Modelo de eventos tradicional

- 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 la lección anterior vimos cómo funcionaba el **modelo de eventos en línea**, o **eventos en atributos HTML**, que permitía incluir los eventos en Javascript directamente sobre la etiqueta HTML. Como ya viste anteriormente, esta opción no es la más indicada puesto que mezcla código en diferentes lenguajes dentro del mismo archivo.

En esta lección, por el contrario, trabajaremos con el modelo de eventos tradicional, que consiste en **aplicar sobre un elemento** seleccionado a través de Javascript, un evento de la forma "***onnombreevento***" (`onclick`, `onmouseover`, `onblur`...).

Desde el punto de vista de la organización, esta forma es mucho más correcta, ya que **separa el código HTML del código Javascript.**

INTRODUCCIÓN

Sin embargo, esta opción **tampoco es la más recomendada**, puesto que únicamente permite asociar **un solo evento del mismo tipo a un elemento**. Es decir, si hiciéramos lo siguiente:

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

Al hacer clic sobre el párrafo con id "*parrafo*" únicamente se ejecutaría la función *hazLoOtro()*.

Por eso la **mejor opción** es siempre utilizar el **Modelo de eventos de W3C**, que veremos en una lección posterior.

Recuerda: Aunque el modelo de eventos tradicional separa código Javascript de HTML, no permite asociar dos eventos del mismo tipo a un elemento... ¡utiliza el del W3C!

MODELO DE REGISTRO DE EVENTOS TRADICIONAL

Para estudiar cómo es la sintaxis del modelo de eventos tradicional, se plantea el siguiente escenario.

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

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

Como el objetivo de este modelo de eventos es separar el código HTML del código JavaScript, implementamos esta lógica en un fichero externo.

Por simplicidad didáctica, en este ejemplo se implementa entre etiquetas script.

MODELO DE REGISTRO DE EVENTOS TRADICIONAL

Este script nos permite asociar una funcionalidad mediante eventos a cada elemento de nuestro documento HTML.

```
<script>
    document.getElementById("tradicional").onclick = cambiar;
    //¡¡Sin paréntesis!! Ya que si los incluimos, la estaríamos invocando.
    function cambiar() {
        alert("Entramos en cambiar");
        document.getElementById("tradicional").innerHTML = "Modelo de
        registro de eventos tradicional";
    }
</script>
```

MODELO DE REGISTRO DE EVENTOS TRADICIONAL

Si queremos que la funcionalidad asociada al evento (`onclick` en este caso), se deshabilite una vez activa, podemos implementar lo siguiente

```
document.getElementById("tradicional").onclick = null;
```

De esta manera, des-asignamos el evento `onclick` al elemento con `id` `tradicional`.

Finalmente, el código completo resultaría:

MODELO DE REGISTRO DE EVENTOS TRADICIONAL

```
<script>
    document.getElementById("tradicional").onclick = cambiar;
    //¡¡Sin paréntesis!! Ya que si los incluimos, la estaríamos invocando.
    function cambiar() {
        alert("Entramos en cambiar");
        document.getElementById("tradicional").innerHTML = "Modelo de
        registro de eventos tradicional";
        //Des-asignamos el evento onclick al elemento
        document.getElementById("tradicional").onclick = null;
    }
</script>
```

MODELO DE REGISTRO DE EVENTOS TRADICIONAL

Inconveniente

INCONVENIENTE

Para que JavaScript sea capaz de acceder a los elementos implementados en el documento HTML, mediante las funciones del `DOM` que asignan manejadores a los elementos HTML, es requisito indispensable que este se haya terminado de cargar.

SOLUCIÓN

- Puedes hacer uso de los atributos `defer` o `async`, estudiados en la lección UT2 - FORMAS DE INSERTAR CÓDIGO JS

- Implementar un evento de finalización de carga del documento HTML.

EVENTO DE FINALIZACIÓN DE CARGA DEL DOCUMENTO HTML

Se parte del escenario anterior, donde implementábamos:

```
<h3 id="tradicional2">Pulsa aquí para ver lo que se ejecuta</h3>
```

Y en la funcionalidad JavaScript, incluimos el evento que nos indica que el documento HTML ha terminado de cargarse: `window.onload`.

Finalmente, el código completo resultaría:

EVENTO DE FINALIZACIÓN DE CARGA DEL DOCUMENTO HTML

```
<script>
    window.onload = function () {
        alert("La página ha cargado correctamente");
        document.getElementById("tradicional2").onclick = miMensaje;
        //Recuerda no incluir los paréntesis
    }

    function miMensaje() {
        document.getElementById("tradicional2").innerHTML = "Modelo de
        registro de eventos tradicional";
    }
</script>
```