**FUNDAMENTOS DE JAVASCRIPT**

1. **¿Qué es JavaScript?**
   1. **Versiones JavaScript**
2. **Sintaxis básica**
   1. **Datos y Variables**
   2. **Operadores**
   3. **Condicionales**
   4. **Iteración**
   5. **Funciones**
3. **Objetos del Lenguaje**
   1. **Cadenas**
   2. **Matemáticas**
   3. **De Fecha y Hora**
   4. **Windows**
   5. **Navegador**
4. **¿Dónde incluimos JavaScript?**
5. **¿Qué es JavaScript?**

JavaScript es un lenguaje interpretado, de secuencia de comandos, está orientado a objetos y es soportado ampliamente por todos los navegadores.

Es **interpretado**, lo que quiere decir que no necesita ser compilado como otros lenguajes de programación.

Es un lenguaje de **script**, lo que significa que es código que se inserta dentro de un documento.

Como Javascript fue desarrollado para potenciar la creación de páginas web dinámicas, Javascript inserta su código en páginas Html para ampliar las posibilidades de éste.

Los navegadores son los encargados de interpretar este código incluido en las páginas web.

**JavaScript** y Java son dos cosas distintas. Principalmente porque Java sí que es un lenguaje de programación completo. Lo único que comparten es la

misma sintaxis, aunque tiene, por supuesto, sus propias peculiaridades.

Sus principales características son:

* Es **interpretado** (que no compilado) por el cliente.
* Está **basado en objetos**. No es, como Java, un lenguaje de programación orientada a objetos (OOP). JavaScript no emplea clases ni herencia, típicas de la OOP.
* Puede ser **embebido**, su código se incluye en las páginas HTML.
* No es necesario declarar los tipos de variables que van a utilizarse (**loose typing**).
* Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto no se compila.
* No puede escribir automáticamente al disco duro.

JavaScript nos permite programar fácilmente todos los objetos que ofrece el

navegador de Internet.

**1.1. Versiones JavaScript**

**JavaScript** está basado en el estándar [ECMAScript](http://es.wikipedia.org/wiki/ECMAScript) . ECMA es la European Computer Manufacturers Association y es el organismo encargado de intentar estandarizar el lenguaje para que no haya incompatibilidades entre los distintos navegadores.

En esta tabla podemos ver las distintas versiones de JavaScript y su equivalencia con ECMAScript. Además podemos ver las distintas versiones que utilizan los navegadores más comunes. Microsoft llamó JScript a su lenguaje JavaScript para evitar problemas relacionadas con la marca:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Version** | **Release date** | **Equivalent to** | **Netscape Navigator** | **Mozilla Firefox** | **Internet Explorer** | **Opera** | **Safari** | **Google Chrome** |
| 1.0 | March 1996 |  | 2.0 |  | 3.0 |  |  |  |
| 1.1 | August 1996 |  | 3.0 |  |  |  |  |  |
| 1.2 | June 1997 |  | 4.0-4.05 |  |  |  |  |  |
| 1.3 | October 1998 | ECMA-262 1st edition / ECMA-262 2nd edition | 4.06-4.7x |  | 4.0 |  |  |  |
| 1.4 |  |  | Netscape Server |  |  |  |  |  |
| 1.5 | November 2000 | ECMA-262 3rd edition | 6.0 | 1.0 | 5.5 (JScript 5.5), 6 (JScript 5.6), 7 (JScript 5.7), 8 (JScript 5.8) | 6.0-11.0 | 3.0-5 | 1.0-10.0.666 |
| 1.6 | November 2005 | 1.5 + Array extras + Array and String generics + [E4X](http://en.wikipedia.org/wiki/E4X) |  | 1.5 |  |  |  |  |
| 1.7 | October 2006 | 1.6 + Pythonic generators + Iterators + let |  | 2.0 |  |  |  |  |
| 1.8 | June 2008 | 1.7 + [Generator expressions](http://en.wikipedia.org/wiki/Generator_%28computer_science%29) + [Expression closures](http://en.wikipedia.org/wiki/Closure_%28computer_science%29) |  | 3.0 |  |  |  |  |
| 1.8.1 |  | 1.8 + [Native JSON](http://en.wikipedia.org/wiki/JSON#Native_JSON) support + Minor Updates |  | 3.5 |  |  |  |  |
| 1.8.2 | June 22, 2009 | 1.8.1 + Minor updates |  | 3.6 |  |  |  |  |
| 1.8.5 | July 27, 2010 | 1.8.1 + [ECMAScript](http://en.wikipedia.org/wiki/ECMAScript) 5 Compliance |  | 4 | 9 |  |  |  |

El programa que va a interpretar los programas JavaScript es el propio navegador, lo que significa que si el nuestro no soporta JavaScript o lo tenemos desactivado, no podremos ejecutar las funciones que programemos.

<noscript>

Tu navegador no soporta Javascript.

</noscript>

La etiqueta <NOSCRIPT> permite definir un contenido para que aparezca únicamente en los navegadores sin soporte de subprogramas. Esto es válido para cualquier tipo de subprograma, no sólo Javascript. En el caso que el navegador soporte JavaScript todo el contenido que se encuentre entre las etiquetas de <NOSCRIPT> y </NOSCRIPT> será ignorado.

Casi todos los navegadores soportan el uso de esta etiqueta.

**2. Sintaxis básica**

En este apartado veremos los diferentes tipos de **datos** existentes en el lenguaje, veremos cómo definir **variables**, utilizaremos **operadores** para realizar distintas operaciones, y además, condicionaremos el flujo secuencial del intérprete aplicando distintas **condicionales**.

Por último, comprenderemos como se declara y define una **función**, y

visualizaremos las funciones disponibles en el lenguaje **no asociada a objetos.**

**2.1. Datos y Variables**

Las variables se utilizan para almacenar y recuperar datos, también conocidos como "**valores**".

Para hacer referencia a una variable debemos asignarle un nombre, y éste debe cumplir ciertas reglas. Pero antes debemos saber que datos podemos usar en JavaScript:

**Datos**

Los tipos de datos que existen en JavaScript son : numérico, cadena, boolean, objeto.

* **Numérico: c**omprende los números enteros y los decimales. Los números enteros pueden ser positivos y negativos. Los decimales pueden contener el punto decimal o bien estar expresados en notación científica. Los enteros, si no se indica otra cosa, se asumen que están expresados en base 10. Para indicar otras bases se utiliza el 0 para octales y el 0x para hexadecimales, por ejemplo.
* **Cadena:** es el tipo utilizado para las cadenas de caracteres. Las cadenas en JavaScript deben estar delimitadas por comillas dobles. Cuando queremos inicializar una variable con una cadena vacía lo haremos utilizando las comillas dobles de esta forma:

cadena = "";

En el caso de que las comillas dobles formen parte de la cadena de caracteres, debemos delimitar la cadena entre comillas simples:

'Cadena con "comillas dobles" en su interior'

* **Objeto:** JavaScript es un lenguaje de programación que dispone de un conjunto muy reducido de instrucciones.

La potencia de este lenguaje se consigue mediante el uso de una serie de objetos de carácter general que añade estas funcionalidades. Las variables que hacen uso de estos objetos, utilizan este tipo de datos para su procesamiento. En capítulos posteriores de este manual se explicar detalladamente el uso de objetos.

* **Boolean:** el tipo de dato ***boleano*** o ***lógico*** se utiliza en las instrucciones condicionales para evaluar una expresión lógica. Tiene dos valores posibles y contrarios:

verdadero (***true***) y falso (***false***).

El valor lógico ***false*** equivale al número cero (0), mientras que cualquier valor numérico diferente al anterior es equivalente a ***true***.

if (esCorrecto == true) ....

if (esCorrecto == true) ser a equivalente a if (esCorrecto).

Al hacer referencia a la variable sin operador de comparación se entiende que su valor es true.

**Variables**

Para la declaración de una variable local JavaScript utiliza la palabra reservada **var**. Podemos declarar varias variables en una sola línea separándolas por comas.

Ejem: **var curso**;

Ejem: **var curso, temas**;

Las variables para poder utilizarlas hay que inicializarlas, tenemos dos tipos de inicialización: en la propia declaración o posteriormente.

var miCadena = "Hola mundo";

var miCadena;

miCadena = " Hola mundo";

El alcance de una variable puede ser **local** o **global**. Si la variable está declarada dentro de una función su alcance ser dentro de la función, si queremos que su alcance sea global debemos declarar la variable fuera de cualquier función.

Para la declaración de las variables hay restricciones de nombres como son las siguientes:

* El primer carácter debe de ser una letra o un subrayado "**\_**". No se permite comenzar una variable por un valor numérico o símbolo.
* El resto de caracteres no tiene ninguna restricción exceptuando símbolos considerados no válidos por JavaScript. Ejem: **var variable#;**
* Los espacios no son válidos. Ejem: var mi variable; Ejem correcto: **var mi\_variable;**
* No se permite la utilización de palabras reservadas por el lenguaje. Ejem: **var while;**

**2.2. Operadores**

Podemos realizar diversos cálculos aritméticos, lógicos y de comparación mediante los operadores. Vamos a ver los más utilizados.

**Aritméticos:**

|  |  |
| --- | --- |
| Operador | Descripción |
| + | Suma dos números. var suma = 5 + 2; |
| - | Resta dos números var resta = 5 - 2; |
| \* | Multiplica dos números var multiplica = 5 \* 2; |
| / | Divide dos números var división = 5 / 2; |
| % | Devuelve el resto de dividir ambos números. 5%2 = 1 |
| ++ | Suma 1 al contenido de una variable. var x = 1; x++; El valor de x es 2. |
| -- | Resta 1 al contenido de una variable. var x = 2; x--; El valor de x es 1. |

**Asignación y comparación:**

|  |  |
| --- | --- |
| **Operador** | **Descripción** |
| **= Operador de asignación** var x = 5; | Asigna a x el valor 5. |
| += -= \*= /=. | Añade el valor del operando de la derecha a la variable de la izquierda  Ejemplo: x +=100 es igual que hacer x = x + 100; |
| **== " Igual a"** | devuelve true si los operandos son iguales |
| **!= " No igual a"** | devuelve true si los operandos no son iguales |
| **> " Mayor que"** | devuelve true si el operador de la izquierda es mayor que el de la derecha. |
| **>= " Mayor o igual que "** | devuelve true si el operador de la izquierda es mayor o igual que el de la derecha. |
| < " Menor que" | devuelve true si el operador de la izquierda es menor que el de la derecha. |
| <= "Menor o igual que" | igual que el anterior pero además devuelve true si es igual que derecha. |
| **&& Conjunción Lógica (AND).** | Si ambos operadores son true, la instrucción será cierta. Ejem: 3 + 6 = 9 && 4 +2 = 6. Ambos operadores son ciertos, con lo que la instrucción será cierta. 3 + 5 = 2 && 4 + 5 = 9. Ambos operadores no son ciertos por lo que se devolver un false. |
| **|| Disyunción Lógica (OR)**. | en el otro operador. Ejem: 3 + 6 = 9 || 2 + 3 = 1. Un operador es válido por lo que la instrucción es correcta y devolver un true. |
| **! "No"** | Devuelve true si la negación del operando es true. Ejem: var x = true; if (!x = true). Esta instrucción devolver false, ya que la negacin de x es false y no true. |

**2.3. Condicionales**

Mediante las diferentes sentencias de control vamos a podemos controlar la lógica de la programación.

Las funciones nos ayudarán a agrupar líneas de código que realizan una tarea concreta y que podremos reutilizar fácilmente.

**Simple y Compuestas**

JavaScript tiene una instrucción condicional que es if...else.

Se evalúa la condición primera, en caso de que se cumpla o no acceder a las instrucciones que especifiquemos.

Ejemplo:

var x = 3;

If (x !=3)

Instruccion1;

Else

instrucción2;

En el caso de querer introducir n instrucciones en cada condición, se debe introducir las instrucciones entre llaves '{...}' se proceder de la siguiente forma:

var x = 3;

if (x !=3)

{

Instruccion1;

Instruccion2;

}

else

{

Instruccion3;

Instruccion4;

}

La cláusula else no es obligatoria, sólo la pondremos en el caso de querer ejecutar código cuando no se cumpla una condición inicial.

Se podrán anidar varios if de la siguiente forma:

if (x !=3)

{

Instruccion1;

Instruccion2;

}

else if (x > 4)

{

Instruccion3;

Instruccion4;

}

**Multivariable**

Si se requiere utilizar varios if anidados es recomendable utilizar switch.

var x = 3;

switch (x)

{

case 1:

Instruccion1;

break;

case 2:

Instruccion2;

break;

case 3:

Instruccion3;

break;

default:

Instruccion4;

break;

}

La instrucción **break** es necesaria para terminar la instrucción switch, en caso de no ponerlo se ejecutará todo el código del switch. La opción default es opcional para el caso en que no se cumpla alguna condición ejecutar el código deseado.

**2.4. Iteración**

Existen distintos tipos de iteración.

**Bucles for**

var x;

for (x=1; x < 10 ; x ++)

{

Instruccion 1;

}

Se inicializa la variable x a 1, y se evalúa la segunda condición, en el caso de cumplirse, se aumenta en una unidad la variable x y se ejecuta Instruccion 1, y así sucesivamente.

Instruccion 1 se ejecuta 9 veces en este caso.

**Bucles while**

var x=1;

var correcto = true;

while (correcto)

{

Instruccion 1;

x++;

if (x < 10)

correcto= false;

}

Este bucle es similar al bucle for, la diferencia radica en que el contador y la actualización de la misma para el control de la salida de la condición del bucle se realiza dentro del bucle.

En el ejemplo mientras correcto sea true se ejecutarán las Instrucciones.

**Bucles do..while**

Este bucle es idéntico al anterior con la salvedad de que la condición se evalúa al final del bucle.

var x=1;

var correcto = true;

do

{

Instruccion 1;

x++;

if (x >9)

correcto= false;

}while (correcto);

Si no queremos que un bucle realice todas las iteraciones hasta cumplir la condición podemos utilizar la instrucción **break** para salir de un bucle.

Hay que tener especial cuidado para evitar bucles que no cumplan la condición de terminación y así evitar cuelgues del sistema.

var x;

for (x=1;x<10;x--)

{

Instrucción 1;

}

En este ejemplo el bucle se ejecutar a infinitamente, dado que al iniciarse la variable en uno e ir decreciendo nunca llegará a ser superior a 10.

**2.5. Funciones**

Una función en JavaScript es un conjunto de instrucciones a las que se asigna un nombre.

**Definición de funciones:**

De forma general, la sintaxis de una función es la siguiente:

function nombreDeFuncion (argumentos)

{

instrucciones de la función;

}

Ejemplo:

function importe\_IVA (importe)

{

var resultado;

resultado = importe \* 0.18;

}

Si una función recibe más de un parámetro, éstos se separan por una coma '**,**'.

function importe\_IVA (importe, tipo\_iva)

{

if (tipo\_iva == 'empresas')

resultado = importe \* 0.18;

else if (tipo\_iva == 'hoteles')

resultado = importe \* 0.07;

else

resultado = 0;

}

Una vez definida la función podremos llamarla de la siguiente forma:

importe\_IVA(500);

importe\_IVA(500, 'empresas');

La función en nuestro ejemplo no realiza nada con el resultado, lo normal será mostrarlo en un mensaje o retornarlo para guardarlo en una variable para su posterior tratamiento. Para retornar un valor se utiliza la instrucción **return.**

function importe\_IVA (importe, tipo\_iva)

{

if (tipo\_iva == 'empresas')

resultado = importe \* 0.18;

else if (tipo\_iva == 'hoteles')

resultado = importe \* 0.07;

else

resultado = 0;

return resultado;

}

Para llamar a una función que retorne datos se realiza de la siguiente forma:

var importe1, importe2;

importe1 = importe\_IVA(500);

importe2 = importe\_IVA(500, 'empresas');

Una función puede tener parámetros opcionales, para ello se declara el parámetro opcional entre corchetes.

function importe\_IVA (importe[, tipo\_iva])

{

if (tipo\_iva == 'empresas')

resultado = importe \* 0.18;

else if (tipo\_iva == 'hoteles')

resultado = importe \* 0.07;

else

resultado = 0;

return resultado;

}

Para llamar a esta función se podrá incluir el tipo\_iva o no incluirlo. En caso de no incluirlo devolverá el resultado = 0.

**Funciones que no están asociadas a objetos**

Existen funciones disponibles en el lenguaje que no están asociadas a ningún objeto, estas funciones son las siguientes:

* **parseInt:** se suele utilizar para convertir una cadena de caracteres a número. La función intenta convertir el string en número y si no puede devuelve **NaN** (Not a Number) que indica que no es un valor numérico.
* **isNaN:** devuelve un boolean dependiendo de si el parámetro es un número o no.
* **eval:** función que recibe una cadena de caracteres y la ejecuta como si fuera una sentencia JavaScript.
* **parseFloat:** convierte la cadena en un número flotante y lo devuelve.

**3. Objetos del Lenguaje**

Una utilidad JavaScript es mostrar mensajes interactivos para comunicar información al usuario o pedir datos necesarios para la ejecución del código.

En este apartado vamos a hablar de los objetos del lenguaje más utilizados, concretamente del objeto String, Math y Date.

**3.1. Cadenas**

* **Propiedad length:**Valor numérico que nos devuelve la longitud de caracteres de una cadena que se le manda como parámetro.

var cadena = 'Hola Mundo';

var final;

final = cadena.length;

En el ejemplo el resultado de final ser 10, ya que la longitud de caracteres del texto 'Hola Mundo' suma 10.

* **concat:** El método concat une dos cadenas en una sola.

var texto1 = "Hola ";

var text2 = "Mundo";

texto1 = texto1.concat(texto2);

El resultado seria "Hola Mundo". La asignación del texto final se podría haber realizado en una nueva variable, pero también es posible reutilizar las variables ya declaradas.

* **indexOf:** Este método busca la coincidencia de una cadena en otra. Si la encuentra devuelve la posición de comienzo de la cadena, si no se encuentra este método devuelve -1.

texto.indexOf( cadena\_busqueda [, indiceInicio] );

El parámetro indiceInicio contendrá la posición de la cadena desde la que se empezar la búsqueda.

var texto1 = "Hola Mundo";

var posición;

posicion = texto1.indexOf('M');

**posicion** obtendrá un resultado de 5, que es la posición en la que se encuentra la letra M. Las posiciones en las cadenas comienzan en 0.

* **lastIndexOf:** Este método es similar a **indexOf**, pero devuelve la posición de la última coincidencia de la cadena a buscar, si la encuentra devuelve la posición, sino devuelve -1.

texto.lastIndexOf( cadena\_busqueda [, indiceInicio] );

var texto1 = "Hola Mundo";

var posición;

posicion = texto1.lastIndexOf('o');

**posicion** obtendrá un resultado de 9, que es la posición de la última letra 'o'.

* **substr:** Este método devuelve una subcadena a partir del objeto actual. objetoString.substr( inicio [, longitud] ); Si no se indica el parámetro de longitud se tomar hasta el final de la cadena.

var texto1 = "Hola Mundo";

var texto2;

texto2 = texto1.substr(1,5);

El resultado de texto2 ser "ola M".

* **toUpperCase:** Mediante este método convertimos una cadena al completo a mayúsculas.

var texto1 = "Hola Mundo";

var texto2 ;

texto2 = texto1.toUpperCase();

El resultado de texto ser "HOLA MUNDO".

* **toLowerCase:** Mediante este método convertimos una cadena al completo a minúsculas.

var texto1 = "Hola Mundo";

var texto2 ;

texto2 = texto1.toLowerCase();

El resultado de texto ser "**hola mundo**".

**3.2. Matemáticas**

A continuación procederemos a explicar los métodos del objeto Math.

* **abs(numero).** Devuelve el valor absoluto de un número. Math.abs(numero1);
* **cos(numero).** Devuelve el coseno de un número. Math.cos(numero1);
* **sin(numero).** Devuelve el seno de un número. Math.sin(numero1);
* **tan(numero).** Devuelve la tangente de un número. Math.tan(numero1);
* **acos(numero).** Devuelve el arcocoseno de un número. Math.acos(numero1);
* **asin(numero).** Devuelve el arcoseno de un número. Math.asin(numero1);
* **atan(numero).** Devuelve el arcotangente de un número. Math.atan(numero1);
* **sqrt(numero).** Devuelve la raiz cuadrada de un número. Math.sqrt(numero1);
* **round(numero).** Devuelve el entero más cercano resultante de redondear número por el sitio más cercano.
* **ceil(numero).** Devuelve el entero obtenido de redondear número por arriba.
* **floor(numero).** Devuelve el entero obtenido de redondear número por abajo.
* **max(numero1,numero2).** Devuelve el número mayor entre los dos.
* **min(numero1,numero2).** Devuelve el número menor entre los dos.

**3.3. De Fecha y Hora**

El objeto **Date** nos va a permitir manipular las fechas. Antes de entrar en los métodos del objeto conviene hacer unas aclaraciones.

Los meses de Enero a Diciembre vienen dados por un número entero entre 0 y 11, correspondiendo el 0 a Enero.

Los días de la semana vienen dados de Domingo a Sábado por un número entero entre 0 y 6, correspondiendo el 0 al Domingo.

Las horas del día vienen dadas por un número entero entre 0 y 23.

Los objetos Date se pueden crear vacíos o con una fecha por defecto. Si se crean vacíos tomará el valor de la fecha actual del sistema.

var fecha = new Date();

Si ponemos un valor iniciaremos Date con una fecha, hay varias formas de hacerlo:

var fecha = new Date(ao, mes);

var fecha = new Date (a o, mes, da);

var fecha = new Date (a o, mes, da, horas);

var fecha = new Date (a o, mes, da, horas, minutos);

var fecha = new Date (a o, mes, da, horas, minutos, segundos);

* **getDate():** Devuelve el día del mes actual entre 1 y 31.
* **getMonth():** Devuelve el mes actual entre 0 y 11. Se recomienda siempre que se use getMonth sumarle 1 al valor que retorne para así tener los meses en formato español.
* **getFullYear():** Devuelve el año actual.

var fecha=new Date();

var dia;

var mes;

var anio;

dia = fecha.getDate();

mes=fecha.getMonth() + 1;

anio=fecha.getFullYear();

* **getHours():** Devuelve la hora actual en un intervalo entre 0 y 23.
* **getMinutes():** Devuelve los minutos de la hora actual en un intervalo entre 0 y 59.
* **getSeconds():** Devuelve los segundos de la hora actual en un intervalo entre 0 y 59.

var fecha = new Date();

var hora;

var minutos;

var segundos;

hora = fecha.getHours();

minutos=fecha.getMinutes();

segundos=fecha.getSeconds();

* **getDay():** Devuelve el día de la semana de la fecha actual en un intervalo de 0 a 6. El valor 0 es el Domingo.

**3.4. Windows**

En este apartado vamos a ver los métodos alert, confirm y prompt que pertenecen al objeto window, así como la forma de realizar comentarios en JavaScript.

Para realizar un comentario en JavaScript se pueden usar dos formas

// esto es un comentario de una sola linea

/\*

Esto es un comentario de más De una línea.

\*/

Los comentarios son muy útiles para explicar nuestro código para que

en un futuro u otra persona puedan comprenderlo rápidamente.

Como hemos comentado los métodos alert, confirm y prompt

pertenecen al objeto window, pero al invocarlos no es necesario poner

window.alert, ya que por defecto se entiende que estamos en el objeto

window.

* **alert:** Cuadro de mensaje de alerta. Se podrá poner cualquier texto como mensaje siempre entrecomillado.

var numero=1;

alert("El valor de numero es:" || numero);

// texto estático y concatenamos el valor de una variable

****

* **confirm:** Cuadro para aceptar una pregunta formulada al usuario.



El método **confirm** devolverá un true si se ha pulsado el botn Aceptar, y un false si se ha pulsado Cancelar.

If (confirm("Desea guardar los datos") )

{

alert("ha pulsado aceptar");

}

else

{

alert("ha pulsado cancelar");

}

* **prompt:**

El método prompt se utiliza para solicitar datos al usuario. Su estructura es la siguiente:

window.prompt( pregunta [, respPredeterminada]);

El parámetro opcional de respuesta predeterminada cargar un valor por defecto en la respuesta que debe dar el usuario. Si no se pone un valor por defecto, aparecer cargado el valor "undefined".



var respuesta;

var respuesta2;

respuesta = prompt("Introduzca su edad:");

respuesta2 = prompt("Introduzca el precio del articulo:", 20);

**3.5. Navegador**

Existe gran cantidad de objetos relacionados con el Navegador. En este curso explicaremos los objetos más importantes.

**Window:** El objeto window hace referencia a la ventana actual. Vamos a ver sus principales propiedades y métodos.

**Propiedades**

* **closed:** Devuelve un boolean que nos que nos indicar si la ventana está abierta closed = false o si está cerrada closed=true.
* **location:** Devuelve la dirección url de la barra de dirección.
* **navigator:** Representa el navegador que estamos utilizando.
* **status:** String con el mensaje que contiene la barra de estado.
* **opener:** Representa el objeto que abre la ventana.

**Métodos**

* **close():** Cierra la ventana actual.
* **navigate():** Redirecciona el navegador a una URL indicada.
* **open();** open(URL,nombre,caracteristicas). Abre la URL indicada, si no existe crea una nueva ventana, siempre con el nombre que se le pase como parámetro. Las características que se le pueden dar a este método son muy extensas, en cuanto a posición de la ventana, ancho,alto, etc.

**Document**

Este objeto nos permite acceder a las propiedades de una página HTML. Nos centraremos en su propiedad **forms**, que contiene una matriz de los formularios existentes en el HTML. Accediendo a un formulario, podremos acceder a todos los campos que se creen en el mismo, y asignarle

valores, preguntar su longitud, etc.

Para acceder a los valores de los campos de un formulario escribiramos:

**document.nombre\_del\_formulario.nombre\_del\_campo.value**

Para obtener el valor del formulario form1 y del campo nombre en un variable haríamos:

**nom = document.form1.nombre.value;**

**Propiedades**

* **action:** Representa la direccin URL que es llamada por nuestro formulario.

**Métodos**

* **submit():** Se lleva a cabo cuando se envía el formulario.

**4. Dónde incluimos JavaScript?**

A la hora de escribir código JavaScript podemos optar por las siguientes opciones:

* En una página html.
* En un fichero .js externo.
* En un evento de etiqueta html.

**En página HTML:**

Podemos incluir código Javascript en cualquier documento html o en aquel que termine traduciéndose a html como: asp, jsp, php...

En un documento html el código debe incluirse entre etiquetas especiales html para indicar que se trata de un código script.

Estas etiquetas son las siguientes:

<script type="text/javascript">

// código JavaScript

</script>

La etiqueta script puede tener varios atributos como type language.

Se recomienda que se sitúe en la sección Head de la página. Todo el código incluido en esta sección es interpretado por el navegador cuando se carga la página.

**En un fichero externo:**

Podemos incluir un código Javascript desde otro archivo y utilizar las funciones incluidas dentro de nuestro código, en tantos archivos como queramos. De esta forma podemos reutilizar el código todo lo necesario.

El atributo **src** permite especificar el archivo que contiene el código Javascript.

<script language="Javascript" src="saludar.js"></script>

El fichero saludar.js podrá contener una función como la siguiente:

function saludo(nombre) {

alert('Hola, ' + nombre);

}

**En eventos de etiquetas Html:**

Las etiquetas html tienen asociados eventos que responden a determinados sucesos como hacer clic, cargar una página, enviar un formulario, ...

Podemos incluir en estos eventos código JavaScript como el siguiente:

<body onload=" alert('Texto al cargar la página web')">