Eventos

Los eventos son manejadores que nos proporciona el navegador para que cuando detecte que se produzca una acción (***evento***), se ejecute un código asociado a esa acción.

A continuación mostramos un listado de los principales eventos existentes en Javascript:

- ✔ **onfocus**: al obtener un foco.
- ✔ **onblur**: al salir del foco de un elemento.
- ✔ **onchange**: al hacer un cambio en un elemento.
- ✔ **onclick**: al hacer un click en el elemento.
- ✔ **ondblclick**: al hacer doble click en un elemento.
- ✔ **onkeydown**: al pulsar una tecla (sin soltarla).
- ✔ **onkeyup**: al soltar una tecla pulsada.
- ✔ **onkeypress**: al pulsar una tecla.
- ✔ **onload**: al cargarse una página.
- ✔ **onunload**: al descargarse una página (salir de ella).
- ✔ **onmousedown**: al hacer clic de ratón (sin soltarlo).
- ✔ **onmouseup**: al soltar el botón del ratón previamente pulsado.
- ✔ **onmouseover**: al entrar encima de un elemento con el ratón.
- ✔ **onmouseout**: al salir de encima de un elemento con el ratón.
- ✔ **onsubmit**: al enviar los datos de un formulario.
- ✔ **onreset**: al resetear los datos de un formulario.
- ✔ **onselect**: al seleccionar un texto.
- ✔ **onresize**: al modificar el tamaño de la página del navegador.

Importante: El total de eventos disponibles está descrito en esta página http://www.w3schools.com/jsref/dom_obj_event.as

Asignando manejadores de eventos desde HTML

La forma más sencilla (aunque menos práctica para tener un código limpio y ordenado) de indicar que hay un evento asociado a un elemento HTML es indicándolo *en el propio código*.

Ejemplo 1:

```
<input type="button" value="Boton Hola mundo" onclick="alert('Hola mundo');alert('Adios mundo');" />
```

También en lugar de ejecutar una serie de instrucciones, es posible llamar a una función predefinida.

Ejemplo 2:

```
<input type="button" value="Botón miFuncion" onclick="miFuncion('cadenaParam1');" />
```

Asignación de eventos desde código a objetos HTML

Es posible asignar y/o modificar mediante código el manejador de un evento predefinido en un documento HTML de una forma similar a esta. Supongamos que tenemos un objeto dentro del DOM con **id="miObjeto"** que posee el evento "**onclick**" sin asignar y la función "**mostrarMensaje**".
Podemos referenciar al elemento HTML con "**document.getElementById**" y asignar la función como manejador del evento como vemos en este código

```
function mostrarMensaje(){
        alert("Hola");
}
document.getElementById("miObjeto").onclick=mostrarMensaje;
```

Realizar la asignación de eventos de esta forma nos permite facilitar la separación de código e interfaz, ya que se traslada la asignación del manejador del documento HTML al código JS.

Una operación típica dentro del manejo de eventos, es la de comenzar a ejecutar código cuando se haya acabado de cargar una página HTML. Esto ocurre porque si escribimos código Javascript que se ejecute al iniciar la página, pero no nos aseguramos que la página está completamente cargada, es posible que nos de errores (a menudo aleatorios) del estilo de "no encuentro el elemento X del DOM", ya que el código Javascript se ha ejecutado antes de la carga de dicho elemento.

Ejemplo de código que se ejecuta una vez se ha cargado el DOM:

```
// Función que se ejecuta cuando se ha cargado todo el DOM
function inicio(){
 //Código a ejecutar
}
// Asignamos la función inicio al manejador window.onload
// Esto garantiza que el código de "inicio" se ejecute con todo el DOM cargado
window.onload=inicio;
```

Obteniendo información del objeto event

Cuando se crea una función como manejador, al producirse el evento el navegador automáticamente manda como parámetro un objeto de tipo event.

```
function mostrarMensaje(evento){
        alert(evento.type);
}
document.getElementById("miObjeto").onclick=mostrarMensaje;
```

Este objeto posee cierta información útil del evento que se ha producido. Sus atributos más destacados son:

- **type**: dice el tipo de evento que es ("click", "mouseover", etc...). Devuelve el nombre del evento tal cual, sin el "on". Es útil para hacer una función que maneje varios eventos.
- **keyCode**: en eventos de teclado, almacena el código de tecla de la tecla afectada por el evento.
- **clientX** / **clientY**: en eventos del ratón, devuelve las coordenadas X e Y donde se encontraba el ratón, tomando como referencia al navegador.
- **screenX** / **screenY**: en eventos del ratón, devuelve las coordenadas X e Y donde se encontraba el ratón, tomando como referencia la pantalla del ordenador.

Ejemplo:

```
function mostrarMensaje(evento){
        if(evento.type==="keyup"){
                alert(evento.keyCode);
        }
        else if(evento.type==="click"){
                alert(evento.clientX+" "+evento.clientY);
        }
}
document.getElementById("miObjeto").onclick=mostrarMensaje;
document.onkeyup=mostrarMensaje;
```

Drag and drop (arrastrar y soltar)

Además de los eventos típicos tratados, existe un proceso (en el que entran en juego varios eventos) que es útil para el desarrollo de aplicaciones Web, el "**Drag and Drop**" (Arrastrar y soltar).

En W3Schools podéis obtener teoría y ejemplos de esta técnica
https://www.w3schools.com/html/html5_draganddrop.asp

Aquí un ejemplo completo:

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
 ev.preventDefault();
}
function drag(ev) {
 ev.dataTransfer.setData("text", ev.target.id);
}
function drop(ev) {
 ev.preventDefault();
```

```
        var data = ev.dataTransfer.getData("text");
        ev.target.appendChild(document.getElementById(data));
       }
       </script>
       </head>
<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<img
id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height=
"69">
</body>
</html>
```

4.Bibliografía

[1] Javascript Developer https://developer.mozilla.org/es/docs/Web/JavaScript
[2] Javascript ES6 W3C https://www.w3schools.com/js/js_es6.asp
[3] Referencia Javascript http://www.w3schools.com/jsref/dom_obj_event.asp