CAPTURA DE EVENTOS

MÉTODOS CLÁSICOS Y OBSOLETOS

Hace años, en las primeras versiones de JavaScript, la captura de un evento se realizaba desdeel propio código HTML. Todavía se puede hacer porque se ha mantenido esta forma de capturapor cuestiones de compatibilidad, pero no es nada aconsejable, ya que complicaenormemente elmantenimiento del código al estar mezclado HTMl y JavaScript de forma confusa.

Este primer método utiliza atributos en las etiquetas de los elementos de la página quecomienzancon la palabra on seguida del nombre del evento. Por ejemplo: **onclick**. Como valordel atributo seindica el nombre de la función que contiene el código a ejecutar. Ejemplo:

*** Soy 'clickable' ***
*** Yo no ***

Una mejora del método anterior, que permite que el código JavaScript se independice deHTML, loque mejora su mantenimiento. Se trata de usar propiedades que tienen el mismo nombre que losatributos HTML: la palabra on seguida del nombre del evento (onclick, onmouseover, etc). Ha sido elmétodo más utilizado durante muchos años, pero actualmente no serecomienda su uso. Ejemplo:

```
<p id="parrafo1">Soy 'clickable'< /p>
<p id="parrafo2">Yo no< /p>
<script>
let parrafo1=document.getElementById("parrafo1");
parrafo1.onclick=()=>{alert('Me has hecho click')};
</script>
```

Se ha asignado como función callback una función flecha que muestra el mismo mensaje detipo alertque en el ejemplo anterior.

El método que se aconseja actualmente es utilizar el **método addEventListener** que estádisponibleen todos los elementos. A este método hay que indicarle el nombre del evento y lafunción callback aejecutar. Es el método que se debe utilizar según la norma actual. Todos losnavegadores actuales loreconocen (las versiones de Internet Explorer anteriores a la 9, no lo hacen). Este método tiene unaventaja técnica sobre los dos métodos anteriores: ***se puede asignar más de unafunción al mismoevento***. Ejemplo:

<script>
let parrafo1=document.getElementById("parrafo1");
parrafo1.addEventListener("click",()=>{alert('Me has hecho click')});
</script>

PROPAGACIÓN DE EVENTOS

Una cuestión fundamental en la captura de eventos es cómo se propagan los eventos sobreloscontenedores de los elementos. Es decir, supongamos que tenemos una capa de tipo div, enella unpárrafo y dentro del párrafo un botón. Hacemos clic sobre el botón. ¿El párrafo podríacapturar elclick? ¿Y la capa?

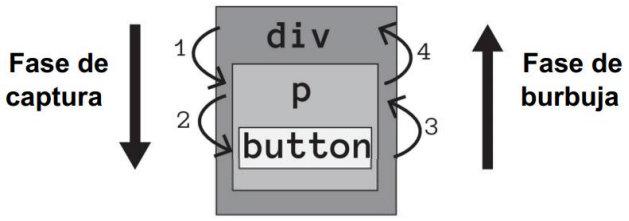
Supongamos el siguiente ejemplo:

```
<body>
<div>
<p>
<button> ¡Hazme elic!</button>
</p>
</div>
<script>
let capa=document.querySelector("div");
let p=document.querySelector("p");
let boton=document.querySelector("button");
capa.addEventListener("click",()=>{console.log("click en capa")});
p.addEventListener("click",()=>{console.log("click en p")});
  boton.addEventListener("click",()=>{console.log("click en boton")});
</script>
</body>
```

Hemos capturado los eventos detipo click en los tres elementos (**div, p y button**). Si ejecutando elcódigo, hacemos clic en el botón,por consola veremos este texto:

click en boton
click en p
click en capa

El evento se propaga desde el botón (elemento más interior) hasta el elemento de tipo div(elementomás exterior, elemento más cercano a la raíz del DOM).El proceso de captura de eventos se describeen esta imagen:



Por defecto, la función asociada al evento se lanza en la fase de burbuja, por eso en el códigoanterior podemos observar primero el mensaje del botón, luego el del párrafo y luego el de lacapa.Podemos modificar este comportamiento, si indicamos que el lanzamiento sea en la fase decaptura.Se consigue este efecto (no disponible en versiones de Internet Explorer anteriores a la9) gracias aque, en realidad, el método addEventListener tiene un tercer parámetro relacionadocon el tipo decaptura. Se trata de un valor booleano que tiene estas posibilidades:

• **true**. El código de captura se lanza en la fase de captura.
• **false**. Es el valor por defecto. El código se lanza en la fase de burbuja.

Si en los ***addEventListener*** anteriores les pasamos un tercer parámetro con valor true la salida sería:

click en capa
click en p
click en boton

ANULAR EVENTOS

Por defecto, el eventose captura indefinidamente, pero existe un método contrario aaddEventListener que tiene losmismos parámetros. Se trata de ***removeEventListener*** que eliminará la funcionalidad de addEventListener. Ejemplo:

```
<p>Hazme clic!< /P>
<script>
function mensaje(){
alert("!Me has hecho clic!")
p.removeEventListener("click",mensaje);
}
let p=document.querySelector("p");
p.addEventListener("click",mensaje);
</script>
```

Dentro de la propia función mensaje, tras escribir el mensaje en sí, se invoca al métodoremoveEventListener el cual provoca que no se vuelva a invocar a esta función tras otro clic.Hay quetener en cuenta que solo podemos retirar funciones de captura que tengan nombre,no podemosanular funciones anónimas asociadas a eventos.

CAPTURA DE EVENTOS EN ELEMENTOSDINÁMICOS

Podemos asignar de forma dinámica eventos a los elementos del DOM. En el siguiente ejemplo se seleccionan todos los párrafos y a cada uno de ellos se le asocia un evento que al pulsar sobre cada párrafo un cuadro de alerta con el texto del párrafo.

```
<p>Uno</p>
<p>Dos</p>
<p>Tres</p>
<p>Cuatro</p>
<script>
let parrafos=document.querySelectorAll("p");
 for(let parrafo of parrafos){
  párrafo.addEventListener("click",()=>{alert(parrafo.textContent)});
}
</script>
```

OBJETO DE EVENTO

Cuando se produce un evento, el navegador crea automáticamente un objeto cuyas propiedadespueden ser muy útiles a los desarrolladores. La información fundamental que graban son lascoordenadas del cursor del ratón, si hay teclas pulsadas, el elemento que ha producido el evento,etc…cuando se produce un evento, se invoca automáticamente el código deuna función. Pues bien,esa función puede tener un parámetro que será una referencia al objetode evento. Gracias a eseparámetro podremos leer la información del evento.Una de las propiedades que posee este objetose llama **target** y es una referencia al elementoque causó el evento.

Los objetos de evento poseen dos propiedades para obtener las coordenadas del ratón en elmomento del evento: **clientX** y **clientY**. La primera obtiene la posición horizontal en píxeles y la

segunda la vertical, utilizando como referencia la esquina superior izquierda dela ventana delnavegador, que será el punto (0,0) de coordenadas.

Hay dos propiedades similares: **screenX** y **screenY**. La diferencia es que el origen de coordenadas noes la esquina del navegador, sino de la pantalla. Coinciden muchas veces, pero si la ventanano está maximizada, no coincidirán.

Otras coordenadas que se almacenan en el objeto de evento son **pageX** y **pageY**. FuncionancomoclientX y clientY, pero no solo toman las de la ventana, tienen en cuenta el desplazamientorealizadoen el elemento. Es decir, si hemos avanzado dos pantallas de 500 píxeles de alto usandolas barras dedesplazamiento y el cursor está a mitad de pantalla, clientY nos diría 250, mientras que pageY nosdiría 1250. Depende de lo que deseemos nos vendrán mejor unas u otrascoordenadas.

EVENTOS DE RATON

En los eventos del ratón hay propiedades comunes con las teclas. Por ejemplo, también tenemos laspropiedades **AltKey, CtrlKey, ShiftKey y MetaKey**, para saber si la pulsación de alguna deestas teclasacompaña al evento de ratón.

Hay otras propiedades de coordenadas muy interesantes (además de las ya vistas). Así **movementX** y**movementY** nos retornan la diferencia en píxeles de las coordenadas X e Y respecto alúltimoovimiento del ratón.

Además, la propiedad **button** devuelve el botón de ratón pulsado en el momento del evento.Losvalores posibles para lo propiedad button son:

VALOR	SIGNIFICADO
0	Botón principal del ratón.
1	Botón central del ratón (en muchos casos, el botón de la rueda).
2	Botón secundario del ratón (en el caso de personas diestras, es el botón derecho).
3	Cuarto botón. En muchos ratones es el de retroceder página.
4	Quinto botón. En muchos ratones es el de avanzar página.

EVENTO	EXPLICACIÓN
click	El usuario hace clic sobre el botón principal del dispositivo apuntador. Esta acción exactamente consiste en bajar y subir rápidamente el botón principal del dispositivo apuntador (o dar un golpe con el dedo en un dispositivo táctil)
dblclick	El usuario golpea dos veces rápidas sobre el botón principal del dispositivo apuntador (o con el dedo en un dispositivo táctil)
mousedown	El usuario pulsa un botón del dispositivo apuntador. Se produce este evento justo cuando el usuario pulsa el botón, antes de que le suelte.
mouseup	Se produce cuando el usuario suelta el botón del dispositivo apuntador que había pulsado previamente.
mousenter	Se produce cuando el usuario mueve el cursor del dispositivo apuntador dentro del elemento que captura el evento. Solo se produce si el cursor estaba fuera del elemento y el usuario le acaba de mover dentro.
mouseleave	Se produce cuando el usuario mueve el cursor del dispositivo apuntador fuera del elemento que captura el evento. Solo se produce si el cursor estaba dentro del elemento y el usuario le acaba de mover fuera.
mousemove	Se produce cada vez que el usuario mueve el cursor dentro del elemento (estando previamente dentro).
mouseover	Se produce cuando el cursor apuntador entra dentro del elemento o de cualquiera de los hijos del elemento.
mouseout	Se produce cuando el cursor apuntador abandona el elemento o cualquiera de los hijos del elemento.
contextmenu	Se produce cuando el usuario acaba de pedir el menú de contexto. Normalmente ese menú aparece al pulsar el botón secundario del dispositivo apuntador. En dispositivos táctiles suele ocurrir cuando el usuario deja el dedo apretado sobre el elemento 2 o 3 segundos al menos.

EVENTOS DE TECLADO

En eventos de teclado, el objeto de evento posee información sobre la tecla pulsada. Sonimportantes estos datos en aplicaciones como la programación de juegos o en el control de laentrada por el teclado.

La propiedad más importante es key que obtiene el texto que indica la tecla pulsada. Es el máscómodo de programar porque es fácil de entender. Así sus valores fundamentales devueltos son:

• Si es una tecla de carácter, nos retorna el carácter. Por ejemplo: "a", "g", "k","ñ",etc. También nosindica si hay mayúsculas porque la letra la retornaría mayúscula: "A", "G","K", "Ñ", etc.

• Con las teclas numéricas actúa igual, nos retorna el número. Pero con shift pulsada a lavez, nosdevuelve el segundo símbolo de la letra (<&, $, %, etc.). Con AltGr nos retorna eltercero (|, <©, #, etc.).

• Actúa igual con cualquier tecla que escriba caracteres, simplemente indica qué carácternormalmente sale escrito.

• En el caso de pulsar, sin más, teclas de control, nos devolverá su nombre: "Control", "Alt","Shift","AltGraph", "F5", "F6", "KeyUp", "KeyDown", "Home", "End", etc.

A veces es necesario saber si se pulsó a la vez que la tecla, las teclas de control especial: Ctrl,Alt o Shift. Tenemos tres propiedades relacionadas con estas teclas que devolverán true si la tecla en cuestión se pulsó. Son: altKey, ctrlKey, shiftKey y metaKey (tecla de los ordenadores Mac).

EVENTO	EXPLICACIÓN
keypress	Se produce cuando un usuario pulsa y suelta una tecla.
keydown	Se produce justo cuando el usuario ha pulsado la tecla (antes de que la suelte).
keyup	Se produce cuando el usuario suelta la tecla.

Otra propiedad interesante es location que permite saber la localización de la tecla en elteclado. Elcaso habitual es tener que distinguir una tecla Shift de la otra. La propiedad locationlo consigue, estapropiedad puede devolver los siguientes valores:

Devuelve un número entero que indica la posición del teclado en la que se encuentra la tecla.

Valores posibles que devuelve:

VALOR	CONSTANTE RELACIONADA	SIGNIFICADO
0	DOM_KEY_LOCAT1ON_STANDARD	Teclado normal
1	**DOM_KEY_LOCATION_LEFT**	Zona izquierda (para distinguir las dos teclas control por ejemplo)
2	**DOM_KEY_LOCATION_R1GHT**	Zona derecha (para distinguir las dos teclas control por ejemplo)
3	**DOM_KEY_LOCATION_NUMPAD**	Teclado numérico

EVENTOS DE MOVIMIENTO EN LA VENTANA

EVENTO	EXPLICACIÓN
scroll	Ocurre cuando se ha desplazado la ventana a través de las barras de desplazamiento o usando el dispositivo táctil.
resize	Evento de window que se produce cuando se cambia el tamaño de la ventana.

EVENTOS SOBRE CARGA Y DESCARGA DEELEMENTOS

Concretamente, el evento load del objeto window es muy importante. Hay que tener encuenta quedesde JavaScript manejamos elementos del DOM continuamente. Además, JavaScriptes un lenguaje asíncrono, como se ha comentado en este libro varias veces, y eso puede implicarque el códigoJavaScript intente manipular un componente que aún no se ha cargado.

Por ello es muy habitual que todo el código JavaScript para manipular elementos del DOM, secoloque dentro de la función callback asociada al evento load del objeto window.

EVENTO	EXPLICACIÓN
load	Se concluyó la carga del elemento. Es uno de los elementos más importantes a capturar para asegurar que el código siguiente funciona con la seguridad de que está cargado lo que necesitamos. Cuando se aplica al elemento **window,** se produce cuando todos los elementos del documento se han cargado.
DOMContentLoaded	Similar al evento ***load.*** Se produce cuando el documento HTML ha sido cargado. A diferencia de ***load,*** se dispara sin esperar a que se terminen de cargar las hojas de estilos, imágenes y elementos en segundo plano. En general, para JavaScript, es más conveniente este evento.
abort	Se produce cuando se anula la carga de un elemento.
error	Sucede si hubo un error en la carga.
progress	Se produce si la carga está en proceso.
readystatechange	Ocurre cuando se ha modificado el estado del atributo readystate, lo cual ocurre cuando se ha modificado el estado de carga y descarga.

EVENTOS SOBRE EL HISTORIAL

EVENTO	EXPLICACIÓN
popstate	Se produce si se cambia el historial.

EVENTOS RELACIONADOS CON LA REPRODUCCIÓN DEMEDIOS

EVENTO	EXPLICACIÓN
waiting	Sucede cuando el vídeo se ha detenido por falta de datos.
playing	El medio está listo para su reproducción después de que se detuviera por falta de datos u otras causas.
canplay	Ocurre cuando se detecta que un vídeo (u otro elemento multimedia) ya se puede reproducir (aunque no se haya cargado del todo).
canplaythough	Se produce si se estima que el vídeo ha cargado suficientes datos para poderse reproducir sin tener que esperar la llegada de más datos.
pause	El medio de reproducción se ha pausado.
play	Ocurre cuando el medio de reproducción se ha empezado a reproducir tras una pausa.
ended	Se produce si la reproducción del vídeo o audio se ha detenido, sea por haber llegado al final o porque no hay más datos disponibles.
loadeddata	Se produce si se ha cargado el frame actual (normalmente el primero).
suspend	Se lanza si se ha suspendido la carga del medio.
emptied	Sucede si se ha vaciado el medio por un nuevo intento de carga por parte del usuario o por otras razones.
stalled	Sucede ante un fallo en la carga, pero sin que se detenga la misma.

EVENTO	EXPLICACIÓN
seeking	Ocurre cuando se ha iniciado una labor de búsqueda en el medio.
seeked	Ocurre cuando se ha finalizado la labor de búsqueda.
loadedmetadata	Se han cargado los metadatos del medio
durationchange	Sucede si se modifica el atributo duration del medio.
timeupdate	Ocurre si se ha modificado el atributo currentTime del medio.
ratechange	Se produce cuando el ratio del vídeo se ha modificado.
volumechange	Se lanza si el volumen se ha modificado.

EVENTOS DE ARRASTRE

Están relacionados con la interfaz de arrastre que es parte de HTML 5. Para que un elementopueda ser arrastrable debe tener el atributo draggable colocado con valor true:

*** ¡Soy arrastrable! ***

En este tipo de operaciones hay dos capas protagonistas: una capa arrastrable (la que realmente searrastra) y un posible destino del arrastre.

La capa que se arrastra puede generar estos eventos:

EVENTO	EXPLICACIÓN
dragstart	Se produce cuando el usuario empieza a arrastrar el elemento.
drag	Ocurre, una vez iniciado el arrastre, cada vez que se sigue arrastrando el elemento (cada vez que lo movemos).
dragstop	Se produce cuando el arrastre finaliza.

Eventos que se produce en el elemento destino del arrastre.

EVENTO	EXPLICACIÓN
dragenter	Se produce cuando el elemento que se está arrastrando, entra en el elemento destino.
dragover	Ocurre cada vez que se continúa, tras haber entrado, arrastrando el elemento origen sobre el destino.
dragleave	Ocurre cuando el elemento que se arrastra, sale del destino.
drop	Ocurre cuando el elemento origen se suelta dentro del destino. Para que este evento se pueda capturar hay que eliminar el comportamiento por defecto del evento **dragover.**

EVENTOS SOBRE ANIMACIONES Y TRANSICIONES

Son eventos que podemos capturar cuando hemos programado animaciones sobre uno o máselementos desde CSS o por transiciones. Tenemos estos eventos posibles a capturar:

EVENTO	EXPLICACIÓN
animationstart	Se produce cuando se inicia una animación sobre el elemento.
animationinteraction	Se produce justo cuando se repite la animación.
animationend	Se produce cuando finaliza la animación.
transitionrun	Se lanza cuando ya se ha preparado para empezar la transición.
transitionstart	Ocurre cuando se inicia una transición.
transitionend	Ocurre al finalizar la transición.

EVENTOS DEL PORTAPAPELES

Permiten capturar los eventos de cortar, copiar y pegar. Todos ellos se relacionan con el objetoclipboard que es el que permite gestionar el portapapeles del sistema.

EVENTO	EXPLICACIÓN
cut	Se produce cuando el usuario intenta cortar contenido del elemento.
copy	Se produce cuando el usuario intenta copiar contenido del elemento.
paste	Se produce cuando el usuario intenta pegar contenido.

EVENTOS ESPECIALES

EVENTO	EXPLICACIÓN
offline	Solo funciona para el objeto window ***y se*** produce si el navegador se desconecta de la red.
online	Solo funciona para window y se produce si el navegador vuelve a conectarse a la red después de haber estado desconectado.
fullscreenchange	Ocurre cuando un elemento pasa a modo de pantalla completa.
fullscreenerror	Sucede si hay un error al pasar un elemento a modo de pantalla completa.
message	Evento que se asocia a numerosos elementos de envío de mensajes como los que se producen a través de las APIs **WebSockets, WebWorkers** y otras.

EVENTOS DE FORMULARIO

El objeto de formulario (form) posee estas propiedades y métodos:

PROPIEDAD O MÉTODO	USO
elements	Devuelve una colección que contiene todos los controles del formulario.
length	Obtiene el número de controles del formulario.
action	Devuelve el contenido de la propiedad action, que marca la URL destino de los datos del formulario. Permite su modificación.

PROPIEDAD O MÉTODO	USO
method	Retorna el contenido de la propiedad method del formulario, que marca la forma de envío de los datos (get o post). Permite su modificación.
enctype	Obtiene o cambia la forma de codificar los datos del formulario.
acceptCharset	Obtiene o modifica el conjunto de caracteres del formulario.
submit()	Envía los datos del formulario a su destino.
resetO	Deja los valores de los controles del formulario a su estado por defecto.

Esos métodos van a permitir automatizar acciones de forma muy efectiva en los formularios.

EVENTO SUBMIT

Ejemplo:

```
<form action="#" id="formulario">
    <input type="text" name="usuario" id="usuario">
    <input type="submit" value="Enviar" id="enviar">
</form>
<div id="mensaje"></div>
```

```
let enviar=document.querySelector("#formulario");

enviar.addEventListener("submit",function (ev) {
    // la funcion va dentro del addEvenListener

    let texto=document.querySelector("#usuario").value;
    if (texto.length<=6) {
        ev.preventDefault();
        document.querySelector("#usuario").style.backgroundColor="red";
        document.querySelector("#mensaje").innerHTML="Minimio 6 caracteres, tron";
    }
    else {
        alert("Formulario enviado");
    }


});
```

Este evento se produce justo antes del envío y se asocia al propio formulario (también sepuedecapturar en el botón de tipo ***submit***del formulario). Tras su captura podemos validar losdatos, peroteniendo la precaución de que si no cancelamos la acción por defecto de este evento(mediante el**preventDefault** del objeto de evento), la acción se llevará a cabo. En el ejemplo anterior si se cumple que la longitud es menor o igual que 6 caracteres la entrada de usuario no es válida y por tanto se cancela el efecto del submit para que no envíe el formulario mediante preventDefault().

EVENTO RESET

Este evento se produce cuando se ha ordenado, a través de un botón de tipo reset por ejemplo,queel formulario muestre los valores iniciales y borre lo que el usuario hubiera escrito. La captura delevento nos permitirá matizar esta operación o añadir acciones a la misma.

EVENTOS DE ENFOQUE

Son eventos asociados a los controles de formulario en los que el usuario puede escribir oelegiropciones. Un control obtiene el foco cuando al usar el teclado manejamos dicho control.El enfoquese consigue gracias a la interacción con el dispositivo apuntador (un clic de ratón porejemplo) o alcambio de control de formulario gracias a la tecla del tabulador.Un control de formulario lanza elevento focus cuando obtiene el foco y lanza blur cuandopierde el foco.

EVENTO DE CAMBIO DE VALOR

El evento change es, seguramente, el evento más importante de los formularios. Se producecuandomodificamos el valor de cualquier control de formulario. Ejemplos de ello son:

Cambiar el estado de un botón de radio o de tipo check.
Modificar el valor de un cuadro de texto o de contraseña.
Elegir otro valor de una lista de opciones.
Arrastrar un deslizador para modificar su valor.
Realmente, el evento se produce cuando se ha modificado el valor y, además, se ha perdido elfocosobre ese control. Es decir, durante el cambio no se lanza el evento.

MÉTODOS DE LOS CONTROLES

MÉTODO	USO	CONTROLES
focus()	Fuerza a que el control obtenga el foco.	Prácticamente todos.
blur()	Provoca la pérdida de foco en el control.	Prácticamente todos.
selectf)	Selecciona todo el texto en el control y también deja el foco en él.	Controles de entrada de texto.
setSelectionRange(inicio,fin)	Selecciona el texto que va desde la posición inicio hasta la posición fin (sin incluir esta última).	Controles de entrada de texto.