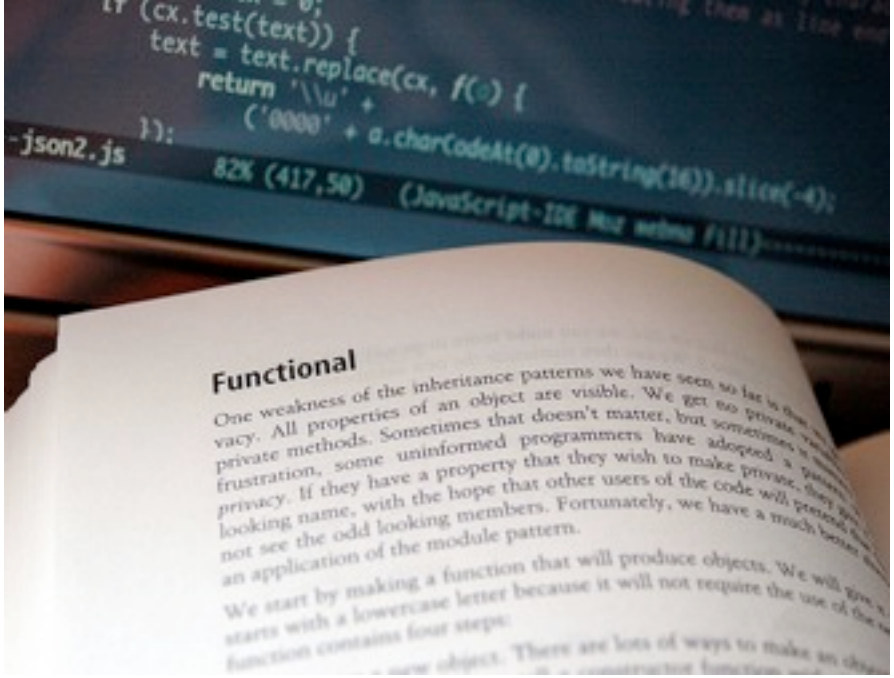
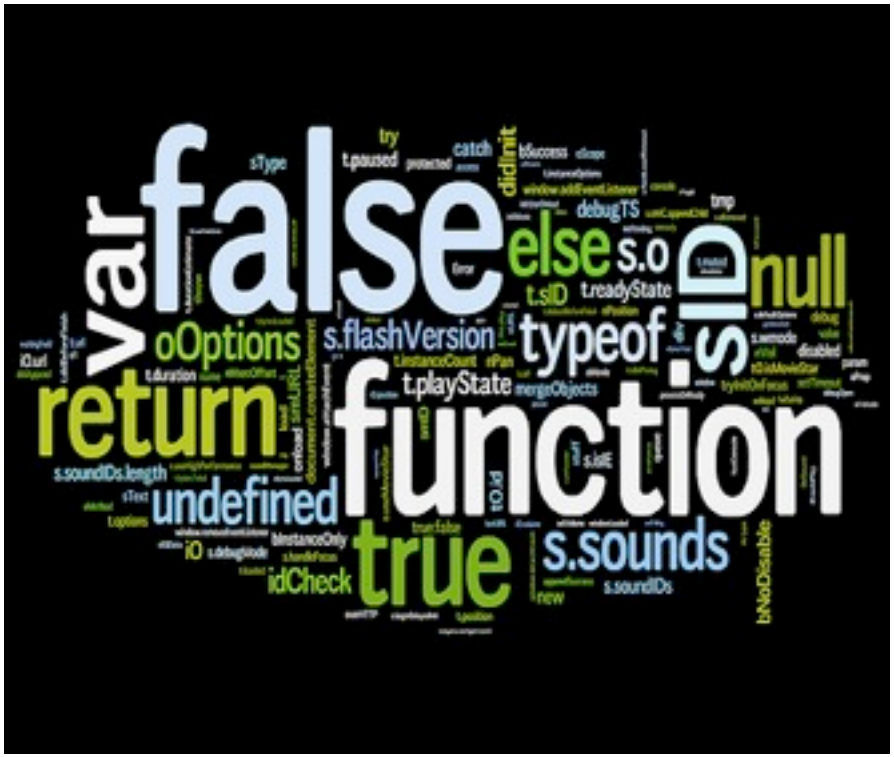




Tema 3

JavaScript

Parte I:
El núcleo del lenguaje.
Interfaz básico con el navegador.

Introducción a Javascript

3.1

Conceptos previos

¿Qué es javascript? ¿En qué se diferencia de otros lenguajes de programación web?

JavaScript

- NO es:
  - Una versión reducida de Java
  - Un lenguaje "simple"
- Tiene
  - Visible el código fuente, ya que es interpretado, no compilado
    - No obstante, el fuente se puede ofuscar
    - Las implementaciones modernas hacen una compilación JIT (*just-in-time*) a código máquina
  - Sintaxis similar a C++ o Java, mucho menos restrictiva
  - Orientación a objetos:
    - Distinta filosofía que C++ o Java: en JavaScript no existen clases.
    - Los objetos son colecciones de métodos y propiedades.

¿Qué se puede hacer con JavaScript?

- Se usa para:
  - Interactuar con el navegador: abrir ventanas, saltar a otra URL,…
  - Interactuar con el documento HTML: verificar formularios, añadir/eliminar contenido, hacer animaciones…
  - Abrir conexiones con el servidor (AJAX)
- No se suele usar para :
  - Multimedia: videos, sonido, etc.
  - Trabajar con bases de datos. Esto se suele hacer desde el servidor
- Estas capacidades no vienen del lenguaje en sí, sino de la forma en la que está actualmente integrado en los navegadores
  - De hecho, existen librerías para acceder a bases de datos locales con SQL y se puede dibujar en 2D/3D

```
var Torrent = Class.create();

Torrent.prototype = {
  initialize: function(xml) {
    var xmlString = new XMLSerial
    var torrentNode = xml.getElem
    var statsNode   = xml.getElem

    this.objectID = getText(torre
    this.name = getText(torrentNo
    this.displayName = (this.name
                        this.name
                        this.name
                        this.name
    this.hash = getText(torrentNo
```


Introducción a Javascript

3.2

El núcleo de Javascript

JavaScript como lenguaje, independientemente de la web

El léxico de JavaScript

- JavaScript es descendiente de la familia C, C++, Java
  - Diferencia mayúsculas/minúsculas
  - Comentarios tipo C/C++ /* Comentario */, //comentario
- El **;** es opcional (si cada sentencia está en una línea), aunque es recomendable para evitar problemas

```
a = 3
b = 4
```

Declaración de variables

- No obligatoria, aunque aconsejable
- Las variables no tienen tipo fijo, se declaran simplemente con la palabra clave **var**
- El valor de una variable declarada pero no inicializada es un valor especial llamado undefined

```
var a;
b = 27;          //válido
a = 3;
a = "hola";   //válido
var c,d;
console.log(c)  //undefined
console.log(d)  //error
```

Tipos de datos

- Numérico (enteros y reales)
- Booleano (`true==1, false==0`)
- Clases básicas del sistema
  - String
  - Object (tipo base para los objetos)
  - function
  - Array
  - Date
  - RegExp (expresión regular)

Operadores y sentencias

- Idem a C++/Java
  - Operadores aritméticos y lógicos (eso sí, no se pueden redefinir)
  - Sentencias: bucles, condicionales, etc.
- Igualdad (==): emplea conversión de tipos
- Identidad(===): sin conversión

```
if ("1"==true)     //esto es cierto
...
if ("1"===true)     //pero esto no
```

- **delete** existe, pero no significa lo mismo que en C++, lo veremos cuando hablemos de objetos

Operador de asignación

- Datos primitivos: por valor (copia)
- Objetos: por referencia (ambas variables apuntan al mismo objeto)
  - Aunque existen punteros, no se pueden emplear como en C++ (desplazarse por la memoria, obtener la direccion, etc.)

```
var a,b;
a = new Array();   // un array es un objeto que se
                   // crea con el constructor Array()
a[0] = 1;
b = a;             // b referencia al array a
a[0] = 100;
alert(b[0]);       // muestra el valor 100
```

Funciones

- Se definen con la palabra clave **function**
- Los parámetros no tienen tipo (como es lógico)
- Es recomendable declarar las variables locales, para no coincidir con una global

```
var res = 1;
function suma (arg1, arg2)  {
      var res = arg1 + arg2;
      return res;
}
```

- Las funciones son objetos, y por tanto se pueden asignar a variables o pasar como parámetros

```
function operar(arg1,arg2,op) {
  return op(arg1,arg2)
}
operar(2,2,suma);
```

Trabajo con objetos

- Creación, inserción de propiedades y borrado dinámico

```
var Homer;
Homer = new Object();
Homer.nombre = "Homer Simpson";
Homer.edad = 34;
Homer.casado = true;
delete Homer.edad    //(Homer.edad==undefined)
```

- Acceso

```
Homer.edad = 40;      //Notación "clásica"
Homer["edad"] = 40;  //Como si fuera un array
var prop = prompt("¿Qué propiedad quieres?");
alert(Homer[prop]);
```

JSON (JavaScript Object Notation)

- Permite representar un objeto con "texto plano"
- Útil para inicializar objetos o recibirlos desde el servidor (lo veremos en AJAX)

```
json = {
  nombre: "Homer",
  apellido: "Simpson",
  tels: [           //Los arrays se definen con el corchete
    "555-123456",
    "555-654321"
  ],
  ocupacion: {       //Una propiedad puede ser un objeto JSON
      puesto: "operario",
      lugar: "central nuclear de Springfield"
  }
}
//"eval" analiza el JSON y lo convierte en objeto Javascript
homer = eval(json);
```

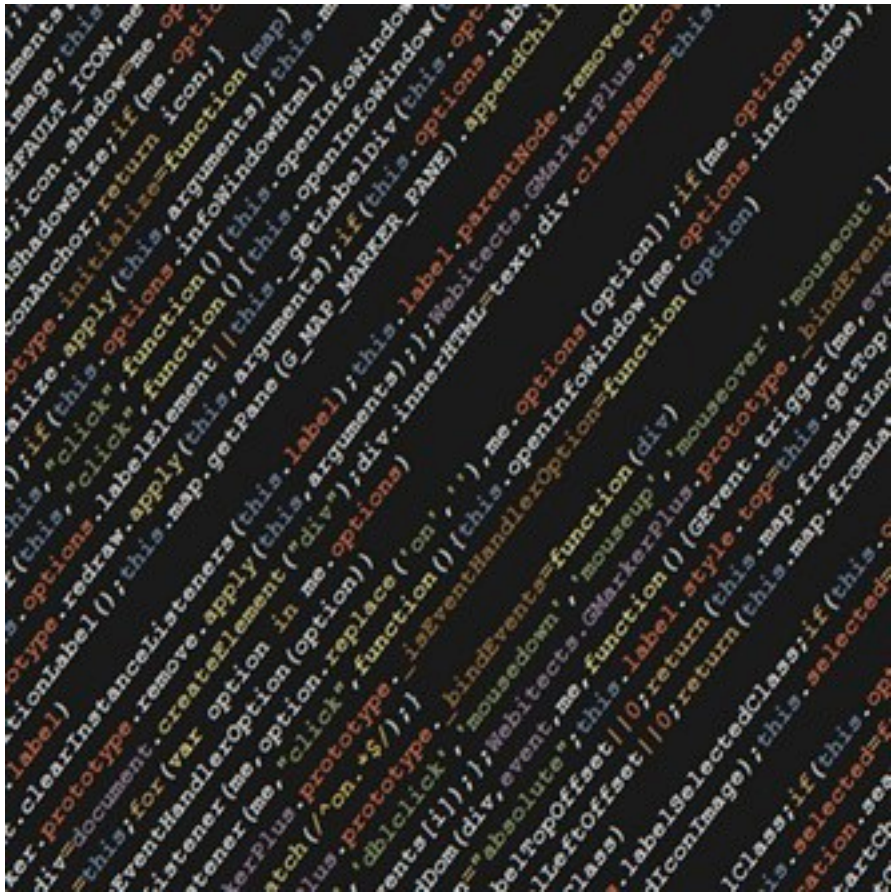
“Imitación” del estilo C++/Java/...


```
var homer;

function Persona(nombre, edad, casado)  {
      this.nombre  = nombre;
      this.edad = edad;
      this.casado = casado;
}

homer = new Persona(“Homer Simpson”, 34, true);
```

Arrays

- **Dispersos**: no todas las posiciones tienen que contener datos. Las vacías tienen `undefined`
- **Heterogéneos**: cada posición puede tener un tipo distinto

```
var a=new Array()
a[5]=4
a[10]="hola"
alert(a.length)   //11!!
```



Introducción a Javascript

3.3

Scripts en páginas web

¿Cómo metemos código Javascript en el HTML?

Texto

Insertar Javascript en el HTML

- En etiquetas **script**
- El ámbito de las variables es la página entera
  - Las variables no se pueden compartir entre páginas
- El código se ejecuta a medida que se va leyendo

```
<html>
<head>
   <script type="text/javascript">  //type obligatorio en HTML4
      function ahora() {            //pero no en HTML5
         var h = new Date();
         return h.toLocaleString(); }
      var verFecha = true;
   </script>
</head>
<body>
   <script type="text/javascript">
      if (verFecha)  alert("Fecha y hora: " + ahora());
   </script>
</body>
</html>
```

Otras formas de incluir código

- En un fichero aparte (.js). Similar al #include de C

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

- En un manejador de evento
  - El código se ejecuta al producirse el evento, no al leer la página

```
<input type="button" value="pulsa aqui"
       onClick="alert('hola')">
```

- Como una URL **javascript:**
  - Para poder ejecutar código en respuesta a un click en un enlace

```
<a href="javascript:alert('hola')"> ¡Hola! </a>
```

http://www.flickr.com/photos/reign4aday/1498945370

Introducción a Javascript

3.4

Interfaz con el navegador (Browser Object Model o BOM)

API orientado a objetos para interactuar con el navegador

APIs de JavaScript para la web

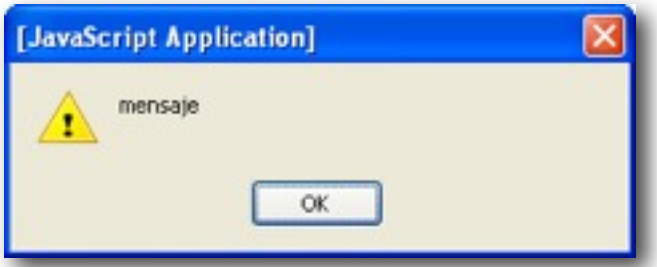
- En web nos puede interesar interactuar con:
  - El navegador:
    - Obtener las propiedades y capacidades
    - Mover la ventana, abrir un popup (cada vez menos),…
    - Hacer que salte a otra dirección
  - El propio HTML: (el API se llama DOM, lo veremos en las 2 clases siguientes)
    - Añadir texto, etiquetas, quitar etiquetas
    - Reordenar partes del documento (por ejemplo una tabla por columnas)
    - Hacer animaciones

Objeto global: window

- El objeto **window** determina el contexto de ejecución
  - Las variables "globales" son propiedades de `window`

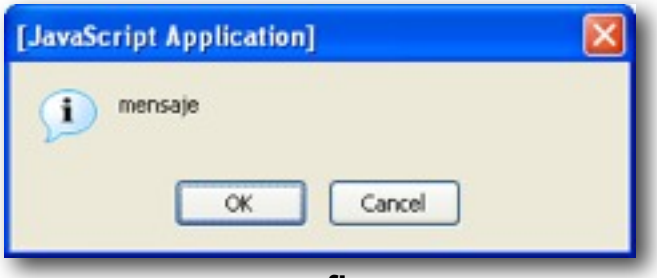
```
a = 7;
alert (window.a)   //7
```

- Cuadros de diálogo modales
  - `alert(mensaje)`
  - `prompt(mensaje, valor_por_defecto)`: devuelve cadena introducida o `null` si se ha pulsado cancelar
  - `confirm(mensaje)`: devuelve `true` o `false` según si se pulsa Aceptar o Cancelar



alert

prompt



confirm

Objeto window: algunas propiedades

Propiedad	Significado
`closed`	valor booleano que indica si la ventana ha sido cerrada
`status`	texto de la barra de estado (para un mensaje temporal)
`defaultStatus`	texto por defecto de la barra de estado (es reemplazado momentáneamente cuando se pasa el ratón por un enlace u otros objetos)
`name`	nombre de la ventana
`opener`	referencia al objeto `window` que creó la ventana actual
`parent`	si la ventana actual es un frame, referencia a la ventana que lo contiene. En caso contrario, es lo mismo que `window`
`top`	si la ventana actual es un frame, referencia a la ventana de nivel superior que lo contiene. En caso contrario, es lo mismo que `window`
`self, window`	la propia ventana

Objeto navigator

Propiedad	Significado
`appName`	Nombre común del navegador. Ejemplos: `Netscape`, `Microsoft Internet Explorer`
`appVersion`	Número de versión e información adicional. Ejemplo: en Navigator 4.6, versión inglesa para Windows tiene el valor: `4.6 [en] (Win98; I)`
`userAgent`	La información que envía el navegador en la cabecera http `USER-AGENT`

- En el pasado, el objeto Navigator se usaba para ejecutar un código u otro dependiendo de la compatibilidad.
  - Pero eso era cuando solo existían Navigator y Explorer
  - En la actualidad es más sencillo comprobar si el método o propiedad que necesitamos existe. Si no, será `undefined` (==false)

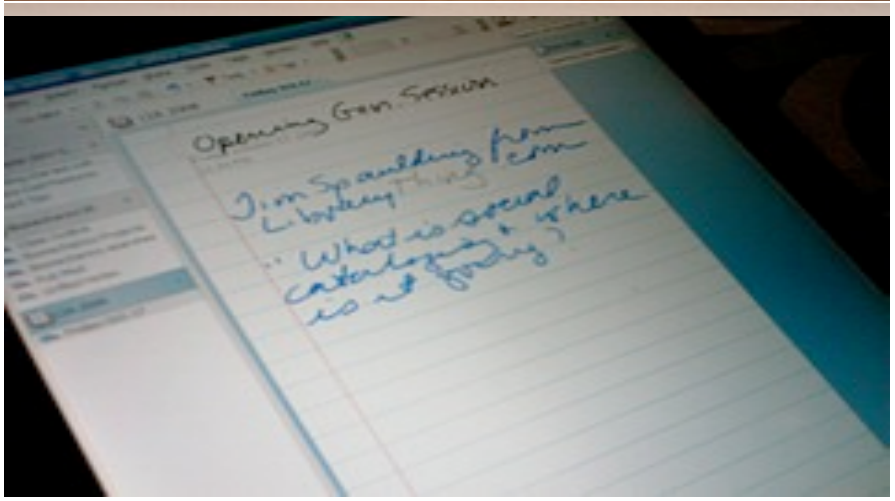
```
if (document.all) {
      …     //Estamos en Internet Explorer
}
```

Objeto location

Propiedad	Significado
`href`	cadena que representa la URL completa
`protocol`	solo la parte del protocolo (ej, http:)
`host`	solo el nombre del host
`pathname`	trayectoria hasta el recurso, incluido el mismo
`search`	parte de la URL que sigue al carácter "?",(incluido) si está presente.

- Por ejemplo, para hacer que el navegador salte a la página "login.htm"

```
location.href="login.htm"
```



http://www.flickr.com/photos/dakdad/6011630039

3.5

Eventos

Reaccionando a las acciones del usuario

Introducción a Javascript

Eventos y manejadores

- Los eventos son acciones que ocurren generalmente porque el usuario hace algo sobre un objeto. Por ejemplo, hacer click sobre un botón, introducir texto en un campo de formulario, mover el ratón sobre un enlace ...
- Se pueden controlar esas acciones con un '**manejador**' de eventos (***handler***), para que el script reaccione ante ese suceso. Estos son los eventos que utiliza y controla javascript.

Algunos tipos de eventos

Evento	Se aplica a...	Ocurre cuando...	Handler
abort	imagenes	El usuario aborta la carga de una imagen (por ejemplo haciendo click sobre un enlace o en el boton del navegador) .	onAbort
blur	ventanas, frames, y todos los elementos de formularios	El usuario cambia el foco a otro elemento (ventana, frame, o elemento del formulario).	onBlur
click	Casi todos los elementos HTML	El usuario hace click en algo. Devolver false para cancelar la acción por defecto	onClick
change	campos de texto, area de texto, listas de selecisn.	El usuario cambia el valor de un elemento.	onChange
error	imagenes, ventanas	La carga de un documento o imagen causa un error	onError
focus	ventanas, frames, y todos los elementos de formularios	El usuario pasa el foco a otro elemento (ventana, frame, o elemento del formulario).	onFocus
load	body	El usuario carga la pagina.	onLoad

Algunos tipos de eventos *(cont.)*

Evento	Se aplica a...	Ocurre cuando...	Handler
mouseout	Casi todos los elementos HTML	El usuario mueve el puntero del ratsn fuera de un area (imagen) o enlace.	onMouseout
mouseover	Casi todos los elementos HTML	El usuario mueve el puntero del ratsn sobre un area (imagen) o enlace. En enlaces, devolver true para que no aparezca en la barra de status	onMouseover
reset	Formularios.	El usuario resetea un formulario (clicks en el botsn Reset) Devolver false para que no se haga el reset.	onReset
select	campos de texto, area de texto	El usuario selecciona un campo de entrada del formulario	onSelect
submit	formularios	El usuario envía un formulario. Devolver false para que no se haga el envío.	onSubmit
unload	Body,frameset	El usuario sale de la pagina	onUnload

Eventos y manejadores (forma inline)

1. Definir un manejador en un atributo HTML *onXXX="codigo-JavaScript"*
   - Problema: mezclamos HTML y javascript

```
<form name="prueba">
   <input type="button" value="Pulsa aquí"
        onClick="alert('hola');/>
</form>
```

2. Especificar valor de retorno
   - Devolviendo false o true se puede anular el comportamiento por defecto de algunos eventos. La mayoría de las veces es false, pero algunas es true

```
<a href="hola.htm" onclick="alert('hola');
return false;">hola</a>
```

Definir manejador de forma “no intrusiva”

- En el navegador, cada etiqueta HTML del documento tiene un objeto JavaScript equivalente.
  - Se accede al objeto (si la etiqueta tiene un `id`) con `document.getElementById`
  - Ese objeto tiene propiedades predefinidas que se corresponden con los atributos HTML.
  - Por ejemplo, un input tendría una propiedad `value`, otra `onclick`,...
- Ventaja: no mezclamos Javascript con HTML

```
<p id="par">Clícame</p>
<script type="text/javascript">
    function saludo() { alert("Hola")}
    document.getElementById(“par”).onclick = saludo;
    //también podríamos definir la función “sobre la marcha”
    //(función “anónima”)
   document.getElementById(“par”).onclick = function() {
                                            alert(“Hola”)}
</script>
```

this: el objeto que dispara el evento

- En un manejador de evento, **this** referencia al objeto sobre el que se produce el evento

```
<input type="text" id="campo">
<script type="text/javascript">
  function verContenido() {
    alert(this.value);
  }
  campo.onclick = verContenido;
</script>
```

Modificar el HTML: innerHTML

- Propiedad que nos permite leer/modificar el HTML que hay dentro de una etiqueta
- No es estándar en HTML4, pero sí en HTML5
  - Posteriormente veremos la forma estándar (DOM)
  - `innerHTML` no se recomienda para trabajar con tablas

```
<input type="button" value="Pon texto" onclick="ponTexto()"/>
<div id="texto"></div>
<script type="text/javascript">
  function ponTexto() {
    var mensaje = prompt("Dame un texto y lo haré un párrafo")
    var etiq = document.getElementById("texto")
    etiq.innerHTML += "<p>" + mensaje + "</p>"
  }
</script>
```

Eventos de formulario

- onsubmit (en el <form>): al intentar enviar los datos al servidor
  - Si el manejador devuelve `false`, se anula el envío
- onchange (en un campo): al cambiar el valor del campo (normalmente se dispara al pasar el foco a otro campo)

```
<script type="text/javascript">
  function checkForm() {
    var valor = document.getElementById("nombre").value
    if ((valor=="") || (valor==null))
        {alert("No puede estar vacío")
        return false;}
    else return true;}
</script>
<form action="loquesea.php" id="miform" onsubmit="return checkForm()">
    Nombre: <input type="text" id="nombre">
    <input type="submit" value="Enviar">
</form>
<!-- forma ALTERNATIVA a poner el "onsubmit=..." del formulario -->
<script type="text/javascript">
    document.getElementById("miform").onsubmit = checkForm;
</script>
```

Información sobre el evento

- El evento también es un objeto JavaScript con propiedades
  - Por ejemplo en un click, la propiedad `button` nos dice qué botón se ha pulsado
- Incompatibilidad
  - Explorer: el evento es un objeto global llamado "event"
  - Estándar W3C: se le pasa como argumento al manejador de evento

```
<p id="par">Clícame</p>
<script type="text/javascript">
    function saludo(e) {                    //"e" solo definido en Estándar
        if (window.event) {                 //event solo definido en Explorer
           e = window.event;
        }
        alert("Se ha pulsado el botón  " + e.button);
    }
   document.getElementById("par").onclick = saludo;
</script>
```

Event bubbling

- En realidad el evento no se dispara solo sobre el objeto que actúa directamente, continúa con los objetos "padre", "abuelo",...
  - Se puede cancelar el *bubbling*, o "ascenso" del evento, pero el Explorer es incompatible con el estándar, no lo vamos a ver aquí.

```
<body onclick="alert('body')">
   <p onclick="alert('p')">
        <a href="" onclick="alert('a')">Clícame</a>
   </p>
</body>
```

Eventos “avanzados” (estándar W3C)

- Propagación: capturing + bubbling
  - Capturing significa que el evento comienza en realidad en el nivel superior
  - El capturing no funciona con los handler convencionales
- En lugar de un manejador de evento se usa un Event listener
  - Ante el mismo evento, un objeto puede tener solo un handler, pero varios listeners
  - Definición
    - objeto.addEventListener(evento, funcion, captura?)(W3C)
    - objeto.attachEvent(evento, funcion) (Explorer)
  - Eliminación
    - objeto.removeEventListener(evento, funcion, captura?)(W3C)
    - objeto.detachEvent(evento, funcion)(Explorer)