Gestión de Eventos

Tema 7

Eventos

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

Eventos

- Un mismo tipo de evento (por ejemplo, pinchar el botón izquierdo del ratón) puede estar definido para varios elementos XHTML diferentes y un mismo elemento XHTML puede tener asociados varios eventos diferentes.

Eventos

- El nombre de cada evento se construye mediante el prefijo on, seguido del nombre en inglés de la acción asociada al evento. Así, el evento de pinchar un elemento con el ratón se denomina onclick y el evento asociado a la acción de mover el ratón se denomina onmousemove.

Manejadores de eventos

- Un evento de JavaScript por sí mismo carece de utilidad.
- Para que los eventos resulten útiles, se deben asociar funciones o código JavaScript a cada evento.
- De esta forma, cuando se produce un evento se ejecuta el código indicado, por lo que la aplicación puede *responder* ante cualquier evento que se produzca durante su ejecución.
- Las funciones o código JavaScript que se definen para cada evento se denominan ***"manejador de eventos"*** .

Formas para indicar manejadores

- Manejadores como atributos de los elementos XHTML.
- Manejadores como funciones JavaScript externas.
- Manejadores *"semánticos"*.

Manejadores de eventos y variable this

- En los eventos, se puede utilizar la variable this para referirse al elemento XHTML que ha provocado el evento.
- Normalmente haremos funciones externas para programar un evento, lo haremos con addEventListerner.
- (ojo, aquí no se puede utilizar el this directamente, habrá que pasarselo)

Probando

Creamos un formulario con tres botones, cada uno con un valor en la propiedad value diferente.

Se trata de asignar el evento click a cada uno y visualizar el valor de la propiedad value.

Los tres tendrán la misma función en común que mostrará en un alert el contenido de value.

Manejadores de eventos

- 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);

- Para eliminar eventos asociados:**removeEventListener**

Eventos

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

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 clicksobre 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 esonmouseout**

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 unloadactúa sobre el objeto Window cuando la pagina 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

Obteniendo información del evento (objeto event)

- JavaScript permite obtener información sobre el ratón y el teclado mediante un objeto especial llamado event.

- La principal diferencia reside en la forma en la que se obtiene el objeto event.
- Internet Explorer considera que este objeto forma parte del objeto window y el resto de navegadores lo consideran como el único argumento que tienen las funciones manejadoras de eventos.

Obteniendo información del evento (objeto event)

- Excepto Internet Explorer todos los navegadores crean *mágicamente* y de forma automática un argumento que se pasa a la función manejadora, por lo que no es necesario incluirlo en la llamada a la función manejadora.
- De esta forma, para utilizar este *"argumento mágico"*, sólo es necesario asignarle un nombre, ya que los navegadores lo crean automáticamente.

Ejemplos event

```
var evento = window.event;(ie)

function manejadorEventos(elEvento) {
var evento = elEvento;
}(resto)
```

```
function manejadorEventos(elEvento) {
var evento = elEvento || window.event;
}(todos)
```

Información sobre el evento

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;
- La propiedad type devuelve el tipo de evento producido, que es igual al nombre del evento pero sin el prefijo on.

Probando

- Retocar el ejemplo anterior.
- El ejercicio se trata de asignar varios eventos a un mismo elemento y según que evento visualizamos un mensaje indicando el evento que se ha producido.
- A parte del click que ya lo tenemos, añadimos el mouseover y el blur .
- La función es la misma para los dos , en el primer caso se colorea el fonde del botón y en el segundo se visualiza el nombre del evento.

Funcionalidades del objeto Event

- Obtener el elemento del actual evento disparado.
- Una de las propiedades más importantes de nuestro objeto de destino se llama **target.**

Ejemplo:

```
<span id="pulsame"> Pulse aquí </ span>
<script>
var pulsame = document.getElementById ('pulsame');
pulsame.addEventListener ('click', function (e)
{e.target.innerHTML = 'Ha hecho clic en‘;}, false);
</script>
```

Funcionalidades del objeto Event

- - Obtener el elemento en el origen del evento desencadenante.
- La solución es simple: utilizar la propiedad **currentTarget** en lugar de **target.** Probar el ejemplo primero con target y después con currentTarget.

Ejemplo:

```
<p id="resultado"></p>
<div id="padre1“>Padre
<div id="child1"> Niño N º 1 </div>
<div id="child2"> Niño N ° 2 </div>
</div>
<script>
var padre1 = document.getElementById ('padre1')
result = document.getElementById ('resultado');
padre1.addEventListener ('mouseover', function (e) {result.innerHTML = "El
    desencadenador del evento \ "MouseOver \" tiene la id: "+ e.target.id;
}, false);
</script>
```

Funcionalidades del objeto Event

- - Recuperar la posición del cursor
- Ejemplo:

```
<div id="posicion"> </div>
<script>
var posicion = document.getElementById ('posicion');
document.addEventListener ('mousemove', function (e) {
posicion.innerHTML = 'Posición X' + e.clientX + "px <br/>
    Posición
Y: '+ e.clientY +' px ‘;}, false);
</script>
```

Funcionalidades del objeto Event

- Recuperar el elemento en relación a un evento de ratón: **relatedTarget** y se utiliza con los eventos mouseover y mouseout.
- En el caso:
- mouseout, proporciona el objeto del elemento en el cual el cursor se ha introducido.
- mouseover, dará el objeto del elemento del que el cursor acaba de salir.

Funcionalidades del objeto Event

- Recuperar las teclas pulsadas por el usuario
- Los eventos KeyUp y KeyDown están diseñados para capturar las pulsaciones de teclado.(letras, crtl…etc)
- El evento keypress, tiene un propósito completamente diferente: capturará las teclas que escriben un carácter.
- Su ventaja radica en su capacidad para detectar las combinaciones de teclas.

Funcionalidades del objeto Event

si se ejecuta la combinación:

- Mayús + A, el evento keypress detectará una A mayúscula donde los eventos KeyDown y

  KeyUp se dispararán dos veces, una vez para la tecla Shift y una segunda vez para la tecla A.