5.A. Gestión de eventos y formularios en JavaScript

3.- Eventos.

3.1.- Modelo de registro de eventos en línea.

En el modelo de registro de eventos en línea (estandarizado por Netscape), el evento es añadido como un atributo más a la etiqueta HTML, como por ejemplo:

```
<a href="pagina.html" onClick="alert('Has pulsado en el enlace')">Pulsa aqui</a>
```

Cuando hacemos clic en el enlace, se llama al gestor de eventos onClick (al hacer clic) y se ejecuta el script; que contiene en este caso una alerta de JavaScript. También se podría realizar lo mismo pero llamando a una función:

```
<a href="pagina.html" onClick="alertar()">Pulsa aqui</a>


function alertar() {

     alert("Has pulsado en el enlace");

}
```

La mezcla de minúsculas y mayúsculas en los nombres de evento (onClick, onMouseOver) es sólo por tradición, ya que HTML es insensible a mayúsculas y minúsculas. En cambio en XHTML, sí que los atributos tendrán que ir obligatoriamente siempre en minúsculas: tienes que escribir onclick y onmouseover. Es muy importante que te fijes en ésto, ya que probablemente trabajarás con XHTML y deberás cumplir el estándar: "**todos los nombres de atributos irán siempre en minúscula**".

No uses el modelo de registro de eventos en línea

Este modelo no se recomienda, y aunque lo has visto en ejemplos que hemos utilizado hasta ahora, tiene el problema de que estamos mezclando la estructura de la página web con la programación de la misma, y lo que se intenta hoy en día es separar la programación en JavaScript, de la estructura HTML, por lo que este modelo no nos sirve.

En el ejemplo anterior, cuando haces clic en el enlace se mostrará la alerta y a continuación te conectará con la pagina.html. En ese momento desaparecerán de memoria los objetos que estaban en un principio, cuando se programó el evento. Ésto puede ser un problema, ya que si por ejemplo la función a la que llamamos, cuando se produce el evento, tiene que realizar varias tareas, éstas tendrían que ser hechas antes de que nos conecte con la nueva página.

Éste modo de funcionamiento ha sido un principio muy importante en la gestión de eventos. Si un evento genera la ejecución de un script y además también se genera la acción por defecto para ese objeto entonces:

1. El script se ejecutará primero.
2. La acción por defecto se ejecutará después.

Evitar la acción por defecto

A veces es interesante el bloquear o evitar que se ejecute la acción por defecto. Por ejemplo, en nuestro caso anterior podríamos evitar que nos conecte con la nueva pagina.html. Cuando programamos un gestor de eventos, ese gestor podrá devolver un valor booleano true o false. Eso tendremos que programarlo con la instrucción return truefalse. False quiere decir "no ejecutes la acción por defecto". Por lo tanto nuestro ejemplo quedará del siguiente modo:

```
<a href="pagina.html" onClick="alertar(); return false">Pulsa aqui</a>
```

De esa forma, cada vez que pulsemos en el enlace realizará la llamada a la función alertar() y cuando termine ejecutará la instrucción "return false", que le indicará al navegador que no ejecute la acción por defecto asignada a ese objeto (en este caso la acción por defecto de un hiperenlace es conectarnos con la página href de destino).

También sería posible que nos preguntara si queremos o no queremos ir a la pagina.html. Eso podríamos hacerlo sustituyendo true o false por una función que devuelva true o false según la respuesta que demos al "confirm":

```
<a href="pagina.html" onClick="return preguntar()">Pulsa aqui</a>


function preguntar() {

     return confirm("¿Desea realmente ir a esa dirección?");

}
```

Créditos de la imagen
Autoría: ldskater.

◀ Solución a la tarea para DWEC05

Ir a...

5.B. Eventos JavaScript en W3Schools ▶