5.A. Gestión de eventos y formularios en JavaScript

3.- Eventos.

3.3.- Modelo de registro avanzado de eventos según W3C.

El W3C en la especificación del DOM de nivel 2, pone especial atención en los problemas del modelo tradicional de registro de eventos. En este caso ofrece una manera sencilla de registrar los eventos que queramos, sobre un objeto determinado.

La clave para poder hacer todo eso está en el método `addEventListener()`.

Este método tiene tres **argumentos**: el **tipo de evento**, la **función a ejecutar** y un **valor** booleano (true o false), que se utiliza para indicar cuando se debe capturar el evento: en la fase de captura (true) o de burbujeo (false). En el apartado 3.5 veremos en detalle la diferencia entre estas dos fases.

```
elemento.addEventListener('evento', función, false|true)
```

Por ejemplo para registrar la función `alertar()` de los ejemplos anteriores, haríamos:

```
document.getElementById("mienlace").addEventListener('click',alertar,false);


function alertar() {

     alert("Te conectaremos con la página: "+this.href);

}
```

La ventaja de este método, es que podemos añadir tantos eventos como queramos. Por ejemplo:

```
document.getElementById("mienlace").addEventListener('click',alertar,false);

document.getElementById("mienlace").addEventListener('click',avisar,false);

document.getElementById("mienlace").addEventListener('click',chequear,false);
```

Por lo tanto, cuando hagamos clic en "mienlace" se disparará la llamada a las tres funciones. Por cierto, el W3C no indica el orden de disparo, por lo que no sabemos cual de las tres funciones se ejecutará primero. **Fíjate también, que el nombre de los eventos al usar `addEventListener` no lleva 'on' al comienzo**.

También se pueden usar funciones anónimas (sin nombre de función), con el modelo W3C:

```
element.addEventListener('click', function () {

     this.style.backgroundColor = '#cc0000';

}, false)
```

Uso de la palabra reservada `this`

La palabra reservada `this`, tiene exactamente la misma funcionalidad que hemos visto en el modelo tradicional.

¿Qué eventos han sido registrados?

Uno de los problemas de la implementación del modelo de registro del W3C, es que no podemos saber con antelación, los eventos que hemos registrado a un elemento.

En el modelo tradicional si hacemos: `alert(elemento.onclick)`, nos devuelve *undefined*, si no hay funciones registradas para ese evento, o bien el nombre de la función que hemos registrado para ese evento. Pero en este modelo no podemos hacer eso.

El **W3C** en el reciente **nivel 3 del DOM**, introdujo un método llamado `eventListenerList`, que almacena una lista de las funciones que han sido registradas a un elemento. Tienes que tener cuidado con este método, porque no está soportado por todos los navegadores.

Para eliminar un evento de un elemento, usaremos el método `removeEventListener()`:

```
elemento.removeEventListener('evento', función, false|true);
```

Para cancelar un evento, este modelo nos proporciona el método `preventDefault()`.

Créditos de la imagen

Autoría: Alpha TangoBravo/Adam Baker.

Licencia: CC BY 2.0.

◀ Solución a la tarea para DWEC05

Ir a...

5.B. Eventos JavaScript en W3Schools ▶