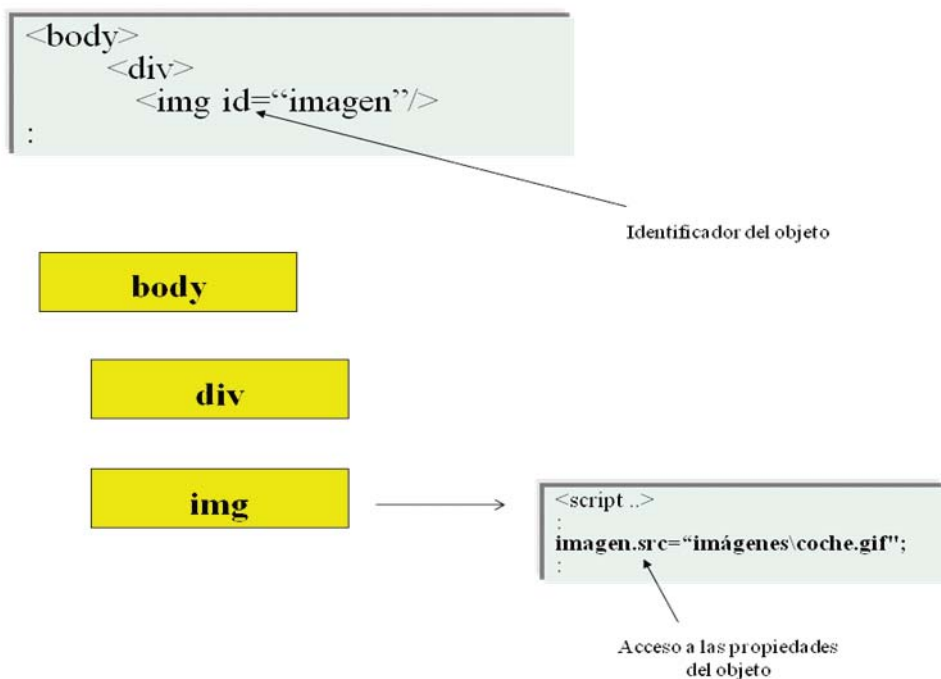
Una de las propiedades más importantes que se indican en la tabla anterior es `style`. A través de ella podemos acceder individualmente a todas las propiedades de estilo CSS aplicables sobre la etiqueta.

Por ejemplo, el siguiente script modificaría tanto el grosor del texto contenido en una etiqueta como el estilo de dicho texto:

```
<html>
<head>
<script language="javascript">
    var par = document.getElementById("parrafo");
    par.style.fontWeight="bold";
    par.style.fontStyle="italic";
</script>
</head>
<body>
    <p id="parrafo">
        Texto de prueba
    </p>
</body>
</html>
par.innerHTML = "<center>Nuevo texto</center>";
```

LENGUAJE JAVASCRIPT

La alteración de estas propiedades de estilo en funciones manejadoras de evento permiten alterar el aspecto y formato de la página de forma dinámica ante acciones del usuario.



Propiedades	Significado
className	Permite establecer un estilo generico al objeto, asignando a esta propiedad el nombre de dicho estilo
style	Esta propiedad proporciona acceso a las propiedades de estilo CSS de la etiqueta
innerText	Representa texto contenido en la etiqueta
innerHTML	Representa el texto HTML (datos + etiquetas) incluido dentro de la etiqueta.
bgColor	Indica el color de fondo del área delimitada por la etiqueta
tagName	Contiene el nombre genérico de la etiqueta

Objetos controles HTML

Los controles HTML se utilizan para solicitar datos de usuario a través de una página HTML. Estos objetos proporcionan propiedades y métodos que pueden utilizarse dentro de un script de cliente para realizar tareas diversas como la validación de datos antes de ser enviados al servidor para su procesamiento.

A través del atributo id podemos acceder desde código a los valores de los distintos atributos de la etiqueta. Por ejemplo, dada la definición del siguiente control caja de texto: `<input type = "text" id = "dato" />`

El siguiente bloque de instrucciones JavaScript nos mostraría en un cuadro de diálogo el valor introducido por el usuario en el campo de texto:

```
var caja = document.getElementById("dato");
alert (caja.value); //acceso al atributo value
```

En el caso de un control lista de selección, la propiedad options del mismo nos da acceso al array de objetos `<option>` definidos en su interior. Por su parte, `selectedIndex` indica el índice del elemento seleccionado. Cada uno de los objetos `<option>` dispone a su vez de la propiedad `value` para conocer el valor del mismo, `text` que nos informa sobre el texto delimitado por la etiqueta o `selected`, que nos informa sobre si la opción se encuentra o no seleccionada.



Ejemplo

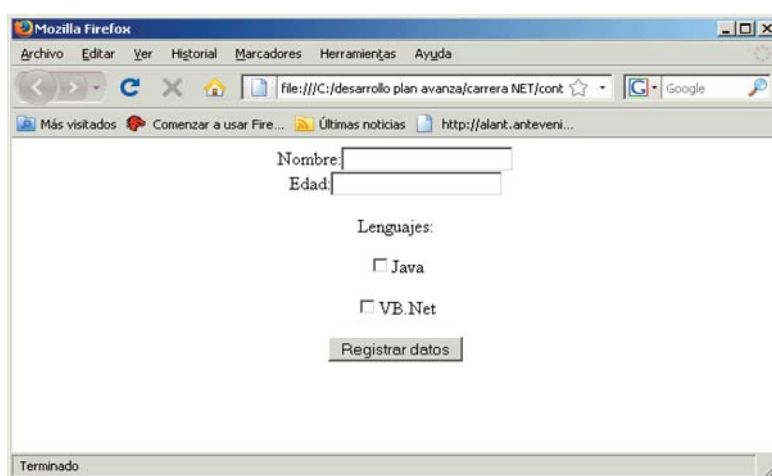
La siguiente página nos muestra en un cuadro de diálogo el texto de la opción seleccionada por el usuario cuando este activa el botón de pulsación:

```
<html>
<head>
<script language="javascript">
  function pulsar(){
    var obj=document.getElementById("lista");
    alert(obj.options[obj.selectedIndex].text);
  }

</script></head>
<body id="cuerpo">
<center>
<select id="lista">
  <option>Opción primera</option>
  <option>Opción segunda</option>
  <option>Opción tercera</option>
  <option>Opción cuarta</option>
</select>
<br/><br/>
<input type="button" value="ver opción"
  onclick="pulsar();" />
</center>
</body>
</html>
```

Comprobación de datos

Como hemos comentado anteriormente, mediante la gestión de eventos en la interfaz y utilizando las propiedades de los controles HTML, se pueden comprobar los datos introducidos por los usuarios en un formulario antes de ser enviados al servidor.



Veamos un ejemplo, correspondiente al formulario de la imagen, se comprobará que los datos introducidos en código sean siempre numéricos y que siempre se seleccione al menos un lenguaje. El control se lleva a cabo en los eventos `onChange()` de la caja de texto y `onClick()` del botón de pulsación.

Ejemplo

```
<html>
<head>
<script language="javascript">
function comprobar(){
var java = document.getElementById("c1");
var visual = document.getElementById("c2");
if(!java.checked && !visual.checked){
    window.alert("Debe seleccionar algún lenguaje");
}
else{
    //realize el envío del formulario
    document.getElementById("formulario").submit();
}
}
function numero(){
    var edad = document.getElementById("edad");
    if(isNaN(edad.value)){
        window.alert("Debe escribir un valor numérico");
        edad.value="";
        edad.focus();
    }
}
</script></head>
<body id="cuerpo">
<center>
<form name="formulario" action="registro.asp"
    method="post">
    Nombre:<input id="nombre"/><br/>
    Edad:<input id="edad" onchange="numero()"/><br/>
    <p>Lenguajes:</p>
    <p><input id="c1" type="checkbox">Java</p>
    <p><input id="c2" type="checkbox">VB.Net</p>
    <p align="center">
        <input type="button" onclick="comprobar()"
        value="Registrar datos">
    </p>
</form>
</center>
</body>
</html>
```



6. Resumen

Has llegado al final de esta lección de formación que denominamos “El lenguaje JavaScript”
En esta lección hemos estudiado los siguientes contenidos:

