UT6_2.- JAVASCRIPT. AVANZADO - EVENTOS

RESULTADOS DE APRENDIZAJE ASOCIADOS
1. Reconoce las características de lenguajes de marcas analizando e interpretando fragmentos de código. 2. Utiliza lenguajes de marcas para la transmisión de información a través de la Web analizando la estructura de los documentos e identificando sus elementos
CRITERIOS DE EVALUACIÓN
- De RA1 – desde CEA hasta CEK - De RA2 – desde CEA hasta CEH

UT6_2.- JAVASCRIPT. AVANZADO - EVENTOS

Índice de contenido

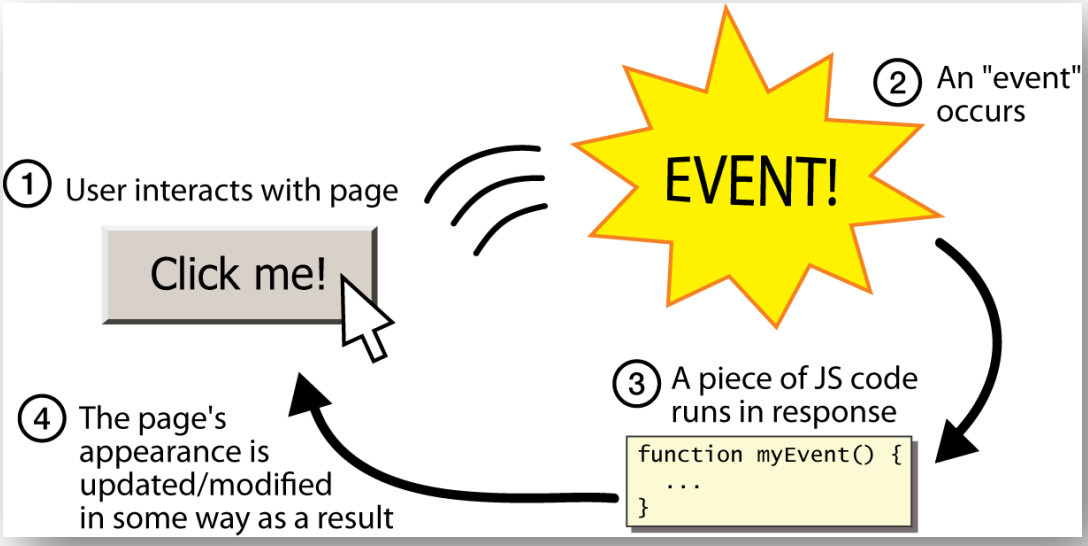
1.- Introducción

En casi todas las páginas web que incorporan JavaScript, suele haber eventos programados que disparan la ejecución de dichos scripts.

La razón es muy simple, JavaScript fue diseñado para añadir interactividad a las páginas: el usuario realiza algo y la página reacciona.

Por lo tanto, JavaScript necesita de alguna forma **detectar las acciones del usuario**, para saber cuándo reaccionar. También necesita saber las funciones a ejecutar cuando se producen esas acciones.

Cuando el usuario hace algo, se produce un evento.



También **habrá algunos eventos que no están relacionados directamente con acciones de usuario**: por ejemplo, el evento de carga (load) de un documento, que se disparará automáticamente cuando un documento ha sido cargado en el navegador.

2.- Tipos de eventos

Al principio, los eventos se solían asociar en línea junto con la etiqueta HTML, con un atributo que comenzaba por "on" seguido del tipo del evento, por ejemplo: **onClick=..., onSubmit=..., onChange=..., pero hoy en día esa forma de uso está quedando obsoleta**, debido a los nuevos modos de registro de eventos propuestos por el W3C, y que soportan ya la mayoría de los navegadores modernos.

Actualmente utilizamos un **listener sobre el evento**, que nos permite saber cuándo se produce el evento y de esta manera poder manejarlo mediante una función de callback:

elemento.addEventListener("evento", función callback);

Ejemplo:

```
const element = document.getElementById("miBoton");
element.addEventListener("click", miFuncion);

function miFuncion() {
    document.getElementById("demo").innerHTML = "Hola mundo";
}
```

2.1.- Formas de manejar eventos

Forma	Ejemplo	Más información

Mediante **atributos HTML**	`<button onClick="...">`
	
	`</button>`
		https://lenguajejs.com/javascript/eventos/eventos-html/

Mediante **propiedades Javascript**	`.onclick = function( ) {`
	`...`
	`}`
		https://lenguajejs.com/javascript/eventos/eventos-javascript/

Mediante **addEventListener()**	**.addEventListener("click", ...)**	https://lenguajejs.com/javascript/eventos/addeventlistener/

Muy importante entender muy bien la 3ª forma (añadir evento mediante addEventListener())

2.2.- Pasar parámetros dentro de la función de addEventListener

Para pasar parámetros a una función que se define dentro de un evento "addEventListener", puedes utilizar una función de flecha que envuelva la llamada a tu función, y luego pasar los parámetros deseados como argumentos a esa función de flecha.

En lugar de pasar la función "asignarOperador" directamente como argumento al evento "click", puedes utilizar una función de flecha para envolver la llamada a la función, de la siguiente manera:

```
sumar.addEventListener("click", () => asignarOperador(sumar));
restar.addEventListener("click", () => asignarOperador(restar));
mult.addEventListener("click", () => asignarOperador(mult));
div.addEventListener("click", () => asignarOperador(div));
```

Si no se utiliza una función de flecha para envolver la llamada a la función, y se pasa la función directamente como argumento al evento "click", **entonces la función se ejecutará inmediatamente**, en lugar de esperar a que se haga clic en el botón.

3.- Eventos comunes en el W3C

Hay una colección enorme de eventos que pueden ser generados para la mayor parte de elementos HTML:

- ✓ Eventos de ratón.

- ✓ Eventos de teclado.
- ✓ Eventos objetos frame.
- ✓ Eventos de formulario.
- ✓ Eventos de interfaz de usuario.
- ✓ Eventos de mutación (notifican de cualquier cambio en la estructura de un documento).

Teclado	**keydown**	Este evento se dispara justo antes del eventokeypress al presionar una tecla.	Sí.	Sí.
	keypress	Este evento se dispara después de keydown al presionar una tecla.	Sí.	Sí.
	keyup	Al soltar una tecla.	Sí.	Sí.

Categoría	Tipo de Evento	Descripción	Burbujea	Se puede cancelar
Ratón	**click**	Al hacer click sobre un elemento. Un click se define como mousedown y mouseup sobre la misma localización en pantalla.	Sí.	Sí.
	dblclick	Al hacer doble click sobre un elemento.	Sí.	Sí.
	mousedown	Al mantener presionado el botón del ratón sobre un elemento.	Sí.	Sí.
	mousedownmouseup	Al soltar el botón del ratón que estaba sobre un elemento.	Sí.	Sí.
	mouseover	Al pasar el ratón justo sobre un elemento.	Sí.	No.
	mousemove	Cuando el ratón se mueve mientras está sobre un elemento.	Sí.	Sí.
	mouseout	Cuando el ratón sale fuera de un elemento.	Sí.	Sí.

Frame HTML	**load**	Se dispara cuando se ha terminado de cargar todo el contenido de un documento, incluyendo ventanas, frames, objetos e imágenes.	No.	No.
	unload	Al salir de un documento y modificar el contenido de una ventana.	No.	No.
	abort	Cuando se detiene la carga de un objeto/imagen antes de que esté completamente cargado.	Sí.	No.
	error	Cuando se detiene la carga de un objeto/imagen antes de que esté completamente cargado.	Sí.	No.
	resize	Cuando se redimensiona un documento.	Sí.	No.
	scroll	Cuando nos desplazamos por el documento con scroll.	Sí.	No.

4.- El objeto EVENT

Generalmente, los manejadores de eventos necesitan información adicional del evento que se ha producido. Si se ha producido el evento click, que nos haga falta saber la posición en la que estaba el ratón en el momento de realizar el click, qué botón de ratón se ha pulsado, ….

También puede que necesitemos información adicional de los eventos del teclado.

Por ejemplo, cuando pulsamos una tecla nos interesa saber cuál ha sido la tecla pulsada, o si tenemos alguna tecla especial pulsada como Alt, Control, etc.

Para gestionar toda esa información disponemos del objeto **event**, el cual nos permitirá acceder a esas propiedades adicionales que se generan en los eventos.

Ejemplo:

```
const gestionar=(miEvento)=>{

        // Mostrará el tipo de evento en este caso es click.
        console.log(miEvento.type);
}
const elemento = document.getElementById("unparrafo");
elemento.addEventListener('click',gestionar)
```

En el código del ejemplo anterior cuando se produce el evento de click en un párrafo con id="unparrafo", se llamará a la función gestionar. En la función gestionar hemos creado un argumento que le llamamos miEvento, y es justamente en ese argumento que hemos puesto en la función, donde el navegador de forma automática pondrá todos los datos referentes al evento que se ha disparado.

Una vez dentro de la función, mostramos un mensaje con el tipo de evento (propiedad type del objeto **event**) que se acaba de disparar.

4.1.- Eventos del teclado en Javascript

Uno de los eventos más complicados de gestionar en JavaScript son los eventos de teclado, debido a que suele haber bastantes incompatibilidades entre navegadores, teclados, idiomas, etc.

Para el teclado disponemos de 3 tipos de eventos: ***keydown***, ***keypress*** y ***keyup***. Y además disponemos de dos tipos de teclas: las especiales (Shift, Alt, AltGr, Enter, etc.) y las teclas normales, que contienen letras, números, y símbolos.

En el proceso de pulsación de una tecla se generan tres eventos seguidos: keydown, keypress y keyup.

Y para cada uno de ellos disponemos de las **propiedades *keyCode*** y ***charCode***. Para saber la tecla que se ha pulsado lo más cómodo es acceder al evento **keypress**.

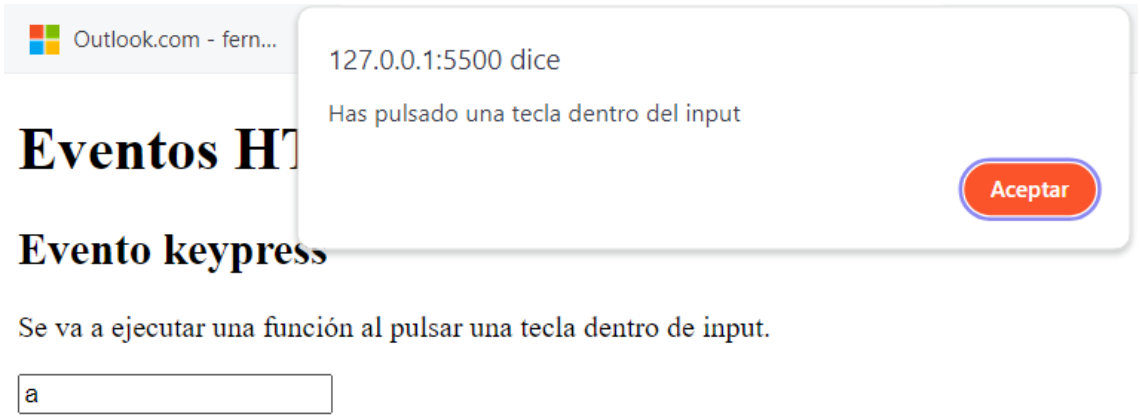
Ejemplo:

```
function miFuncion() {
    alert("Has pulsado una tecla dentro del input");
}

document.addEventListener("DOMContentLoaded", function() {
    let input = document.getElementById("miInput");
    input.addEventListener("keypress", miFuncion);
})
```

```
<h1>Eventos HTML DOM</h1>
<h2>Evento keypress</h2>

<p>Se va a ejecutar una función al pulsar una tecla dentro de input.</p>
<input type="text" id="miInput">
```



4.2.- Eventos del ratón en Javascript

Los eventos del ratón son uno de los **eventos más importantes** en JavaScript.

Cada vez que un usuario hace *click* en un elemento, al menos se disparan tres eventos y en el siguiente orden:

1. **mousedown,** cuando el usuario presiona el botón del ratón sobre el elemento.
2. **mouseup**, cuando el usuario suelta el botón del ratón.
3. **click**, cuando el usuario pulsa y suelta el botón sobre el elemento.

En general, los eventos de mousedown y mouseup son mucho más útiles que el evento click.

Si por ejemplo presionamos el botón sobre un elemento A, nos desplazamos y soltamos el botón sobre otro elemento B, se detectarán solamente los eventos de mousedown sobre A y mouseup sobre B, pero no se detectará el evento de click. Esto quizás pueda suponer un problema, dependiendo del tipo de interacción que quieras en tu aplicación.

Generalmente a la hora de registrar eventos, se suele hacer para ***mousedown*** y ***mouseup***, a no ser que quieras el evento de click y no los otros.

El evento de ***dblclick*** no se usa muy a menudo. Incluso si lo usas, tienes que ser muy prudente y no registrar a la vez click y dblclick sobre el mismo elemento, para evitar complicaciones.

El evento de ***mousemove*** funciona bastante bien, aunque debes tener en cuenta que la gestión de este evento le puede llevar cierto tiempo al sistema para su procesamiento.

Por ejemplo, si el ratón se mueve 1 píxel, y tienes programado el evento de mousemove, para cada movimiento que hagas, ese evento se disparará, independientemente de si el usuario realiza o no realiza ninguna otra opción. En ordenadores antiguos, esto puede ralentizar el sistema, ya que para cada movimiento del ratón estaría realizando las tareas adicionales programadas en la función. Por lo tanto, se recomienda utilizar este evento solo cuando haga falta, y desactivarlo cuando hayamos terminado.

Otros eventos adicionales del ratón son los de ***mouseover*** y ***mouseout***, que se producen cuando el ratón entra en la zona del elemento o sale del elemento. Si, por ejemplo, tenemos tres contenedores anidados divA, divB y divC: si programamos un evento de mouseover sobre el divA y nos vamos moviendo hacia el contenedor interno, veremos que ese evento sigue disparándose cuando estemos sobre divB o entremos en divC. Esta reacción se debe al burbujeo de eventos. Ni en divB o divC tenemos registrado el evento de mouseover, pero cuando se produce el burbujeo de dicho evento, se encontrará que tenemos registrado ese evento en el contenedor padre divA y por eso se ejecutará.

Muchas veces es necesario saber de dónde procede el ratón y hacia dónde va, y para ello W3C añadió la propiedad relatedTarget a los eventos de mouseover y mouseout. Esta propiedad contiene el elemento desde donde viene el ratón en el caso de mouseover, o el elemento en el que acaba de entrar en el caso de mouseout.

Propiedades de destino y origen del objeto ***Event***.

Para saber los botones del ratón que hemos pulsado, disponemos de las propiedades ***which*** y ***button***. Y para detectar correctamente el botón pulsado, lo mejor es hacerlo en los eventos de mousedown o mouseup. Which es una propiedad antigua de Netscape, así que simplemente vamos a citar button que es la propiedad propuesta por el W3C:

Los valores de la propiedad button pueden ser:

- ✓ **Botón izquierdo**: 0
- ✓ **Botón medio**: 1
- ✓ **Botón derecho**: 2

También es muy interesante conocer la posición en la que se encuentra el ratón, y para ello disponemos de un montón de propiedades que nos facilitan esa información:

- ✓ **clientX, clientY:** devuelven las coordenadas del ratón relativas a la ventana.
- ✓ **offsetX, offsetY:** devuelven las coordenadas del ratón relativas al objeto destino del evento.
- ✓ **pageX, pageY:** devuelven las coordenadas del ratón relativas al documento.
- ✓ **screenX, screenY:** devuelven las coordenadas del ratón relativas a la pantalla.

Ejemplo:

```
function capturarEvento(evento){
    let texto = "Has pulsado el botón: " + evento.button;
    document.getElementById("demo").innerHTML = texto;
  }

  document.addEventListener("DOMContentLoaded", function() {
    var elemento = document.querySelector('.botones');
    elemento.addEventListener('mousedown', capturarEvento);
  });
```

```
<div class="botones">
    <p>Haz click en la caja con los distintos botones del ratón.</p>
    <p>
        0 = Botón izquierdo<br>
        1 = Botón central<br>
        2 = Botón derecho
    </p>
</div>

<p id="demo"></p>
```

Evento de ratón

Propiedad button

Haz click en la caja con los distintos botones del ratón.

0 = Botón izquierdo
1 = Botón central
2 = Botón derecho

Has pulsado el botón: 1

Más ejemplos:

```
document.addEventListener("DOMContentLoaded", function() {

const button = document.querySelector("button");
//Al pulsar el botón, saldrá un saludo en pantalla
button.onclick = function () {
    alert("Hola!");
}

});
```

```
//Al pinchar el enlace, se abrirá la página de Google
document.addEventListener("DOMContentLoaded", () => {
    const link = document.getElementById("link");
    link.addEventListener('click', (event) => {
        event.target.setAttribute("href", "http://www.google.es");
    });
});
```

5.- ANEXOS

La siguiente tabla resume los eventos más importantes definidos por JavaScript:

Evento	Descripción	Elementos para los que está definido
`onblur`	Un elemento pierde el foco	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
`onchange`	Un elemento ha sido modificado	`<input>`, `<select>`, `<textarea>`
`onclick`	Pulsar y soltar el ratón	Todos los elementos
`ondblclick`	Pulsar dos veces seguidas con el ratón	Todos los elementos
`onfocus`	Un elemento obtiene el foco	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
`onkeydown`	Pulsar una tecla y no soltarla	Elementos de formulario y `<body>`
`onkeypress`	Pulsar una tecla	Elementos de formulario y `<body>`
`onkeyup`	Soltar una tecla pulsada	Elementos de formulario y `<body>`
`onload`	Página cargada completamente	`<body>`
`onmousedown`	Pulsar un botón del ratón y no soltarlo	Todos los elementos
`onmousemove`	Mover el ratón	Todos los elementos
`onmouseout`	El ratón "sale" del elemento	Todos los elementos
`onmouseover`	El ratón "entra" en el elemento	Todos los elementos

`onmouseup`	Soltar el botón del ratón	Todos los elementos
`onreset`	Inicializar el formulario	`<form>`
`onresize`	Modificar el tamaño de la ventana	`<body>`
`onselect`	Seleccionar un texto	`<input>`, `<textarea>`
`onsubmit`	Enviar el formulario	`<form>`
`onunload`	Se abandona la página, por ejemplo al cerrar el navegador	`<body>`