Unidad 9

Captura de eventos con Javascript

"Coming events cast their shadows before"

Thomas Campbell

Una parte muy importante del trabajo que se realiza mediante Javascript es lo que se denomina "programación visual orientada a eventos." Sin meternos en demasiada profundidad, diremos que los eventos son acciones que ocurren sobre los elementos de nuestro formulario, generalmente debido a la acción del usuario: cambiar el contenido de una caja de edición de texto en un formulario, pulsar o hacer doble click con el ratón en algún elemento, hacer pasar el ratón por encima de algo, redimensionar el navegador, etc. Los eventos vienen predeterminados según el elemento del formulario y no podemos añadir ninguno nuevo.

Los métodos son las funciones que se invocan al producirse los eventos. Existen ya algunos métodos por defecto que vienen predefinidos con los formularios pero nosotros siempre tendremos la posibilidad de crear otros nuevos o modificar los ya existentes.

Por último, las propiedades son, por regla general (aunque no siempre), valores asociados a los objetos de la página web: valores de los argumentos con que están construidos los elementos del formulario, los valores de los mismos introducidos o elegidos por el usuario y que nosotros podremos manipular desde nuestros métodos, estilos, contenido de las etiquetas, etc.

DOM, Modelo de objetos del documento

Puesto que Javascript tiene que interactuar con HTML y CSS que no son lenguajes de programación, es necesario facilitarle una forma de referenciar los diferentes elementos de la página web para poder leer y modificar sus atributos, valores, etc. Esto se consigue mediante DOM (Modelo de objetos del documento) que proporciona una forma de representar los diferentes elementos de una web y sus propiedades de forma que podamos interactuar con ellos desde Javascript. Puede parecer abstracto pero en realidad se trata de una idea muy sencilla. DOM no es más que un método que nos va a permitir interactuar con los elementos del lenguaje de marcas (HTML) y CSS mediante una nomenclatura especial (que a los programadores habituados a la orientación a objetos le resultará familiar).

Lo primero que tienes que hacer es recordar que los diferentes objetos de una página web realizada con html se organizan de forma jerárquica en forma de árbol. DOM los "representa" de esta misma forma introduciendo además la misma sintaxis que usamos en los vectores cuando tenemos varios elementos repetidos.

Supongamos una web muy sencilla cuyo body es así:

```
<body>
	<h1 id="titular">Titular</h1>
	<div id="caja1">Uno</div>
	<div id="caja2">Dos</div>
</body>
```

La forma de referenciar el segundo div sería esta:

```
window.document.getElementById("caja2")
```

En realidad puedo prescindir de "window":

```
document.getElementById("caja2")
```

Lo habitual es guardar la referencia al elemento HTML obtenida de esta forma en una variable para poder utilizarla más cómodamente:

```
caja = document.getElementById("caja2");
```

Tanto document como windows son objetos con una amplia colección de propiedades y atributos que nos permiten interactuar de forma fácil con características interesantes del navegador y el documento HTML.

Por ejemplo, el objeto window nos permite obtener las dimensiones en píxeles de la superficie útil del navegador. La siguiente función nos muestra estas dimensiones en una ventana de alerta:

```
function muestraDimensiones(){
        alert("Ancho: " + window.innerWidth + " píxeles.\nAlto: " +
        window.innerHeight + " píxeles.");
}
```

Tienes una buena referencia de las posibilidades que te ofrece el objeto window aquí:

http://www.w3schools.com/jsref/obj_window.asp

La principal utilidad del objeto document es que nos permite obtener una referencia a cualquier elemento HTML de la página para interactuar con él a través del valor de su atributo id que, como ya sabes, debería de ser único. Lo hacemos mediante la función getElementById() como ya hemos visto antes.

Para ver más propiedades del objeto document puedes consultar esta página:

http://www.w3schools.com/jsref/dom_obj_document.asp

Y si quieres ampliar conocimientos o ver más ejemplos sobre cualquier otra cosa de javascript, aquí tienes una completa referencia de todas las funciones disponibles:

http://www.w3schools.com/jsref/

Las propiedades de los elementos también pueden modificarse o leerse usando DOM. La propiedad innerHTML, por ejemplo, nos da el contenido de la etiqueta que estamos referenciando:

```
caja = document.getElementByID("caja2");
texto = caja.innerHTML;
caja.innerHTML = "OLA K ASE";
```

Cuando queremos modificar el mismo elemento desde el que capturamos un evento tenemos otra posibilidad: usar la palabra reservada **this** que hace referencia al objeto desde el que se provoca la llamada. Los que estéis acostumbrados a la POO os sonará. Al capturar el evento usamos esta sintaxis

```
<h1 onclick="modifica(this)">Titular</h1>
```

Y luego en la función modifica lo hacemos así:

```
function modifica(elemento){
	elemento.innerHTML = "Titular modificado"
}
```

La propiedad className nos permite modificar el valor del atributo class de una etiqueta y, por lo tanto, nos permite modificar sus estilos CSS:

```
objeto = document.getElementById("caja1");
objeto.className="estilo2";
```

Eventos

La forma de "capturar" los eventos es bien fácil: mediante nuevos argumentos que se introducen en las etiquetas HTML. El argumento en si es el evento a capturar (onclick, ondblclick, onload, etc.) y la mayoría de las veces es autoexplicativo. El valor de dicho argumento será el código javascript que se ejecutará cuando dicho evento se produzca y que por regla general suele ser una llamada a función. Por ejemplo, en el siguiente párrafo se captura el evento onmouseover para que cuando el ratón pase por encima del mismo se abra una ventana de alert saludando:

```
<p onmouseover="alert('¡Hola gente!')">Texto del párrafo</p>
```

Los eventos más usuales son los siguientes:

Evento	Descripción
onclick	Cuando se pulsa sobre un objeto una sóla vez con el botón izquierdo del ratón
ondblclick	Cuando se hace doble click con el botón izquierdo del ratón
onmousewheel	Cuando se mueve la rueda del ratón. No disponible en Firefox
onload	Cuando la página se carga (sólo se puede usar en la etiqueta body)
onmousedown	Cuando se presiona un botón cualquiera del ratón
onmouseup	Cuando se levanta el dedo de un botón cualquiera del ratón después de haberlo pulsado

onmousemove	Cuando el ratón pasa por encima del elemento
onmouseout	Cuando el ratón después de estar encima de un objeto sale de él
onmouseover	Cuando el ratón pasa a estar encima de un objeto
onresize	Cuando se cambia el tamaño de la ventana (sólo puede usarse en la etiqueta body)

Tienes una lista completa de los eventos con explicaciones aquí:

http://www.w3schools.com/jsref/dom_obj_event.asp

IMPORTANTE: *Fíjate bien que puesto que el valor del argumento debe de ir entrecomillado si en el código javascript necesitamos usar comillas (como en el caso anterior) debemos de usar las simples para que no haya problemas con el anidamiento de las mismas.*

Piensa que el argumento que define el evento es en realidad algo propio de HTML y por lo tanto no es sensible a mayúsculas y minúsculas (aunque si seguimos las recomendaciones de XHTML deberíamos de escribirlo completamente en minúsculas) mientras que el valor del mismo si es Javascript y es sensible a la forma en que lo escribamos. O sea, es indiferente si escribimos onmouseover, OnMouseOver u ONMOUSEOVER pero no si escribimos ALERT, Alert o alert.

Veamos otro trozo de HTML con más ejemplos sencillos:

```
<body onLoad="alert('La página se ha cargado completamente');"
onResize="alert('Haz cambiado el tamaño de la página');">
        <h1 onMouseOver="alert('Haz pasado el ratón sobre el
        titular');" >Titular</h1> <p onClick="alert('Haz hecho click en
        el primer párrafo');" >Primer párrafo</p>
        <p onDblClick="alert('Haz hecho doble click en el segundo
        párrafo');" >Segundo párrafo</p>
</body>
```