JAVASCRIPT
Capítulo 6

Lenguajes de Marcas y Sistemas de Gestión de Información

Curso 2019/2020

Capítulo 6: EVENTOS

1. Introducción
2. Modelos de Eventos
3. Modelo Básico de Eventos
4. Obteniendo Información del Evento

1. INTRODUCCIÓN

Todas las aplicaciones y scripts que se han creado se **ejecutan** desde la primera instrucción hasta la última de **forma secuencial**. Gracias a `if`, `for` o `while` es posible modificar este comportamiento y repetir partes del script y saltarse otras en función de ciertas condiciones.

Este tipo de aplicaciones son poco útiles, ya que no interactúan con los usuarios y no pueden responder a los diferentes eventos que se producen durante la ejecución de una aplicación → Crear aplicaciones web con JavaScript que utilizan el **modelo de programación basada en eventos**.

1. INTRODUCCIÓN

Los scripts se dedican a esperar a que **el usuario interactúe** con la aplicación. Después, **el script responde** a la acción del usuario procesando esa información y generando un resultado. Los eventos posibilitan que los usuarios transmitan información a los programas:

- Pulsación de una tecla.
- Pinchar o mover el ratón.
- Seleccionar un elemento de un formulario.
- Redimensionar la ventana del navegador.

JavaScript permite **asignar una función** a cada uno de los eventos. De esta forma, cuando se produce cualquier evento, JavaScript ejecuta su función asociada. Se denominan funciones ***event handlers***.

2. MODELOS DE EVENTOS

A pesar de los estándares creados, hay navegadores **no los soportan** o **los ignoran**. Las principales incompatibilidades se producen en XHTML, CSS y JavaScript → La **incompatibilidad** más importante se da precisamente en el **modelo de eventos del navegador**. Existen tres modelos diferentes para manejar los eventos dependiendo del navegador en el que se ejecute la aplicación.

- **Modelo Básico de Eventos**: único modelo que es compatible en todos los navegadores.
- **Modelo de Eventos Estándar**: versión avanzada del anterior. Todos los navegadores modernos lo incluyen, salvo Internet Explorer.
- **Modelo de Eventos de Internet Explorer**: utiliza su propio modelo de eventos, que es similar pero incompatible con el modelo estándar.

3. MODELO BÁSICO DE EVENTOS: TIPOS DE EVENTOS

Un mismo tipo de evento puede estar definido para **varios elementos** XHTML diferentes y un mismo elemento XHTML puede tener asociados **varios eventos** diferentes.

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`.

3. MODELO BÁSICO DE EVENTOS: TIPOS DE EVENTOS

Evento	Descripción	Elementos para los que está definido
`onblur`	Deseleccionar el elemento	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
`onchange`	Deseleccionar un elemento que se ha modificado	`<input>`, `<select>`, `<textarea>`
`onclick`	Pinchar y soltar el ratón	Todos los elementos
`ondblclick`	Pinchar dos veces seguidas con el ratón	Todos los elementos
`onfocus`	Seleccionar un elemento	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
`onkeydown`	Pulsar una tecla (sin soltar)	Elementos de formulario y `<body>`
`onkeypress`	Pulsar una tecla	Elementos de formulario y `<body>`
`onkeyup`	Soltar una tecla pulsada	Elementos de formulario y `<body>`
`onload`	La página se ha cargado completamente	`<body>`
`onmousedown`	Pulsar (sin soltar) un botón del ratón	Todos los elementos
`onmousemove`	Mover el ratón	Todos los elementos

3. MODELO BÁSICO DE EVENTOS: TIPOS DE EVENTOS

`onmouseout`	El ratón *"sale"* del elemento (pasa por encima de otro elemento)	Todos los elementos
`onmouseover`	El ratón *"entra"* en el elemento (pasa por encima del elemento)	Todos los elementos
`onmouseup`	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
`onreset`	Inicializar el formulario (borrar todos sus datos)	`<form>`
`onresize`	Se ha modificado el tamaño de la ventana del navegador	`<body>`
`onselect`	Seleccionar un texto	`<input>`, `<textarea>`
`onsubmit`	Enviar el formulario	`<form>`
`onunload`	Se abandona la página (por ejemplo al cerrar el navegador)	`<body>`

3. MODELO BÁSICO DE EVENTOS: MANEJADORES DE EVENTOS

Para que los eventos resulten útiles, se deben **asociar funciones o código JavaScript a cada evento**. 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***":

- Manejadores como **Atributos** de los Elementos XHTML.
- Manejadores como **Funciones JavaScript** Externas.
- Manejadores **Semánticos**.

3.1. MANEJADORES DE EVENTOS COMO ATRIBUTOS XHTML

El código se incluye en un atributo del propio elemento XHTML. *Mostrar un mensaje cuando el usuario pinche con el ratón sobre un botón*:

```
<input type="button" value="Pinchame y verás"
onclick="alert('Gracias por pinchar');" />
```

En este método, se definen atributos XHTML con el **mismo nombre** que los eventos que se quieren manejar. Sólo queremos controlar el evento de pinchar con el ratón (`onclick`) → El elemento XHTML para el que se quiere definir este evento, debe incluir un atributo llamado `onclick`.

El **contenido del atributo** es una cadena de texto que contiene todas las instrucciones JavaScript que se ejecutan cuando se produce el evento. En este caso, el código JavaScript es muy sencillo (`alert('Gracias por pinchar');`), ya que solamente se trata de mostrar un mensaje.

3.1. MANEJADORES DE EVENTOS COMO ATRIBUTOS XHTML

Cuando un usuario pincha sobre el elemento `<div>` se muestra un mensaje y cuando pasa el ratón por encima del elemento, se muestra otro mensaje:

```
<div onclick="alert('Has pinchado con el ratón');"
onmouseover="alert('Acabas de pasar el ratón por
encima');">

  Puedes pinchar sobre este elemento o simplemente
pasar el ratón por encima

</div>
```

3.1. MANEJADORES DE EVENTOS COMO ATRIBUTOS XHTML

```
<body onload="alert('La página se ha cargado completamente');">

    ...

</body>
```

El mensaje anterior se muestra **después** de que la página se haya **cargado completamente**, es decir, después de que se haya descargado su código HTML, sus imágenes y cualquier otro objeto incluido en la página.

3.2. 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**.

Queremos que cuando el usuario pasa el ratón por encima del `<div>`, el color del borde se muestre de color negro. Cuando el ratón sale del `<div>`, se vuelve a mostrar el borde con el color gris claro original.

```
<div id="contenidos" style="width:150px; height:60px;
border:thin solid silver">

  Sección de contenidos...

</div>
```

3.2. MANEJADORES DE EVENTOS Y VARIABLE `this`

Si no se utiliza la variable `this`, el código necesario para modificar el color de los bordes, sería el siguiente:

```
<div id="contenidos" style="width:150px; height:60px;
border:thin solid silver"
onmouseover="document.getElementById('contenidos').style.borderColor='black';"
onmouseout="document.getElementById('contenidos').style.borderColor='silver';">

  Sección de contenidos...

</div>
```

3.2. MANEJADORES DE EVENTOS Y VARIABLE `this`

El código anterior es **demasiado largo** y demasiado propenso a **cometer errores**. Así, el ejemplo anterior se puede reescribir de la siguiente manera:

```
<div id="contenidos" style="width:150px; height:60px;
border:thin solid silver"
onmouseover="this.style.borderColor='black';"
onmouseout="this.style.borderColor='silver';">

  Sección de contenidos...

</div>
```

3.3. MANEJADORES DE EVENTOS COMO FUNCIONES EXTERNAS

Si se realizan **aplicaciones complejas**, como por ejemplo la validación de un formulario, es aconsejable **agrupar** todo el **código JavaScript** en una **función externa** y llamar a esta función desde el elemento XHTML.

Siguiendo con el ejemplo anterior que muestra un mensaje al pinchar sobre un botón:

```
<input type="button" value="Pinchame y verás" onclick="alert('Gracias por pinchar');" />
```

3.3. MANEJADORES DE EVENTOS COMO FUNCIONES EXTERNAS

```
function muestraMensaje() {
  alert('Gracias por pinchar');
}

<input type="button" value="Pinchame y verás"
onclick="muestraMensaje()" />
```

Se extraen todas las instrucciones de JavaScript agrupándolas en una función externa. En el atributo del elemento XHTML se incluye el nombre de la función.

La **llamada a la función** se realiza de la forma habitual, indicando su **nombre** seguido de los paréntesis y de forma opcional, incluyendo todos los **argumentos y parámetros** que se necesiten.

3.3. MANEJADORES DE EVENTOS COMO FUNCIONES EXTERNAS


```
function resalta(elemento) {
  switch(elemento.style.borderColor) {
    case 'silver':
    case 'silver silver silver silver':
    case '#c0c0c0':
      elemento.style.borderColor = 'black';
      break;
    case 'black':
    case 'black black black black':
    case '#000000':
      elemento.style.borderColor = 'silver';
      break;
  }
}
```

```
<div style="width:150px; height:60px; border:thin solid silver" onmouseover="resalta(this)" onmouseout="resalta(this)">
  Sección de contenidos...
</div>
```

3.4. MANEJADORES DE EVENTOS SEMANTICOS

Una de las **buenas prácticas** en el diseño de páginas y aplicaciones web es la **separación** de los contenidos (XHTML), aspecto (CSS) y comportamiento o programación (JavaScript).

Afortunadamente, existe un método alternativo para **definir** los **manejadores de eventos** de JavaScript. Esta técnica es una evolución del método de las funciones externas, ya que se basa en **utilizar las propiedades DOM** de los elementos XHTML para asignar todas las funciones externas que actúan de manejadores de eventos.

3.4. MANEJADORES DE EVENTOS SEMANTICOS

```
<input id="pinchable" type="button" value="Pinchame y verás"
onclick="alert('Gracias por pinchar');" />
```

Se puede transformar en:

```
// Función externa
function muestraMensaje() {
  alert('Gracias por pinchar');
}

// Asignar la función externa al elemento
document.getElementById("pinchable").onclick = muestraMensaje;

// Elemento XHTML
<input id="pinchable" type="button" value="Pinchame y verás" />
```

1. **Asignar un identificador único al elemento XHTML mediante el atributo `id`.**
2. **Crear una función de JavaScript encargada de manejar el evento.**
3. **Asignar la función externa al evento correspondiente en el elemento deseado.**

3.4. MANEJADORES DE EVENTOS SEMANTICOS

El evento se asigna a la función externa mediante su nombre **sin paréntesis**. Lo más importante es indicar solamente el nombre de la función:

```
document.getElementById("pinchable").onclick = muestraMensaje;
```

Si se añaden los paréntesis después del nombre de la función, en realidad se está ejecutando la función y guardando el valor devuelto por la función en la propiedad `onclick` de elemento.

```
// Asignar una función externa a un evento de un elemento
document.getElementById("pinchable").onclick = muestraMensaje;


// Ejecutar una función y guardar su resultado en una propiedad de un elemento
document.getElementById("pinchable").onclick = muestraMensaje(); //ERROR
```

3.4. MANEJADORES DE EVENTOS SEMANTICOS

El único **inconveniente** de este método es que la **página** se debe **cargar completamente** antes de que se puedan utilizar las funciones DOM que asignan los manejadores a los elementos XHTML.

Una de las formas más sencillas de **asegurar que cierto código se va a ejecutar** después de que la página se cargue por completo es utilizar el evento `onload`:

```
window.onload = function() {

  document.getElementById("pinchable").onclick =
muestraMensaje;

}
```

4. OBTENIENDO INFORMACIÓN DEL EVENTO (objecto `event`)

Los manejadores de eventos requieren **información adicional** para procesar sus tareas. Si una función procesa el evento `onclick`, quizás necesite saber en que posición estaba el ratón en el momento de pinchar el botón.

El caso más habitual en el que es necesario conocer información adicional sobre el evento es el de los eventos **asociados al teclado**. Es muy importante conocer la tecla que se ha pulsado: diferenciar las teclas normales de las teclas especiales (ENTER, tabulador, Alt, Ctrl., etc…).

4. OBTENIENDO INFORMACIÓN DEL EVENTO (objecto `event`)

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

Todos los navegadores modernos crean automáticamente 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.

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

4.1. INFORMACIÓN SOBRE EL EVENTO

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;
```

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

4.1. INFORMACIÓN SOBRE EL EVENTO

```
function resalta(elEvento) {
  var evento = elEvento || window.event;
  switch(evento.type) {
    case 'mouseover':
      this.style.borderColor = 'black';
      break;
    case 'mouseout':
      this.style.borderColor = 'silver';
      break;
  }
}

window.onload = function() {
  document.getElementById("seccion").onmouseover = resalta;
  document.getElementById("seccion").onmouseout = resalta;
}

<div id="seccion" style="width:150px; height:60px; border:thin solid silver">
  Sección de contenidos...
</div>
```

4.2. INFORMACIÓN SOBRE LOS EVENTOS DE TECLADO

Los eventos relacionados con el teclado son los más difíciles de manejar. Existen muchas **diferencias** entre navegadores, teclados y sistemas operativos, principalmente debido a las diferencias entre **idiomas**.

Existen **tres eventos** diferentes para las pulsaciones de teclas:

- `onkeydown`: **corresponde al hecho de pulsar una tecla y no soltarla.**
- `onkeypress`: **es la propia pulsación de la tecla.**
- `onkeyup`: **hace referencia al hecho de soltar una tecla que estaba pulsada.**

4.2. INFORMACIÓN SOBRE LOS EVENTOS DE TECLADO

Lista con todas las propiedades diferentes de todos los eventos de teclado:

- Evento `keydown`:
  - Mismo comportamiento en todos los navegadores:
    - Propiedad `keyCode`: código interno de la tecla
    - Propiedad `charCode`: no definido
- Evento `keypress`:
  - Internet Explorer:
    - Propiedad `keyCode`: el código del carácter de la tecla que se ha pulsado
    - Propiedad `charCode`: no definido
  - Resto de navegadores:
    - Propiedad `keyCode`: para las teclas normales, no definido. Para las teclas especiales, el código interno de la tecla.
    - Propiedad `charCode`: para las teclas normales, el código del carácter de la tecla que se ha pulsado. Para las teclas especiales, 0.
- Evento `keyup`:
  - Mismo comportamiento en todos los navegadores:
    - Propiedad `keyCode`: código interno de la tecla
    - Propiedad `charCode`: no definido

4.2. INFORMACIÓN SOBRE LOS EVENTOS DE TECLADO

A continuación se incluye un script que muestra toda la información sobre los tres eventos de teclado:

```
window.onload = function() {
  document.onkeyup = muestraInformacion;
  document.onkeypress = muestraInformacion;
  document.onkeydown = muestraInformacion;
}

function muestraInformacion(elEvento) {
  var evento = window.event || elEvento;

  var mensaje = "Tipo de evento: " + evento.type + "<br>" +
                "Propiedad keyCode: " + evento.keyCode + "<br>" +
                "Propiedad charCode: " + evento.charCode + "<br>" +
                "Carácter pulsado: " + String.fromCharCode(evento.charCode);

  info.innerHTML += "<br>---------------------------------------<br>" + mensaje
}

...

<div id="info"></div>
```

4.3. INFORMACIÓN SOBRE LOS EVENTOS DE RATÓN

La información más relevante sobre los eventos relacionados con el ratón es la de las **coordenadas de la posición del puntero del** ratón. El origen de las coordenadas siempre se encuentra en la esquina superior izquierda, el punto que se toma como referencia de las coordenadas puede variar.

De esta forma, es posible obtener la posición del ratón respecto de:

- La pantalla del ordenador.
- La ventana del navegador (más sencillas de obtener).
- La propia página HTML.

Se obtienen mediante las propiedades `clientX` **y** `clientY`.

4.3. INFORMACIÓN SOBRE LOS EVENTOS DE RATÓN

```
function muestraInformacion(elEvento) {

  var evento = elEvento || window.event;

  var coordenadaX = evento.clientX;

  var coordenadaY = evento.clientY;

  alert("Has pulsado el ratón en la posición: " + coordenadaX + ", "
+ coordenadaY);

}

document.onclick = muestraInformacion;
```

Las coordenadas de la posición del puntero del ratón respecto de la pantalla completa del ordenador del usuario se obtienen de la misma forma, mediante las propiedades `screenX` **y** `screenY`**:**

```
var coordenadaX = evento.screenX;

var coordenadaY = evento.screenY;
```

4.3. INFORMACIÓN SOBRE LOS EVENTOS DE RATÓN

En muchas ocasiones, es necesario obtener otro par de coordenadas diferentes: las que corresponden a la **posición del ratón respecto del origen de la página**.

Estas coordenadas no siempre coinciden con las coordenadas respecto del origen de la ventana del navegador, ya que el usuario puede hacer `scroll` sobre la página web.

Internet Explorer no proporciona estas coordenadas de forma directa, mientras que el resto de navegadores sí que lo hacen.

Es necesario detectar si el navegador es de tipo Internet Explorer y en caso afirmativo realizar un cálculo sencillo.

4.3. INFORMACIÓN SOBRE LOS EVENTOS DE RATÓN

```
// Detectar si el navegador es Internet Explorer
var ie = navigator.userAgent.toLowerCase().indexOf('msie')!=-1;

if(ie) {
  coordenadaX = evento.clientX + document.body.scrollLeft;
  coordenadaY = evento.clientY + document.body.scrollTop;
}
else {
  coordenadaX = evento.pageX;
  coordenadaY = evento.pageY;
}
alert("Has pulsado el ratón en la posición: " + coordenadaX +
", " + coordenadaY + " respecto de la página web");
```

VENTANA DEL NAVEGADOR + DESPLAZAMIENTO

ORIGEN DE LA PÁGINA