

# Gestión de Eventos

## Tema 7

[http://www.um.es/docencia/barzana/DAWEB/  
Lenguaje-de-programacion-JavaScript-5.pdf](http://www.um.es/docencia/barzana/DAWEB/Lenguaje-de-programacion-JavaScript-5.pdf)

# Eventos

- Los eventos son mecanismo que se accionan cuando el usuario realiza un cambio sobre una página web.
- el DOM el encargado de gestionar los eventos.

# Eventos

- Para poder controlar un evento se necesita un manejador.
- El manejador es la palabra reservada que indica la acción que va a manejar.
- En el caso del evento *click*, el manejador sería *onClick*. Ejemplo:

**<IMG SRC="mundo.jpg" onclick="alert('Click en imagen');">**

# Eventos

- El ejemplo anterior se puede realizar de otro modo llamando a una función:

```
<html>
```

```
<head>
```

```
<title>Paginade Evento</title>
```

```
<script>function func1()
```

```
{alert("Click en imagen");}</script>
```


```
</head>
```

```
<body><IMG SRC="mundo.jpg"  
onclick="func1();"></body></html>
```

# Eventos

- La especificación DOM define cuatro grupos de eventos dividiéndolos según su origen:
- Eventos del ratón.
- Eventos del teclado.
- Eventos HTML.
- Eventos DOM.

# Eventos

- Eventos del ratón (1): 
- **Click.** Este evento se produce cuando pulsamos sobre el botón izquierdo del ratón. **El manejador de este evento es onclick.**
- **Dblclick.** Este evento se acciona cuando hacemos un doble click sobre el botón izquierdo del ratón. **El manejador de este evento es ondblclick.**
- **Mousedown.** Este evento se produce cuando pulsamos un botón del ratón. **El manejador de este evento es onmousedown.**
- **Mouseout.** Este evento se produce cuando el puntero del ratón esta dentro de un elemento y este puntero es desplazado fuera del elemento. **El manejador de este evento es onmouseout**

# Eventos

- Eventos del ratón (2):
- **Mouseover.** Este evento al revés que el anterior se produce cuando el puntero del ratón se encuentra fuera de un elemento, y este se desplaza hacia el interior. **El manejador de este evento es onmouseover.**
- **Mouseup.** Este evento se produce cuando soltamos un botón del ratón que previamente teníamos pulsado. **El manejador de este evento es onmouseup.**
- **Mousemove.** Se produce cuando el puntero del ratón se encuentra dentro de un elemento. Es importante señalar que este evento se producirá continuamente una vez tras otra mientras el puntero del ratón permanezca dentro del elemento. **El manejador de este evento es onmousemove.**

# Eventos

- Eventos del teclado:
- **Keydown.** Este evento se produce cuando pulsamos una tecla del teclado. Si mantenemos pulsada una tecla de forma continua, el evento se produce una y otra vez hasta que soltemos la misma. **El manejador de este evento es onkeydown.**
- **Keypress.** Este evento se produce si pulsamos una tecla de un carácter alfanumérico (El evento no se produce si pulsamos enter, la barra espaciadora, etc...). En el caso de mantener una tecla pulsada, el evento se produce de forma continuada. **El manejador de este evento es onkeypress.**
- **Keyup.** Este evento se produce cuando soltamos una tecla. **El manejador de este evento es onkeyup**



# Eventos

- Eventos HTML (1):
- **Load.** El evento load hace referencia a la carga de distintas partes de la página. Este se produce en el objeto Window cuando la página se ha cargado por completo. En el elemento <img> actúa cuando la imagen se ha cargado. En el elemento <object> se acciona al cargar el objeto completo. **El manejador es onload.**
- **Unload.** El evento unload actúa sobre el objeto Window cuando la página ha desaparecido por completo (por ejemplo, si pulsamos el aspa cerrando la ventana del navegador). También se acciona en el elemento <object> cuando desaparece el objeto. **El manejador es onunload.**
- **Abort.** Este evento se produce cuando el usuario detiene la descarga de un elemento antes de que haya terminado, actúa sobre un elemento <object>. **El manejador es onabort.**

# Eventos

- Eventos HTML (2):
- **Error.** El evento error se produce en el objeto Window cuando se ha producido un error en JavaScript. En el elemento <img> cuando la imagen no se ha podido cargar por completo y en el elemento <object> en el caso de que un elemento no se haya cargado correctamente. **El manejador es onerror.**
- **Select.** Se acciona cuando seleccionamos texto de los cuadros de textos <input> y <textarea>. **El manejador es onselect.**
- **Change.** Este evento se produce cuando los cuadros de texto <input> y <textarea> pierden el foco y el contenido que tenían ha variado. También se producen cuando un elemento <select> cambia de valor. **El manejador es onchange.**
- **Submit.** Este evento se produce cuando pulsamos sobre un botón de tipo submit. **El manejador es onsubmit.**

# Eventos

- Eventos HTML (3):
- **Reset.** Este evento se produce cuando pulsamos sobre un botón de tipo reset. **El manejador es onreset.**
- **Resize.** Este evento se produce cuando redimensionamos el navegador, actúa sobre el objeto Window. **El manejador es onresize.**
- **Scroll.** Se produce cuando varía la posición de la barra de scrollen cualquier elemento que la tenga. **El manejador es onscroll.**
- **Focus.** Este evento se produce cuando un elemento obtiene el foco. **El manejador es onfocus.**
- **Blur.** Este evento se produce cuando un elemento pierde el foco. **El manejador es onblur.**

# Eventos

- Eventos DOM:
- **DOMSubtreeModified**. Este evento se produce cuando añadimos o eliminamos nodos en el subárbol de un elemento o documento.
- **DOMNodeInserted**. Este evento se produce cuando añadimos un nodo hijo a un nodo padre.
- **DOMNodeRemoved**. Este evento se produce cuando eliminamos un nodo que tiene nodo padre.
- **DOMNodeRemovedFromDocument**. Este evento se produce cuando eliminamos un nodo del documento.
- **DOMNodeInsertedIntoDocument**. Este evento se produce cuando añadimos un nodo al documento

# Gestión de Eventos

- La ejecución de scripts se asociará normalmente a la ejecución de un evento.
- Se pueden asociar mediante los eventos intrínsecos de html o mediante técnicas de script.

Se podrá asociar código JavaScript a un evento de un elemento determinado:

`<elemento nombreEvento = "códigoJavaScript">`

- El nombreEvento aparece en el elemento en forma de atributo.
- El códigoJavaScript será el manejador de eventos (event handler).
- Puede estar formado por una o más instrucciones JavaScript.
- Lo normal es que se trate de una llamada a una función JavaScript contenida en el elemento head de la página o en un archivo externo.

# Gestión de Eventos

- Ejemplo: Al pasar el ratón sobre el párrafo, aparece con fondo negro sobre blanco. Cuando sale se vuelve al modo normal.
- [Ejemplo1](#)
- [ejemplo2](#)
- El uso de `this` en el código evita tener que buscar la referencia al párrafo en las instrucciones JavaScript.
  - `this` pasa la referencia al elemento actual, en este caso, el propio párrafo.
  - Si no se utilizara habría que haber puesto:
  - `document.getElementById("otroparrafo").style.color=`
  - `'white'...`

# Gestión de eventos

## Asociar event handlers mediante script.

- Los eventos se representan **como propiedades de un elemento**, por lo que podemos asignar su valor mediante instrucciones JavaScript.

**elemento.nombreEvento = referenciaAFuncion**

referenciaAFuncion sería el nombre de la función **sin los paréntesis**.

Si aparecen los paréntesis sería una llamada a función.

- Esta técnica permite “limpiar” el código html, separando la gestión de eventos de la estructura y contenido de la página.

Para realizarlo es necesario...

- **Asignar un id** a cada elemento al que se quiera asociar un evento.
  - Esto es así, porque hay que localizar el elemento.
- **Crear la función** que controle el evento.
- **Asociar al atributo de evento del elemento la referencia a la función.**
  - Eso se debe hacer después de haber cargado todos los elementos de la página, generalmente en el evento window.onload.

# Gestión de eventos

- Ejemplo: resaltar un párrafo al pasar el ratón sobre él.
- Las funciones reciben el argumento event, que puede guardar información sobre el evento como la tecla que se ha pulsado (keyCode) o la posición horizontal del cursor (clientX).

[ejemplo](#)



# Gestión de eventos

- Objeto Event

- JavaScript permite obtener información sobre eventos mediante un objeto especial llamado event ( sobre todo se utiliza para eventos de ratón y teclado ).
- La propiedad type indica el tipo de evento producido, lo que es útil cuando una misma función se utiliza para manejar varios eventos:

```
var tipo = evento.type;
```

# event

- A veces necesitamos obtener información adicional sobre algunos aspectos del evento. Los casos más normales es obtener la posición del ratón en los eventos de ratón, o la tecla con la que se actúa en los eventos de teclado. Para ello javascript tiene un objeto especial llamado event.
- Sin embargo los diferentes navegadores tienen distintas formas de obtener este objeto.

# event

- Internet explorer considera que forma parte del objeto window, con lo cual se obtiene mediante el código `window.event`
- El resto de navegadores lo consideran como un argumento implícito que se pasa a la función al ser llamada por un evento, con lo que no tenemos más que darle un nombre al argumento para obtener este objeto:
- `function manejaEventos(elEvento) { ...`
- En este caso al llamar a la función desde un evento, el objeto que nos da la información se llamaría `elEvento`

# event

- Para obtener un código compatible con ambos tipos de navegadores, deberemos escribir el código de la siguiente manera:
- ```
function manejaEventos(elEvento) {  
  var evento = elEvento || window.event;
```
- Donde evento es el objeto que nos dará la información sobre el evento, tanto en Internet Explorer como en el resto de navegadores. Podemos cambiar el nombre del argumento, en lugar de llamarlo **elEvento** podemos llamarlo de cualquier otra manera, al igual que el nombre del objeto - aquí lo llamamos evento -, pero lo que no podemos variar es el window.event, ya que esta es la forma de obtener el objeto en Internet Explorer.

# event

- La propiedad `type` permite distinguir el tipo de evento producido, lo cual permite distinguir entre los distintos eventos que puede manejar una función.
- ```
function manejaEventos(elEvento) {  
  var evento = elEvento || window.event;  
  var tipo = evento.type
```

# Gestión de eventos

- La propiedad type se utiliza sobre todo cuando varios eventos llaman a una misma función.
- [Ejemplo](#)

# Gestión de eventos

- Ejemplo: En un formulario, resaltar los campos cuando están activos.
- Podemos asignar eventos a la vez a distintos elementos, recorriéndolos mediante bucles.

[Ejemplo](#)

- Ejemplo: hacer un rollover de una imagen.

[ejemplo](#)

# addEventListener

- El método **addEventListener()** es la técnica ideal y la que es **considerada como estándar por la especificación de HTML5.**
- Este método tiene tres argumentos: el nombre del evento, la función a ser ejecutada y un valor booleano (falso o verdadero) que indica cómo un evento será disparado en elementos superpuestos
- ```
var elemento=document.getElementsByTagName('p')[0];  
elemento.addEventListener("click", mostraralerta, false);
```
- [Ejemplo](#)
- Para eliminar eventos asociados:**removeEventListener**