Modelos de eventos. Eventos en línea

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Ene
- iPasen - cjaedia071@g.educaand.es

Carmelo José Jaén Díaz

Índice

OBJETIVO

- Descubrir elementos propios de JavaScript como los eventos
- Registrar en el front-end información de navegación, usuario, etc., mediante cookies o mediante el almacenamiento local que ofrece el estándar HTML5.

GLOSARIO

Expresión regular. Secuencia de caracteres que forman un patrón determinado, generalmente un patrón de búsqueda.

Listener. Código responsable de controlar eventos. Están a la escucha (de ahí su nombre) y, cuando ocurre un evento, se ejecuta el código que el programador haya implementado.

INTRODUCCIÓN

Antes de empezar a hablar de **modelos de eventos**, conviene saber **qué es un evento**.

En la programación tradicional, los programas se ejecutan siguiendo una secuencia de instrucciones que van de arriba a abajo. No obstante, actualmente es posible trabajar con la **programación basada en eventos**, de manera que ciertas **instrucciones se ejecutan en el momento que ocurre algo: un evento**.

Javascript admite los dos tipos de métodos de programación. Hasta ahora habíamos trabajado la programación tradicional; ahora nos adentraremos en el mundo de los eventos.

INTRODUCCIÓN

Hay **muchos tipos de eventos en Javascript**, normalmente asociados a las acciones que realiza un usuario mientras interactúa con el ordenador mediante el ratón o el teclado; aunque hay otros eventos que se ejecutan automáticamente, como el evento `onload`, que sucede cuando termina de cargarse una página web. Podéis ver un listado con todos los eventos en Mozilla MDN Web Docs.

En esta lección vamos a ver el **modelo de eventos en línea**, o también conocido como **eventos en atributos HTML**. Es el modo **más sencillo y menos profesional o recomendado** de indicar qué código debe ejecutarse cuando se produzca un evento; dicho código se asocia a un atributo del elemento HTML.

Importante: El modelo de eventos en línea no se debe utilizar porque implica mezclar código HTML con Javascript, práctica que no es muy recomendable... ¡evítalo!

INTRODUCCIÓN

Un ejemplo sería el siguiente, en que se muestra una ventana de diálogo al pulsar un botón:

```
<input type="button" value="¡Haz clic sobre mí!" onclick="alert('¡Hola mundo!');" />
```

GLOSARIO DE EVENTOS

EVENTO:

- Mecanismo que se acciona cuando el usuario realiza un cambio sobre la página web.
- Capturar un evento es programar una acción para que se realice una tarea.
- El encargado de gestionar los eventos es el `DOM` (Document Object Model).

MANEJADOR:

- Acción que se va a manejar. Por ejemplo, en el evento *click* (hacer click con el ratón) el manejador es `onClick`.

En W3SCHOOLS podemos encontrar un listado de todos los tipos de eventos que existen.

MODELO DE REGISTRO DE EVENTOS EN LÍNEA

Cada elemento XHTML tiene sus posibles eventos como propiedades: puede tener un evento de cada tipo. Es decir, por ejemplo puede tener como propiedades `id`, `value`, ... y también todos los eventos como `onclick`, `onmouseout`, `onmouseover`, ...

La nomenclatura del evento sigue la regla: empieza por “on” seguido del nombre de la acción.

A continuación, vamos a ver los tipos de manejadores que nos podemos encontrar.

MANEJADORES COMO ATRIBUTO DE UNA ETIQUETA XHTML

Se muestra el comportamiento de algunos eventos mediante un ejemplo, para ello se implementa una etiqueta de título de nivel tres con su identificador y texto:

```
<h3 id="cabecera">Pulsa aquí para ver lo que se ejecuta</h3>
```

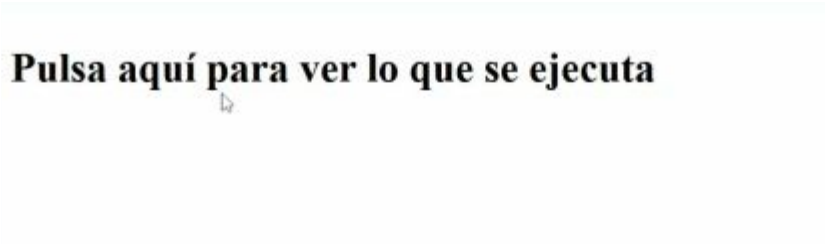
A la que le añadimos una serie de eventos mediante manejadores:

- `onclick="this.innerHTML='Javascript en XHTML'"` : se ejecuta en caso de clicar sobre el elemento (en este caso modifica el texto de la propia etiqueta mediante la palabra reservada `this`).
- `onmouseover="this.style.background='red'"` : se ejecuta cuando se coloca el cursor del ratón sobre el elemento (en este caso modifica el estilo de la web cambiando a rojo el color de fondo).
- `onmouseout="this.style.background='white'"` : se ejecuta cuando desplazamos el cursor del ratón fuera del elemento (en este caso modifica el estilo de la web cambiando a blanco el color de fondo).

MANEJADORES COMO ATRIBUTO DE UNA ETIQUETA XHTML

Finalmente, la etiqueta completa quedaría:

```
<h3 id="cabecera"
onclick="this.innerHTML='Javascript en
XHTML'"
onmouseover="this.style.background='red'
"
onmouseout="this.style.background='white
'">Pulsa aquí para ver lo que se
ejecuta</h3>
```



Puedes apreciar el comportamiento de este código en el siguiente link o en la imagen adjunta:

https://codepen.io/Carmelo-Jos-Ja-n-D-az/pen/jENvqao

MANEJADORES COMO ATRIBUTO DE UNA ETIQUETA XHTML

La implementación anterior sería la más recomendable y correcta, aunque análogamente podemos hacer referencia a la etiqueta accediendo a ella a través del DOM:

```
<h3 id="cabecera"
onclick="document.getElementById("cabecera").innerHTML="...">...</h3>
```

Recuerda: La manera correcta es usar la palabra reservada `this` para *apuntar* al propio elemento.

MANEJADORES COMO FUNCIONES EXTERNAS

Se muestra el comportamiento de algunos eventos mediante un ejemplo, para ello se implementa una etiqueta de título de nivel tres con su identificador y texto:

```
<h3 id="cabecera">Pulsa aquí para ver lo que se ejecuta</h3>
```

A la que le añadimos una serie de eventos mediante manejadores vinculado a una función externa en JS:

- `onclick="cambiar(this)"` : invoca a la función `cambiar()` pasándole como argumento la palabra reservada `this` (apunta al propio elemento).

MANEJADORES COMO FUNCIONES EXTERNAS

El siguiente código debería implementarse en un fichero .js externo, vinculado al documento HTML. Por simplicidad se incrusta el código JS directamente en el fichero HTML (no recomendable).

```
<script>
        function cambiar(elem) {
            elem.innerHTML = "Javascript en XHTML y función externa";
        }
</script>
```

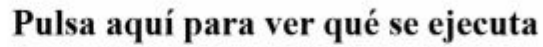
MANEJADORES COMO FUNCIONES EXTERNAS

Finalmente, la etiqueta completa quedaría:

```
<h3 onclick="cambiar(this)">Pulsa aquí
para ver qué se ejecuta</h3>
```

Puedes apreciar el comportamiento de este código en el siguiente link o en la imagen adjunta:

https://codepen.io/Carmelo-Jos-Ja-n-D-az/pen/jENvqXN



ACCIONES QUE DESENCADENAN VARIOS EVENTOS

Hay acciones que desencadenan varios eventos. Por ejemplo, cuando clicamos un botón de tipo `submit` (vinculado a formularios) desencadena automáticamente los eventos `onmousedown`, `onclick`, `onmouseup` y `onsubmit`.

Por tanto, al ser un botón cuya función principal es enviar los datos del formulario a la dirección web especificada en la etiqueta `<form>`, para evitar que el navegador ejecute la acción por defecto necesitamos añadir "**`return false;`**" al final del código HTML.

A continuación, se ilustra con un ejemplo.

ACCIONES QUE DESENCADENAN VARIOS EVENTOS

A continuación, se implementa una etiqueta de enlace (cuya función principal es la de dirigirnos a la dirección web especificada):

```
<a href="http://www.google.com">Pulsa aquí para ver qué se ejecuta</a>
```

A la que le añadimos una serie de eventos mediante manejadores vinculado a una función externa en JS:

- `onclick="alertar(); return false;"` : invoca a la función `alertar()`. Al incluir el return false, indicamos que no se ejecute la función por defecto (dirigirnos a la dirección www.google.com).

ACCIONES QUE DESENCADENAN VARIOS EVENTOS

El siguiente código debería implementarse en un fichero .js externo, vinculado al documento HTML. Por simplicidad se incrusta el código JS directamente en el fichero HTML (no recomendable).

```
<script>
    function alertar() {
        alert("Vamos a Google");
    }
</script>
```

ACCIONES QUE DESENCADENAN VARIOS EVENTOS

Finalmente, la etiqueta completa quedaría:

```
<a href="http://www.google.com"
onclick="alertar(); return
false;">Pulsa aquí para ver qué se
ejecuta</a>
```

Puedes apreciar el comportamiento de este código en el siguiente link o en la imagen adjunta:

https://codepen.io/Carmelo-Jos-Ja-n-D-az/pen/MYqqeeb

DECISIÓN POR EL USUARIO EJECUCIÓN VARIOS EVENTOS

También es posible dejar la elección de ejecutar o no ambas funciones al usuario, por ejemplo mediante un elemento `confirm()`. La etiqueta de enlace quedaría:

```
<a href="http://www.google.com" onclick="return preguntar();">Pulsa aquí
para ver qué se ejecuta</a>
```

Y el código JS asociado:

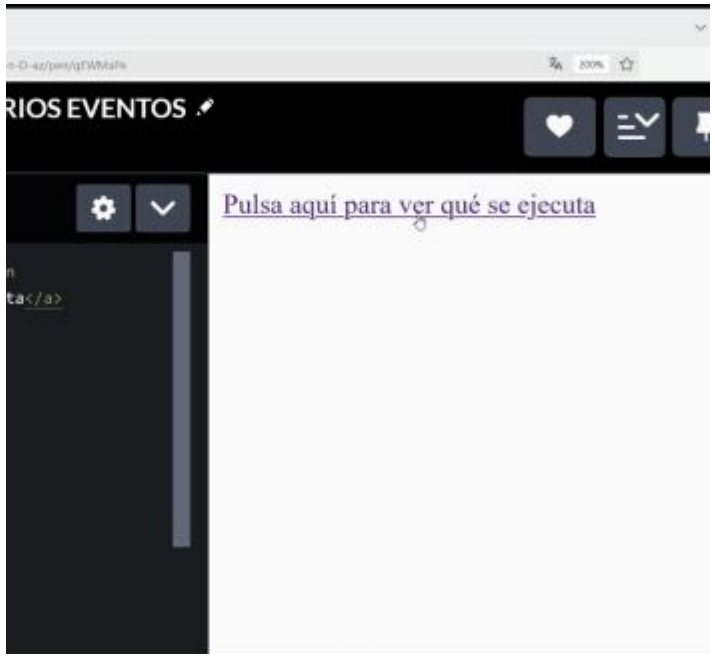
```
<script>
    function preguntar() {
        return confirm("¿Deseas ir a Google?");
    }
</script>
```

- Si el usuario pulsa en *Aceptar*, el `confirm()` devolverá un `true` y la etiqueta nos dirigirá a www.google.com.
- En caso contrario, el `confirm()` devolverá un `false` y estaremos en la casuística anterior (no dirige a www.google.com).

DECISIÓN POR EL USUARIO
EJECUCIÓN VARIOS EVENTOS

Puedes apreciar el comportamiento de este código en el siguiente link o en la imagen adjunta:

https://codepen.io/Carmelo-Jos-Ja-n-D-az/pen/qEWMaPe



EVENTO *onload*

El elemento `onload` nos permite, entre otras funcionalidades, indicarle a una página web que no cargue un fichero JavaScript hasta que no haya terminado de cargar todo el contenido de un documento HTML.

Para ello, incluimos la etiqueta:

```
<body onload="funcion_ejecutar"> →
```

La `funcion_ejecutar` se ejecutará una vez se haya cargado todo el contenido del documento HTML.

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