Modelo de eventos del W3C

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Ene

Carmelo José Jaén Díaz

- iPasen - cjaedia071@g.educaand.es

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

En las lecciones anteriores trabajamos con el modelo de eventos en línea, o **eventos en atributos HTML**, y también el modelo de eventos tradicional.

Aunque ambos son perfectamente válidos, **el primero** no es recomendable porque **mezcla código Javascript con HTML**, y **tanto el primero como el segundo no permiten asignar a un mismo elemento más de un evento**.

Esto lo podemos suplir con el **modelo de eventos del W3C**. Este modelo trabaja con lo que definiríamos como **escuchadores de eventos**. Para que te hagas una idea es como poner a los elementos HTML que queramos un *sensor* que detecta cada vez que hacemos clic o cada vez que ponemos el ratón encima. Y luego decidimos qué hacer cada vez que ocurra eso, es decir, qué función ejecutar.

INTRODUCCIÓN

Para añadir esos escuchadores de eventos, utilizamos el método `addEventListener` con dos o tres atributos: el primero lleva el **nombre del evento** que estamos *escuchando*, el segundo **la función** (que puede ser anónima o no) y el tercero, opcional, tiene que ver con la **fase de burbujeo** (explicada en lecciones posteriores). La sintaxis es la siguiente:

```
elemento.addEventListener("<evento_sin_on>",<funcion>,<false|true>);
```

Recuerda que en el modelo de eventos tradicional no es posible hacer esto:

```
document.getElementById("parrafo").onclick = hazEsto;
document.getElementById("parrafo").onclick = hazLoOtro;
```

INTRODUCCIÓN

Sin embargo, en el **modelo de eventos del W3C** podríamos asignar dos funciones diferentes cada vez que hacemos clic sobre el elemento con `id="parrafo"` de la siguiente manera:

```
document.getElementById("parrafo").addEventListener("click", hazEsto);
document.getElementById("parrafo").addEventListener("click", hazLoOtro);
```

MODELO DE EVENTOS AVANZADOS DEL W3C

Para estudiar cómo es la sintaxis de este modelo, se plantea el siguiente escenario:

En primer lugar, se implementa una etiqueta de título de nivel tres con el siguiente identificador:

```
<h3 id="w3c">Modelo del W3C</h3>
```

Y otra similar, que usaremos más adelante:

```
<h3 id="w3canonima">Modelo del W3C con funciones anónimas</h3>
```

Al igual que pasa con el modelo de eventos tradicional, este modelo separa el código HTML del código JavaScript, por lo que implementamos esta lógica en un fichero externo. Por simplicidad didáctica, en este ejemplo se implementa entre etiquetas script.

MODELO DE EVENTOS AVANZADOS DEL W3C

Este script realiza las siguientes funcionalidades:

- En primer lugar, en este modelo, tenemos que identificar el elemento: `document.getElementById("id").`
- Una vez identificado el elemento, añadimos el método `addEventListener` con los siguientes argumentos:
    - El tipo de evento entre comillas sin poner *on* delante: `"<evento_sin_on>"`.
    - El nombre de la función sin comillas sin paréntesis: `<funcion>`.
    - Por último la fase de burbujeo, que determina en qué momento se ejecuta este evento: `<false|true>`. De momento, indicaremos siempre `false`.

MODELO DE EVENTOS AVANZADOS DEL W3C

Por tanto, nuestro script resultaría:

```
<script>
   document.getElementById("w3c").addEventListener("click", saludarUnaVez,
   false);

   function saludarUnaVez() {
      alert("¡Hola, caracola!");
    }
</script>
```

MODELO DE EVENTOS AVANZADOS DEL W3C

Este modelo de eventos nos permite añadir todas las acciones que queramos (todos los manejadores de eventos que consideremos), para cada evento. Es decir, podríamos implementar adicionalmente a la ejecución del `alert()` mediante el evento `onclick`, la funcionalidad `colorearse()`, también mediante el evento `onclick` y no se sobreescribiría.

El código completo resulta:

```
<script>
   document.getElementById("w3c").addEventListener("click", saludarUnaVez,
   false);
   document.getElementById("w3c").addEventListener("click", colorearse,
   false);
```

MODELO DE EVENTOS AVANZADOS DEL W3C

```
    function saludarUnaVez() {
        alert("¡Hola, caracola!");
     }

    function colorearse() {
        document.getElementById("w3c").style.color = "red";
    }
</script>
```

MODELO DE EVENTOS AVANZADOS DEL W3C

También podríamos añadir otra acción (otro manejador de eventos) accionado por otro evento, como por ejemplo `onmouseover`, pasar el cursor del ratón por encima.

El código completo resulta:

```
<script>
   document.getElementById("w3c").addEventListener("click", saludarUnaVez,
   false);
   document.getElementById("w3c").addEventListener("click", colorearse,
   false);
   document.getElementById("w3c").addEventListener("mouseover", fondo,
   false);
```

MODELO DE EVENTOS AVANZADOS DEL W3C

```
    function saludarUnaVez() {
        alert("¡Hola, caracola!");
     }

    function colorearse() {
        document.getElementById("w3c").style.color = "red";
    }

    function fondo() {
        document.getElementById("w3c").style.background = "blue";
    }
</script>
```

MODELO DE EVENTOS AVANZADOS DEL W3C

Supongamos ahora, que queremos eliminar uno de los `addEventListener()`, por ejemplo el asociado a la función `saludarUnaVez()`. Es decir, solo queremos que se ejecute la primera vez que se hace clic y después se deshabilite esta funcionalidad. Para ello, modificamos dicha función, añadiendo:

```
document.getElementById("w3c").removeEventListener("click", saludarUnaVez);
```

Debemos indicar tanto el evento, como su función asociada ya que para el mismo evento podemos tener varias funciones. Resultando finalmente esa función:

```
function saludarUnaVez() {
    alert("¡Hola, caracola!");
    document.getElementById("w3c").removeEventListener("click",
    saludarUnaVez);
 }
```

MODELO DE EVENTOS AVANZADOS DEL W3C

Funciones anónimas

Si quisiéramos emplear funciones anónimas, aquellas que no han sido declaradas con un nombre:

- En primer lugar, tenemos que identificar el elemento: `document.getElementById("id").`
- Una vez identificado el elemento, añadimos el método `addEventListener` con los siguientes argumentos:
  - El tipo de evento entre comillas sin poner *on* delante: `"<evento_sin_on>"`.
  - La función sin comillas sin paréntesis y su código: `function(){<codigo_funcion})`.
  - Por último la fase de burbujeo, que determina en qué momento se ejecuta este evento: `<false|true>`. De momento, indicaremos siempre `false`.

MODELO DE EVENTOS AVANZADOS DEL W3C

Funciones anónimas

Un ejemplo de implementación sería:

```
document.getElementById("w3canonima").addEventListener("click",
function () {
   this.style.background = "#C0C0C0";
});
```

En este caso el puntero `this`, hace referencia al elemento (el cual se identifica mediante la `id="w3canonima"`) que tiene asociado el método `addEventListener()`.

MODELO DE EVENTOS AVANZADOS DEL W3C

Navegadores anteriores a MS IE9

Este es el método recomendable a la hora de implementar eventos y manejadores en nuestras aplicaciones web.

Este modelo de registros de eventos se utilizan en todos los navegadores modernos pero no funciona con navegadores de Microsoft, como Internet Explorer, anteriores a la versión IE9. Para estos últimos necesitamos otro modelo de registros de eventos.

A modo de curiosidad o si deseas tener compatibilidad con estos navegadores, en este caso se utiliza el método `attachEvent()`.

MODELO DE EVENTOS AVANZADOS DEL W3C
Navegadores anteriores a MS IE9

La sintaxis es similar, tal y como podemos ver en el siguiente ejemplo:

```
document.getElementById("parrafo").attachEvent("click", hazEsto);
```

Se proporciona a continuación, un escenario análogo al estudiado para el método `addEventListener()`.

Se parte de los siguientes elementos HTML:

```
<h3 id="ms">Modelo de Microsoft</h3>
<h3 id="msanonima">Modelo de Microsoft</h3>
```

MODELO DE EVENTOS AVANZADOS DEL W3C
Navegadores anteriores a MS IE9

Donde la funcionalidad JavaScript sería:

- En primer lugar, tenemos que identificar el elemento: `document.getElementById("id").`
- Una vez identificado el elemento, añadimos el método `attachEvent` con los siguientes argumentos:
  - El tipo de evento entre comillas incluyendo *on* delante: `"<evento_con_on>"`.
  - El nombre de la función sin comillas sin paréntesis: `<funcion>`.

Resultando el código de la siguiente manera:

MODELO DE EVENTOS AVANZADOS DEL W3C

Navegadores anteriores a MS IE9

```
<script>
    document.getElementById("ms").attachEvent("onclick", saludarUnaVez);
    document.getElementById("ms").attachEvent("onclick", colorearse);
    document.getElementById("ms").attachEvent("onmouseover", fondo);

    function saludaUnaVez() {
        alert("¡Hola, caracola!");
        //Nos permite deshabilitar el evento una vez ejecutado
        document.getElementById("ms").detachEvent("onclick", saludarUnaVez);
    }
```

MODELO DE EVENTOS AVANZADOS DEL W3C
Navegadores anteriores a MS IE9

```
    function colorearse() {
        document.getElementById("ms").style.color = "red";
    }

    function fondo() {
        document.getElementById("ms").style.backgroundColor = "blue";
    }
</script>
```

MODELO DE EVENTOS AVANZADOS DEL W3C
Navegadores anteriores a MS IE9

Para el caso de funciones anónimas:

- En primer lugar, tenemos que identificar el elemento: `document.getElementById("id").`
- Una vez identificado el elemento, añadimos el método `attachEvent` con los siguientes argumentos:
  - El tipo de evento entre comillas incluyendo *on* delante: `"<evento_con_on>"`.
  - La función sin comillas sin paréntesis y su código: `function(){<codigo_funcion})`.

MODELO DE EVENTOS AVANZADOS DEL W3C
Navegadores anteriores a MS IE9

Resultando el código de la siguiente manera:

```
document.getElementById("msanonima").attachEvent("onclick", function () {
        this.style.backgroundColor = "#C0C0C0";
    });
```

En este caso el puntero **this**, hace referencia al elemento (el cual se identifica mediante la **id**="msanonima") que tiene asociado el método attachEvent().