



Tema 2: JavaScript.

1. Introducción.
2. Sintaxis.
3. Cadenas, Matrices y Objetos.
4. Eventos.
5. Modelo de Objetos.
6. Aplicaciones.




1. Introducción.

Lenguaje interpretado.

- Creado por Netscape para añadir interactividad a los documentos HTML.
- Es un lenguaje interpretado (por el browser), embebido dentro del documento HTML.
- La sintaxis es similar a la del C++ y Java, aunque bastante más relajada:
 - P.e: El ";" es recomendable, pero no obligatorio.

1. Introducción.


Versiones.



■ Navigator 2.0	JavaScript 1.0
■ Internet Explorer 3.0	JScript 1 (ECMAScript v.1)
■ Navigator 3.0	JavaScript 1.1 (ECMAScript v.1)
■ Internet Explorer 4.0	JScript 3 (ECMAScript v.2)
■ Navigator 4.0	JavaScript 1.2
■ Navigator 4.06	JavaScript 1.3 (ECMAScript v.2)
■ Internet Explorer 5.0	JScript 5
■ Navigator 5.0	JavaScript 1.4
■ Internet Explorer 5.5	JScript 5.5 (ECMAScript v.3)
■ Navigator 6.0	JavaScript 1.5 (ECMAScript v.3)

1. Introducción.

Insertar código JavaScript (I).

- 
- Escribir el código dentro de la página, utilizando la directiva **<SCRIPT>** .. **</SCRIPT>**.

```
<html>
<head> ...
<script language="javascript"> (o <script type="text/javascript">)
<!--
function EstaVacio(objeto) {
    if ( !objeto.value.length ) {
        alert("Este campo no puede estar vacío");
        objeto.focus();
    }
}
// -->
</script>
</head>
<body>
....
```

1. Introducción.

Insertar código JavaScript (II).

- En un fichero separado:

```
<script src="programa.js"></script>
```

- Como valor de un atributo de tipo evento:

```
<body OnUnLoad="alert('Hasta Luego!')">
```

- Como destino de un hipervínculo:

```
<a href="javascript:window.open('hist.html',  
'historia', 'width=600, height=500')">
```

Un poco de historia

2. Sintaxis.

Comentarios.

- Los comentarios al código se realizan del mismo modo que en el C, C++ y Java.

```
<script language=JavaScript>  
<!--  
// Esto es un comentario.  
function EstaVacio(objeto)  
{  
    /* Esto es otro comentario  
       que ocupa varias líneas */  
    if ( !objeto.value.length )  
    {  
        alert("Este campo no puede estar vacío");  
        objeto.focus();  
    }  
}  
// -->  
</script>
```

2. Sintaxis.

Declaración de variables.

- Para declarar una variable se utiliza la palabra var (la declaración no es obligatoria, aunque si aconsejable):

```
var i = 5.5, j;  
var nombre="Juan Marin";  
k = i + j + 3; // OK.  
i = w + 3; // Problemas.
```

- El nombre de la variable empieza por una letra o por el símbolo '_', y esta formado por caracteres alfanuméricos (es sensible al contexto).

```
var obj_3d, Obj_3D; // Son variables diferentes.
```

- Las variables no tiene un tipo fijo:

```
var i = 3; i = 5.5; i = "Juan"; // Sería valido.
```

2. Sintaxis.

Tipos de datos.

- 5 tipos básicos:

- Numérico: Enteros y reales.

```
var i = 3;
```

- Booleano: true y false (convertibles a los valores 1 y 0).

- Cadena: delimitadas por comillas dobles o simples.

```
var cad1 = "Juan", cad2='Pepe', cad3 = cad1 + ' Marín';
```

- Objetos: `var obj1 = new Object();`

- Función: `var g; g = EstaVacio;`

- Array: `var m = new Array();`

- Fecha: `var m = new Date(2002, 1, 28);`

- Nulo: `cad1 = null;`

2. Sintaxis.

Operadores.

- Aritméticos: ++, --, +, -, *, /, %, \ (div).
- Lógicos: &&, ||, !
- Relacionales: ==, !=, <, <=, >, >=, === (identidad), !==
- Bit a bit: &, |, ^, ~, <<, >>
- Asignación: =, +=, -=, *=, /=, %=, &=, |=
- Otros: ? :, ., new, delete, typeof, void.

Ejemplos:

```
if ("1" == true ) alert("son iguales");  
if ("1" !== true ) alert("no son identicos");
```

2. Sintaxis.

Estructuras de control de flujo.

- *if*(cond) { .. } *else* { .. }.
- *switch* (valor){ *case*: .. }
- *while* (cond) { .. }
- *do* { .. } *while* (cond);
- *for* (valor inicial; cond; incr) { .. }
- *for-in*, para enumeraciones.

Ejemplo:

```
var test = new Object();  
test.camp1 ="Campo uno"; test.camp2 ="Campo dos";  
test.camp3 ="Campo tres";  
var a;  
for ( a in test )  
    document.write( a + " = " + test[a] + "<br>");
```

2. Sintaxis.

Funciones (I).

- Para la definición se utiliza la palabra reservada "function":

```
function nombre_func ( a, b ) { ... }
```

- El paso de parámetros es siempre por valor.
- Devuelve valores a través de la sentencia *return*
- Las variables definidas dentro de la función tiene ámbito local. Cualquier variable definida fuera de una función tiene un ámbito global.
- Las variables globales son accesibles desde otros frames.

2. Sintaxis.

Funciones (II).

- Ejemplo de variables globales compartidas entre frames distintos:

```
<html> <frameset cols="*,*">
  <frame name="frame_izq" src="doc1.htm">
  <frame name="frame_der" src="doc2.htm">
</frameset> </html>
```

```
-----doc1.html-----
<script language="JavaScript">
var v_izquierda = 0;
function f_izquierda() {
  alert ('valor =' + v_izquierda );
}
</script>
```

```
-----doc2.html-----
<script language="JavaScript">
top.frame_izq.v_izquierda ++;
top.frame_izq.f_izquierda();
</script>
```

2. Sintaxis.

Funciones (III).

■ Existen varias formas de definir funciones:

■ De forma similar al C:

```
function menor(x,y) {  
    if (x<y) return x;  
    else return y;  
}  
document.write(menor(5,6));
```

■ Utilizando el constructor *Function* (creadas de forma dinámica):

```
var f = new Function("x", "y", "if(x<y) return x; else return y;");  
document.write(f(15,9));
```

■ Mezclando las dos anteriores (estática):

```
var g = function(x,y) { if(x<y) return x; else return y; }  
document.write(g(23,29));
```

3. Objetos: String, Array y Object.

Cadenas (I).

■ Delimitadas por comillas dobles o sencillas.

```
var nom1="Juan ", apel='Marin';
```

■ Concatenadas con el operado '+'.

```
nombre_completo = nom1 + apel;
```

■ Constructor: **String()**

```
var cad = new String();
```

■ Propiedades: **length**.

```
longitud = ("Juan Morillo").length + nom1.length;
```

■ Métodos:

- **charAt** (pos): Carácter de la cadena situado en una posición dada.

```
letra = "javascript".charAt(4);
```

3. Objetos: String, Array y Object.

Cadenas (II).

- **indexOf** (subcad [,pos_ini]): Posición de comienzo de una subcadena dentro de una cadena:

```
pos = nombre_completo.indexOf("an");
```
- **substr** (pos_ini, longitud): Subcadena dentro de una cadena.

```
cadena = "javascript".substr(4,6);
```
- **toLowerCase()** y **toUpperCase()** : Convierte la cadena a minúsculas y a mayúsculas respectivamente.

```
if ("JavaScript".toUpperCase() == "javascript".toUpperCase() )  
    document.write('<p>Son iguales.');
```

3. Objetos: String, Array y Object.

Arrays (I).

- Son dinámicos (su tamaño puede cambiar en tiempo de ejecución) y heterogéneos (el tipo de sus elementos puede ser diferente).
- Constructor: **Array()**.

```
var vec1 = new Array(); vec1 = [ 4, 7, "hola", true];  
var vec2 = new Array( 2, "Pepe", 13.56);
```
- Son dispersos: reserva espacio sólo para las posiciones ocupadas.

```
var v = new Array(); v[100]="Hola"; v[100000000]=2.8;
```
- Arrays de varias dimensiones:

```
var Matriz = new Array();  
for ( i = 0; i<5; i ++ )  
    Matriz[i] = new Array();  
Matriz[3][4] = 45;
```


3. Objetos: String, Array y Object.

Arrays (II).

- Propiedades: **length** (Índice del último elemento del array):

```
ultimo = v.length;
```

- Métodos:

- **concat** (segundo array). Concatena arrays:

```
var v1 = new Array(1,24), v2 = new Array(6,7);  
v = v1.concat(v2);
```

- **join** ([delimitador]). Agrupa los elementos en una cadena.

```
alert(v.join('-'));
```

- **reverse** (). Invierte el array:

```
v.reverse();
```

- **sort** ([función]). Ordena el array:

```
v.sort();
```

3. Objetos: String, Array y Object.

Objetos (I).

- No existe el concepto de clase propiamente dicho, ni el de herencia.

- Crear objetos:

- Constructor genérico **Object()**:

```
var obj = new Object();  
obj.exist = 3; obj.stock = true; obj.estado = "Vend.";
```

- Dando una lista de propiedades:

```
var obj = {exist:3, stock:true, estado:"Vendidos" };
```

- Empleando constructores definidos por el usuario:

```
function Articulo( exi, sto, est) {  
    this.exist = exi; this.stock = sto; this.estado = est;  
}  
var obj = new Articulo(3, true, "Vendidos");
```

3. Objetos: String, Array y Object.

Objetos (II).

- Para acceder a las propiedades del objeto se utiliza el operador ".", o el operador "[]":

```
obj.exist = 45; // Ambas sentencias son equivalentes
obj["exist"] = 45;
```

- Las propiedades y los métodos del objeto se añaden dinámicamente, sin necesidad de ser definidas con antelación:

```
function Resumen() {
    var cad = "";
    for (prop in this)
        cad += this[prop].toString() + "\n";
    return cad;
}
obj.precio = 12.4; obj.extracto = Resumen;
alert(obj.extracto());
```

3. Objetos: String, Array y Object.

Objetos (III).

- Las propiedades y los métodos pueden ser eliminados dinámicamente con el operador delete.

```
delete obj.valor;
alert(obj.extracto());
```

- Con las propiedades y los métodos prototipo (**prototype**) se pueden añadir de forma dinámica propiedades y métodos al constructor.

```
function Valor() { return this.precio * this.exist; }
...
var a1 = new Articulo(2,true,"Vendidos");
var a2 = new Articulo(4,true,"Vendidos");
Articulo.prototype.precio = 1.0;
Articulo.prototype.valor = Valor;
a2.precio = 15;
alert('Total:' + (a1.valor() + a2.valor()));
```

4. Eventos.

Manejadores de Eventos.

- JavaScript es un lenguaje orientado a eventos: El código javascript se resume en un conjunto de subprogramas ejecutados tras activar el evento correspondiente.
- A estas rutinas javascript, ejecutadas al activar un evento, se les llama "manejadores del evento".
- La gestión de los eventos la realiza el browser: Activa el manejador del evento automáticamente al producirse dicho evento.
- El manejador del evento se definen a través de:
 - Atributos de las directivas HTML:
`<BODY onUnload="alert('Hasta Luego');">`
 - Propiedades del objeto:
`window.onunload=function() {alert('Hasta Luego');}`

4. Eventos.

Eventos más importantes (I).

- **onClick**: Click del ratón.
 - Objetos: **link** y botones del formulario.
 - Si el manejador devuelve *false* se cancela la acción.

```
<script language="javascript">
function Aviso()
{
    var mensaje="El contenido de la página podría herir "+
                "la sensibilidad.\n"+
                "¿Seguro que deseas continuar?";
    return confirm (mensaje);
}
</script>
....
<a href="http://informatica.uv.es/fcotomas/"
onClick="return Aviso();">Propaganda electoral</a>
```

4. Eventos.

Eventos más importantes (II).

- **onMouseOver / OnMouseOut:** El ratón entra/sale del elemento.
 - Objetos: **link, image, area**.

```
<a href="http://www.uv.es/" onMouseOver="window.status='Pagina de la universidad'; return true;"
onMouseOut="window.status='';">Univ.</a>
```
- **onFocus / OnBlur:** El objeto toma/pierde el foco.
 - Objetos: **window** y todos los elementos del formulario.

```
<body onFocus="document.bgColor='white'"
onBlur="document.bgColor='lightgrey'">
```
- **onLoad / OnUnload:** El documento ha terminado la carga (*onLoad*) / está a punto de descargarse (*onUnload*).
 - Objetos: **window**.

4. Eventos.

Eventos más importantes (III).

- **onSubmit / OnReset:** ha sido pulsado el botón de *submit/reset* del formulario. Devolviendo *false* se anula la acción del botón.

```
<script language="javascript">
function ValidarDatos()
{
    var OK;
    ...
    if ( !OK ) return false;
    else return true;
}
</script>
...
<form name="miformulario" action="../../../cgi-bin/prog.cgi"
      method="POST" onSubmit="ValidarDatos();">
...
</form>
```

4. Eventos.

Eventos más importantes (IV).

- **onResize**: Cambia el tamaño de la ventana.
 - Objetos: **window**.
- **onChange / onSelect** : Cambia/selecciona el texto.
 - Objetos: entrada de texto, **textarea**.
- **onKeyDown / onKeyPress / onKeyUp**: Pulsa una tecla / mantiene pulsada una tecla / suelta una tecla.

5. Modelo de Objetos.

Objeto Date (I).

- **Date**: Objeto de tipo fecha y hora.
- **Constructor**:
 - **Date()**:

```
var hoy = Date(); // guarda: Thu Mar 7 13:01:00 UTC+0100 2002
```
 - **Date("mes día, año hora:minutos:segundos")**:

```
var f1 = new Date("Mar 17, 2002 21:01");
```
 - **Date(año, mes, día, [hora, [minuto, [segundos, [miliseg]]]]]**:

```
var f1 = new Date( 2002, 2, 17, 21, 01);
```
- **Métodos**:
 - **getDate()** / **setDate(díaDelMes)**: recupera/fija el día del mes.
 - **getDay()** / **setDay(díaDeLaSemana)** : recupera/fija el día de la semana.

5. Modelo de Objetos.

Objeto Date (II).

- **getHours()** / **setHours(hora)** : recupera/fija la hora del día.
- **getMinutes()** / **setMinutes (hora)** : recupera/fija los minutos de la hora.
- **getMonth()** / **setMonth(mes)** : recupera/fija el mes del año.
- **getSeconds()** / **setSeconds (segundos)** : recupera/fija los segundos del minuto.
- **getTime()** / **setTime (hora)** : recupera/fija los milisegundos pasados desde el 1 de Enero de 1970.
- **getFullYear()** / **setFullYear (año)** : recupera/fija el año completo (empleando cuatro dígitos).

```
var f = new Date();  
var aleatorio_0_99 = getTime()%100;  
f.setDate(17); f.setHours(21); f.setMinutes(1);  
f.setMonth(2); f.setFullYear(2002);
```

5. Modelo de Objetos.

Objeto Math.

- Las funciones matemáticas están disponibles sólo a partir del objeto **Math**.
- Propiedades:
E , **LN10** , **LN2** , **LOG10E** , **LOG2E** , **PI** , **SQRT1_2** , **SQRT2**.

```
opera = Math.E * Math.PI / Math.SQRT2;
```

- Métodos:
abs(num), **ceil(num)**, **exp(num)**, **floor(num)**, **log(num)**,
max(num1,num2), **min(num1,num2)**, **pow(basem, expon)**,
random(), **round(num)**, **sqrt(num)**, **acos(num)**, **asin(num)**,
atan(num), **atan2(x,y)**, **cos(num)**, **sin(num)**, **tan(num)**.

```
opera = Math.cos(Math.acos(0));
```

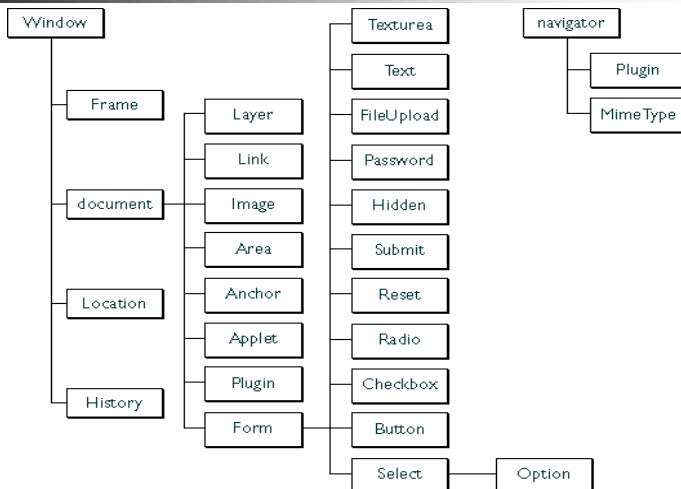
5. Modelo de Objetos.

Objetos del Browser (I).

- El javascript proporciona una serie de objetos predefinidos que permiten interactuar con el browser y con los documentos HTML visualizados.
- Existen algunas diferencias entre el modelo de objetos de Netscape y el de Internet Explorer.
- Estos objetos mantienen una estructura jerárquica.
- *window* es el objeto principal.

5. Modelo de Objetos.

Objetos del Browser (II).



5. Modelo de Objetos.

Objeto navigator.

- Contiene información sobre el propio browser.
- Propiedades:
 - **appName**: nombre común del Navegador.
`alert(navigator.appName); // visualiza: Netscape`
 - **appVersion**: versión del browser.
`alert(navigator.appName);`
`// visualiza: 5.0(Windows; es-ES)`
 - **userAgent**: información que envía el browser al servidor.
p.e: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)

5. Modelo de Objetos.

Objeto window (I).

- **window**: ventana del browser. Objeto padre de los objetos *document*, *location* y *history*.
- Propiedades:
 - **frames**: Array de frames.
 - **length**: Número de frames.
 - **name**: nombre de la ventana (creada p.e. con *open()*).
 - **parent**: ventana padre.
 - **top**: ventana del nivel superior.
 - **status**: texto de la barra de estado (para un mensaje temporal).
 - **defaultStatus**: Texto por defecto de la barra de estado.

5. Modelo de Objetos.

Objeto window (II).

■ Métodos:

- **alert**(mensaje): abre una ventana con un mensaje de alerta.
- **confirm**(mensaje): similar.
- **prompt**(mensaje, [Texto por defecto]) : Entrada de texto en una ventana de dialogo.

```
var g= window.prompt('Introduce un texto');  
document.write('<br> Has introducido ' + g);
```
- **open**(url, nombre, [propiedades]): Abre una ventana.

```
var newwin = window.open("http://www.uv.es", "nueva",  
"width=400, height=300");
```
- **close**(): Cierra la ventana.

```
newwin.close();
```
- **setTimeout**(codigo,delay): Ejecuta el código javascript indicado en el primer parámetro (codigo) cuando hallan pasado los milisegundos fijados en el 2º parámetro (delay).

5. Modelo de Objetos.

Objeto location.

■ Url de la ventana.

■ Propiedades:

- **hash**: porción de la url que sigue a #.
- **hostname**: nombre de la máquina de la url.
- **href**: contenido total de la url.
- **protocol**: protocolo de la url.
- **search**: porción de la url que sigue a ?.
- **pathname**: camino.
- **port**: puerto.

■ Métodos:

- **reload**(): recarga el documento:

```
<body onLoad="setTimeout('window.location.reload()',50000);">
```
- **replace**(nueva url): Remplaza el actual documento.

5. Modelo de Objetos.

Objeto history.

- Historial de la páginas visitadas. Se guarda en forma de array.
- Propiedades:
 - **length**: longitud del historial.
- Métodos:
 - **back()**: Carga el url anterior.
`window.history.back();`
 - **forward()**: Carga el url anterior.
 - **go(n)**: carga el url situado dentro del array n lugares a la derecha (si n es positivo), o n lugares a la izquierda (si n es negativo), tomando como referencia la posición del url actual:
`window.history.go(-2);`

5. Modelo de Objetos.

Objeto document (I).

- Representa al propio documento visualizado en el browser.
- Propiedades:
 - **title**: título del documento.
 - **bgColor** y **fgColor**: Color del fondo y del texto del documento.
 - **images**: Array de imágenes incluidas dentro del documento.
 - **forms**: Array de formularios.
 - **links**: Array de links.
 - **linkColor**, **alinkColor**, **vlinkColor**: Colores de los enlaces.
 - **cookie**: Cadena con el valor del cookie asociado al documento.
- Métodos:
 - **clear()**: Borra el documento.
 - **write(..)** y **writeln(..)**: Añade texto al documento.

5. Modelo de Objetos.

Objeto document (II).

- **open(..)**: Abre un bloque de escritura para añadir nuevos contenidos (una vez cargado todo el documento HTML en el browser, para añadir contenidos con *document.write* es necesario abrir un bloque, en caso contrario podría no mostrarse).
- **close()**: cierra el bloque de escritura, visualizando el nuevo contenido (creado con *document.write*).

```
...
document.open();
document.write("<P>El contenido anterior ha sido
  borrado por motivos de ....");
document.write("<p> Para volver a visualizarlo, pulsa
  el botón de recargar");
document.close();
...
```

5. Modelo de Objetos.

Objeto imagen.

- Imágenes incluidos en el documento a partir de la directiva ****.
- Se acceden a ellas a partir del array *images* de *document*.
- Propiedades:
 - **name**: nombre de la imagen.
 - **src**: dirección url de la imagen.
 - **width / height**: ancho / alto de la imagen.
 - **hspace / vspace**: espacio horizontal/vertical que mantiene la imagen con los elementos circundantes.

```
<IMG name="logotipo" src= "logo.jpg" height= "300" >
...
document.images[0].height = 400;
document.images['logotipo'].height = 400;
document.images.logotipo.height = 400;
```

5. Modelo de Objetos.

Objeto link (I).

- Enlaces hipertexto incluidos en el documento, creados a partir de la directiva `<A>..`.
- Se acceden a partir del array *links* de *document*.
- Propiedades:
 - **hostname**: nombre de la máquina de la url.
 - **href**: contenido total de la url.
 - **protocol**: protocolo de la url.
 - **port**: puerto.
 - **target**: nombre del frame donde se visualizará el contenido del documento apuntado en la url.

```
for (var i=0; i<document.links.length; i++)  
    document.links[i].target = "frame1";
```

5. Modelo de Objetos.

Objeto link (II).

- Manejadores de eventos:
 - **onclick()** / **ondblclick()**: métodos invocados cuando se realiza un click / un doble click sobre el enlace.
 - **onmouse()** / **onmouseout()**: métodos invocados cuando se sitúa el cursor del ratón sobre / fuera del enlace.

```
...  
<a name="ftomas" href="http://informatica.uv.es/fcotomas/">  
Página de Paco Tomás</a>  
...  
document.links[0].onclick = new Function( "return  
confirm('Seguro que quieres visitar la página de Paco  
Tomás');" );  
...
```

5. Modelo de Objetos.

Objeto Form (I).

- Formularios incluidos en el documentos a través de la directiva `<FORM> ... </FORM>`.
- Se acceden a ellos partir del array *forms* de *document*.
- Propiedades:
 - **action**. URL donde se enviarán los datos del formulario.
 - **elements**. array que contiene todos los elementos del formulario: entradas texto, cajas de selección, botones de radio, áreas de texto, combo box, etc.
 - **encoding**: tipo de codificación de los datos.
 - **method**: método de envío (GET o POST).
 - **target**: frame donde se visualizará el resultado.

5. Modelo de Objetos.

Objeto Form (II).

- Manejadores de eventos:
 - **onreset()** y **onsubmit()**: manejadores de los eventos generados tras pulsar el botón de *reset* y de *submit*, respectivamente.

```
function CompruebaNombre() {
    a = document.forms[0].elements[0].value;
    // Las dos siguientes son equivalentes a la primera:
    a = document.forms["formulario1"].elements["fulano"].value;
    a = document.forms.formulario1.fulano.value;
    ...
}
...
<form name='formulario1' action='' onsubmit="alert('Mensaje1');">
Identificate:<input type='text' name='fulano'><br>
<input type='button' value='OK' name='confirma'
    onClick='CompruebaNombre()'><br><input type="submit"></form>
...
document.forms.formulario1.onsubmit=function(){alert('Mensaje2');}
...
```

5. Modelo de Objetos.

Objeto Element.

- Elementos incluidos dentro del formulario. Se acceden a ellos a través del vector *elements* del objeto *form*, o a través de su nombre (*name*).
- Propiedades:
 - **form**: referencia al formulario al que pertenece.
 - **type**: tipo de objeto.
 - **value**: valor asociado.
 - Otros particulares de cada tipo de elemento ...
- Manejadores de eventos:
 - **onfocus()**, **onblur()**: generales a todos los elementos.
 - **onchange()**: sólo para *Password*, *Text*, *Textarea* y *Select*.
 - **onclick()**: *Button*, *Checkbox*, *Radio*, *Reset*, *Submit*.
 - **ondblclick()**: *Button*, *Reset*, *Submit*.

6. Aplicaciones.

Manejo de cookies (I).

- Son utilizadas por la pagina visitada para almacenar información temporal. Se ahorra de esta manera guardar la información en el lado del servidor.
- Son simples ficheros texto.
- Todos los cookies se guardan en directorio fijo (habilitado para el caso).
- El acceso está restringido: en principio sólo las páginas del mismo dominio pueden acceder a la información.
- Utilidad de los cookies:
 - Identificar a un usuario durante una sesión de comercio elect.
 - Evitar introducir el usuario y el password de forma repetida.
 - Personalizar un sitio web para cada visitante.
 - Etc.

6. Aplicaciones.

Manejo de cookies (II).

En javascript, se accede a los datos del cookie a partir de la propiedad *cookie* de *document*:

```
var micookie = document.cookie; alert(micookie);  
document.cookie = "nombre=Juan;apellidos=Ruiz";
```

- Para que se grabe en disco debe fijarse una fecha de expiración. En caso contrario, sólo se mantiene temporalmente en memoria. Para fijar dicha fecha se utiliza la variable *expires*.

```
function AddDataCookie(variable, dato, miliseg_vida) {  
    if ( miliseg_vida ) {  
        var fecha = new Date();  
        fecha.setTime(fecha.getTime() + miliseg_vida);  
        document.cookie = (variable + '=' + escape(dato) +  
            ';expires=' + fecha.toGMTString());  
    }  
    else  
        document.cookie = variable + '=' + escape(dato);  
}
```

6. Aplicaciones.

Manejo de cookies (III).

```
function GetDataCookie(variable) {  
    var pos_ini, pos_fin, pos, mycook = document.cookie;  
    if (!mycook) return "";  
    pos = mycook.indexOf(variable);  
    if ( pos == -1 ) return "";  
    pos_ini = mycook.indexOf("=", pos);  
    if ( pos_ini != (pos + variable.length) ) return "";  
    pos_fin = mycook.indexOf(";", pos_ini) - 1;  
    if (pos_fin == -2) pos_fin = mycook.length - 1;  
    return unescape( mycook.substr(pos_ini+1,(pos_fin-pos_ini)) );  
}  
function DeleteVarCookie(variable) {  
    var fecha_cad = new Date();  
    fecha_cad.setTime(fecha_cad.getTime() - 1000);  
    document.cookie = variable + "=NULL;expires=" +  
        fecha_cad.toGMTString();  
}
```

6. Aplicaciones.



Texto animado en la barra de estado.

```
.....
<script language="javascript">
var speed = 200;
var mensaje = "Bienvenido a la página personal de P. T.      ";
var start = -1;
function AnimaMensaje() {
    start ++;
    if ( start >= mensaje.length ) start = 0;
    var texto = mensaje.substr(start, mensaje.length-start);
    texto += mensaje.substr(0,start);
    window.status = texto;
    window.setTimeout("AnimaMensaje()", speed);
}
</script>
.....
<body onLoad="AnimaMensaje()">
.....
```