Unidad 7
Interacción con el usuario: eventos

CESUR
Tu Centro Oficial de FP

1. Eventos

EVENTOS – mecanismos por los que se consigue una comunicación bidireccional y en tiempo real entre la aplicación y los usuarios. Son sucesos que se han producido en una página, normalmente provocados por la acción del usuario.

Los eventos se asocian a elementos concretos del DOM.

1.- CAPTURA DE EVENTOS

Para capturar un evento se puede definir un listener (con addEventListener) sobre el elemento que queramos y por medio de un callback y del nombre del evento a capturar, ejecutar las acciones deseadas.

Ejemplo: párrafo que contiene un span y muestra por consola cuántas veces el usuario hace clic sobre dicho span

```
<p>
        Así se capturan eventos asociados
        a un <span id="miSpan">elemento</span> HTML.
</p>
```

```
let miSpan = document.getElementById("miSpan");
let veces = 0;
miSpan.addEventListener("click",()=>{
        veces++;
        console.log(`El usuario ha hecho ${veces} clic en el span.`);
});
```

```
El usuario ha hecho 1 clic en el span.
El usuario ha hecho 2 clic en el span.
El usuario ha hecho 3 clic en el span.
>
```

Cada evento tiene asociado un objeto, basado en la interfaz Event, que contiene propiedades y métodos para obtener más información sobre lo sucedido.

2.- OBJETO DEL EVENTO

Para utilizar el objeto asociado a un evento hay que indicarlo como parámetro del callback que gestiona el evento.

Propiedades del objeto evento:

Propiedad	Utilidad
altKey	Devuelve si la tecla [Alt] fue pulsada durante el evento.
button	Devuelve el botón del ratón que activó el evento: ■ 0: botón principal. ■ 1: botón central. ■ 2: botón secundario. ■ 3 y 4: cuarto y quinto botones (si los hubiera).
charCode	Contiene el valor Unicode de la tecla que se pulsó (evento **keypress**).
clientX	Coordenada X del ratón con respecto a la ventana.
clientY	Coordenada Y del ratón con respecto a la ventana.
ctrlKey	Devuelve si la tecla [Ctrl] fue pulsada durante el evento.
pageX	Coordenada X del evento, relativa al documento completo.
pageY	Coordenada Y del evento, relativa al documento completo.
screenX	Coordenada X del evento con respecto a la pantalla.
screenY	Coordenada Y del evento con respecto a la pantalla.
shiftKey	Devuelve si la tecla [Mayús] fue pulsada durante el evento.
target	Referencia al elemento que lanzó el evento.
timeStamp	Devuelve el momento en el que se creó el evento.
type	Nombre del evento.

Evento
https://developer.mozilla.org/es/docs/Web/API/Event

CESUR
Tu Centro Oficial de FP

Ejemplo 1: EVENTO

Crea un programa donde se capture el evento click sobre un elemento y se acceda al objeto del evento.

Con console.Info(), muestra la estructura del objeto en la consola.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <title>Ejemplo de JavaScript</title>
5       <meta charset="UTF-8">
6   </head>
7   <body>
8       <button id="miBoton">Haz click aquí</button>
9       <script>
10          var miBoton = document.getElementById('miBoton');
11          miBoton.addEventListener('click',function(event){
12              console.info(event);
13          });
14      </script>
15  </body>
16  </html>
```

```
▼ PointerEvent i
    isTrusted: true
    altKey: false
    altitudeAngle: 1.5707963267948966
    azimuthAngle: 0
    bubbles: true
    button: 0
    buttons: 0
    cancelBubble: false
    cancelable: true
    clientX: 81
    clientY: 23
    composed: true
    ctrlKey: false
    currentTarget: null
    defaultPrevented: false
    detail: 1
    eventPhase: 0
    fromElement: null
    height: 1
    isPrimary: false
    layerX: 81
    layerY: 23
    metaKey: false
    movementX: 0
    movementY: 0
    offsetX: 72
    offsetY: 13
    pageX: 81
    pageY: 23
    pointerId: 1
    pointerType: "mouse"
    pressure: 0
    relatedTarget: null
    returnValue: true
    screenX: 97
    screenY: 121
    shiftKey: false
```

3.- PROPAGACIÓN DE EVENTOS

Los eventos pueden gestionarse desde un elemento más profundo hasta la superficie (comportamiento de burbuja), es decir, en una jerarquía de elementos, pueden capturarse eventos desde el nivel más profundo hasta el nivel más superficial, cuando se activa el evento más profundo.

```
<section>
        <p>
                Así se capturan eventos asociados
                a un <span id="miSpan">elemento</span> HTML.
        </p>
</section>
```

```
let miSection = document.getElementsByTagName("section")[0];
let miP = document.getElementsByTagName("p")[0];
let miSpan = document.getElementById("miSpan");
miSection.addEventListener("click",()=>{
        console.log("<section>: Capturado el evento.");
});
miP.addEventListener("click",()=>{
        console.log("<p>: Capturado el evento.");
});
miSpan.addEventListener("click",()=>{
        console.log("<span>: Capturado el evento.");
});
```

```
<span>: Capturado el evento.
<p>: Capturado el evento.
<section>: Capturado el evento.
>
```

A pesar de que se han definido los eventos de fuera hacia dentro (section, p y span), los eventos se han ejectuado en orden inverso, de dentro hacia fuera (span, p y section). Este es el comportamiento predeterminando de la propagación.

4.- CANCELACIÓN DE EVENTOS

EVENTOS PREDETERMINADOS – elementos que tienen asignados por defecto un evento de forma predeterminada. Por ejemplo: <a> tiene asociado por defecto el evento click.

preventDefault() – si se invoca sobre el objeto asociado al evento, no realiza la acción que tiene definida de forma predeterminada, es decir, se anula su comportamiento predeterminado.

removeEventListener() – anula completamente el evento. Solo se puede anular un evento cuando se utilizan funciones con identificador.

5.- LANZAR EVENTOS

Event – para crear un evento

dispatchEvent – para lanzar un evento

```
<a id="miEnlace" href="https://www.paraninfo.es">Enlace.</a>
<span id="miSpan">Ejecutar el evento del enlace.</span>
```

```
let miSpan = document.getElementById("miSpan");
let miEnlace = document.getElementById("miEnlace");
miEnlace.addEventListener("click",(evento)=>{
        evento.preventDefault();
        console.log("Gestor del evento del enlace.");
});
miSpan.addEventListener("click",()=>{
        let miEvento = new Event("click");
        console.log("Lanzando evento.");
        miEnlace.dispatchEvent(miEvento);
});
```

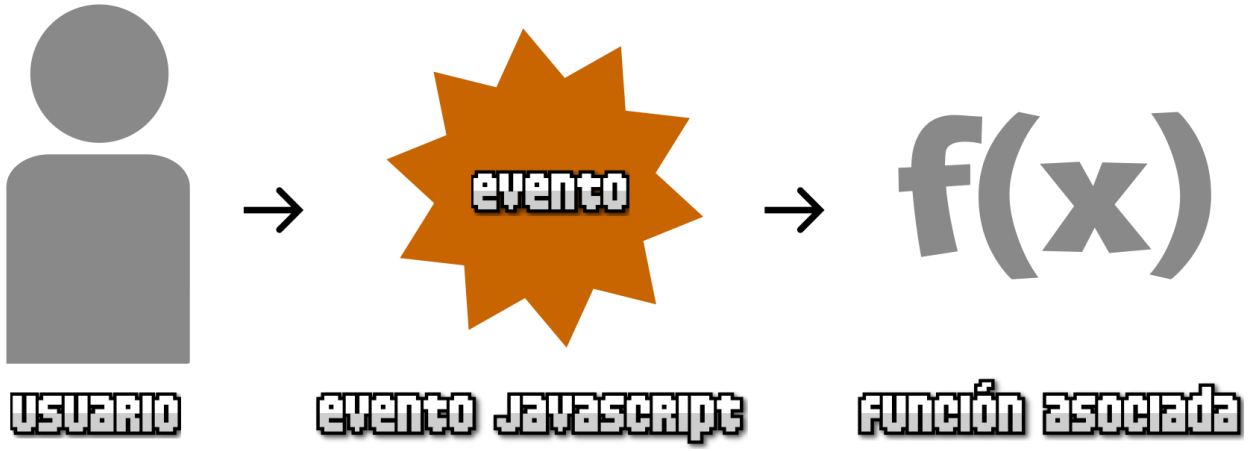
```
Lanzando evento.
Gestor del evento del enlace.
>
```

2. Tipos de eventos

Existen un amplio catálogo de eventos que pueden capturarse por JS.

Los más comunes son:

1) Eventos de formulario
2) Eventos de ratón
3) Eventos de teclado
4) Eventos de arrastrar y soltar
5) Eventos de reproducción multimedia
6) Otros eventos (carga de elementos, historial de sesión, ventana, animación, impresión, portapapeles, etc.)



3. Eventos de formulario

FORMULARIOS – entradas de datos de las aplicaciones web. Todos los elementos de un formulario normalmente se encuentran anidados en el elemento **<form>**.

El objeto document incluye la propiedad **forms**, que contiene todos los formularios de un documento. El primer formulario está en **forms[0]**, el segundo **forms[1]**, etc. Se necesita utilizar **document.getElementsByTagName("form")** para trabajar con ellos.

Cuando se invoca a **document.forms** se obtiene una colección de objetos form (propiedades y métodos):

Propiedad	Utilidad
action	Contiene la URL que recibirá los datos del formulario.
elements	Contiene todos los controles del formulario.
length	Número de controles del formulario.
method	{GET|POST} en función del método elegido para enviarlo.
enctype	Tipo de codificación de los datos del formulario.
acceptCharset	Conjunto de caracteres del formulario.
submit()	Envía los datos del formulario a la URL de **action** usando **method.**
reset()	Devuelve el formulario a su estado inicial.
on*X*	Todos los eventos asociados al formulario, siendo ***X*** el nombre del evento: **onAbort, onBlur, onCancel, onClick…**

CESUR
Tu Centro Oficial de FP

Aparte de trabajar con los formularios como conjunto, cabe la posibilidad de trabajar individualmente sobre sus controles.

Algunas de las propiedades más utilizadas de los controles de formulario son:

Propiedad	Utilidad
accept	Tipos de archivos que se permiten en un control de tipo **file.**
autocomplete	Si el valor del control puede ser autocompletado por el navegador.
name	Nombre del control.
type	Tipo de control.
value	Valor actual del control.
checked	**true** si el control está activado, **false** si no lo está.
defaultChecked	Valor predeterminado de la propiedad **checked.**
disabled	**true** si el control está deshabilitado, **false** si está habilitado.
hidden	El control es invisible para el usuario, pero no para el programador.
readonly	**true** si el control es de solo lectura (no modificable), **false** si no lo es.
required	**true** si proporcionarle un valor al control es obligatorio, **false** si no lo es.
maxLength	Anchura máxima del texto.
min	Valor mínimo para el control.
max	Valor máximo para el control.
pattern	Una expresión regular contra la que el valor del control es evaluado.
placeholder	Una pista para el usuario sobre lo que debe introducir en el control.
size	Tamaño inicial del control.
step	Tamaño del cambio en el valor de un control.
selectionStart	En una selección de texto la posición del primer carácter seleccionado.
selectionEnd	En una selección de texto, la posición del último carácter seleccionado.

3. Eventos de formulario

1.- ENVIAR FORMULARIOS

submit – evento para enviar los datos que contiene el formulario a la url indicada en el atributo **action**.

Hay que tener cuidado con las validaciones de los datos ya que hay que realizar la validación antes de que se produzca el envío del formulario. Esto se puede realizar mediante un framework (Bootstrap, Materialize, Semantic UI, etc. Pero si queremos hacerlo directamente con JS debemos cancelar el comportamiento predeterminado de **submit** para que los datos no se envíen antes de hacer la validación.

```
<form name="formulario" action="procesa.php" method="POST">
        Introduzca una edad mayor de 18 y menor de 65:
        <input type="text" name="edad" placeholder="Edad" value="" />
        <input type="submit" name="submit" value="Enviar >" />
</form>
<div id="errores"></div>
```

```
let formularioEdad = document.forms[0];
let errores = document.getElementById("errores");
formularioEdad.addEventListener("submit",(evento)=>{
        let edad = parseInt(document.forms[0].elements["edad"].value);
        if (edad<=18 || edad>=65) {
                evento.preventDefault();
                errores.innerHTML="La edad debe ser > 18 y < 65";
        }
});
```

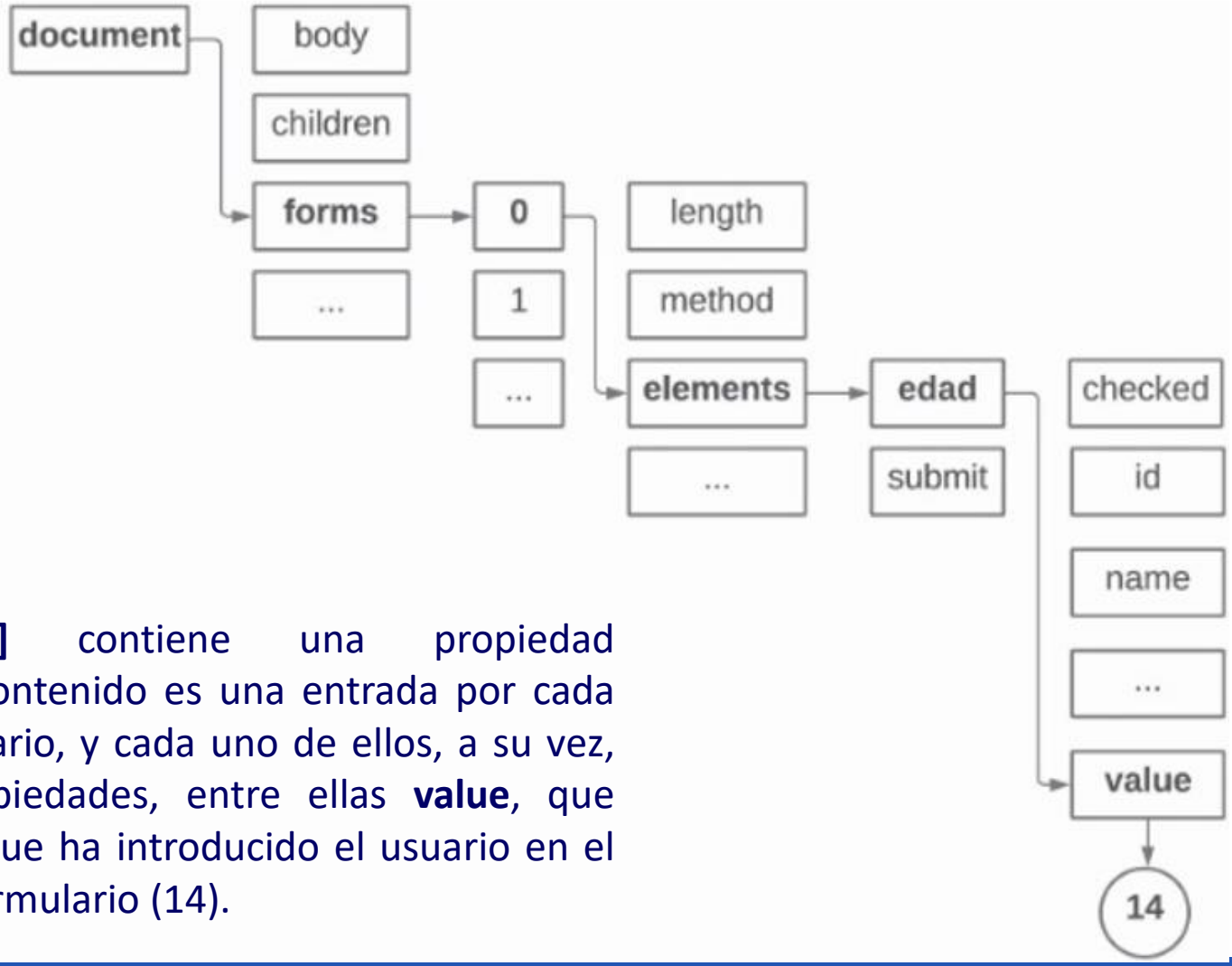
Lo que valida es que 18 < edad < 65 antes de que los datos se envíen.

Para ello se captura el evento **submit**, se realiza la comprobación y si edad no cumple con las restricciones, cancela el envío y muestra un mensaje de error.

Para conocer el valor que ha introducido el usuario en el campo **edad** se utilizar el contenido del objeto **document.forms[0].**

3. Eventos de formulario

Diagrama de la cadena jerárquica de propiedades del objeto forms



document.forms[0] contiene una propiedad (**elements**) cuyo contenido es una entrada por cada control del formulario, y cada uno de ellos, a su vez, otra lista de propiedades, entre ellas **value**, que contiene el valor que ha introducido el usuario en el campo edad del formulario (14).

3. Eventos de formulario

2.- RESETEO DE FORMULARIOS

reset – evento que permite dejar el formulario con sus valores iniciales.

3.- CAMBIAR UN VALOR

change – evento que se lanza cuando se realiza un cambio de valor en un control del formulario. Además, provoca que abandone el foco.

NOTA: **change** no se lanza al cambiar el valor de un control, sino justo cuando el foco abandona el control que se ha cambiado.

4.- GANAR Y PERDER EL FOCO

GANAR EL FOCO – en los formularios significa activar un control, a través del ratón, el teclado, etc. Esto nos permite interactuar directamente con el control.

PERDER EL FOCO – en los formularios significa abandonar la posibilidad de interactuar directamente con el control.

focus – evento que se lanza cuando un control gana el foco.

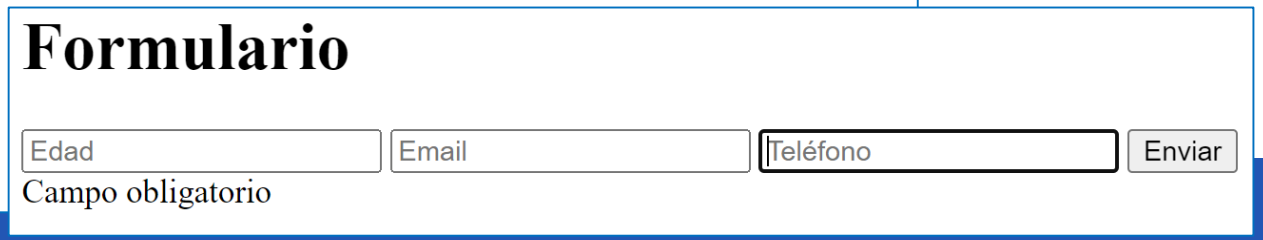
blur – evento que se lanza cuando un control pierde el foco.

rmulario

Ejemplo 2: FORMULARIOS

Crea una página que incluya un formulario con tres controles, dos obligatorios, de manera que cuando cada control obligatorio gane el foco se muestre un mensaje indicándolo y cuando pierda el foco, elimine todos los mensajes mostrados al usuario.

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de JavaScript</title>
    <meta charset="UTF-8">
</head>
<body>
    <h1>Formulario</h1>
    <form name="formulario" action="#" method="post">
        <input type="text" name="edad" placeholder="Edad">
        <input type="text" name="email" placeholder="Email" required>
        <input type="text" name="tlf" placeholder="Teléfono" required>
        <input type="submit" name="submit" value="Enviar"
    </form>
    <div id="errores"></div>
    <script>
        let controles = document.forms[0].elements;
        let numEltos = document.forms[0].length;
        for (let i=0; i<numEltos; i++){
            controles[i].addEventListener("focus",()=>{
                if (controles[i].hasAttribute("required")){
                    document.getElementById("errores").innerHTML="Campo obligatorio";
                }
            });
            controles[i].addEventListener("blur",()=>{
                if (controles[i].hasAttribute("required")){
                    document.getElementById("errores").innerHTML="";
                }
            });
        }
    </script>
</body>
</html>
```



3. Eventos de ratón

Los eventos son producidos no sólo por el ratón, sino por cualquier dispositivo apuntador capaz de mover el cursor por la pantalla. Por ejemplo, el dedo en una pantalla táctil.

Los eventos de ratón más usados son:

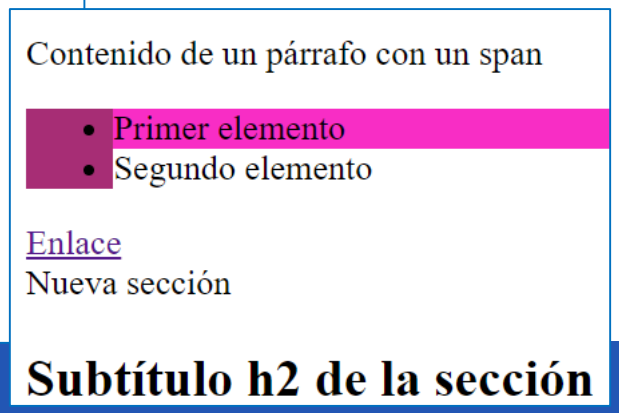
Evento	Funcionamiento
click	Hacer clic sobre el botón principal del dispositivo.
dblclick	Hacer doble clic sobre el botón principal del dispositivo.
mousedown	Cuando se pulsa y justo antes de soltarlo, se lanza el evento.
mouseup	El evento se lanza cuando se suelta el botón.
mousenter	El evento se lanza cuando el puntero se sitúa sobre el elemento que captura el evento.
mouseleave	El evento se lanza cuando el puntero deja de situarse sobre el elemento que captura el evento.
mousemove	Mientras se está dentro del elemento, el evento se lanza cada vez que se mueve el puntero.
mouseover	El evento se lanza cuando el puntero se sitúa sobre el elemento que lo captura o sobre cualquiera de sus hijos.
mouseout	El evento se lanza cuando el puntero deja de situarse sobre el elemento que lo captura o sobre cualquiera de sus hijos.
contextmenu	El evento se lanza cuando se solicita un menú contextual.

ventos de ratón

Ejemplo 3: EVENTOS DE RATÓN

Crea un programa que, a partir de diversos elementos HTML, los coloree con colores aleatorios cada vez que se coloque el ratón sobre ellos y los vuelva a colorear de blanco cuando el ratón los abandone.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de JavaScript</title>
    <meta charset="UTF-8">
</head>
<body>
    <p>Contenido de un párrafo con un <span>span</span></p>
    <ul>
        <li>Primer elemento</li>
        <li>Segundo elemento</li>
    </ul>
    <a href="#">Enlace</a>
    <section>Nueva sección</section>
    <h2>Subtítulo h2 de la sección</h2>
    <script>
        let elementos = document.getElementsByTagName("*");
        for (let i=5; i<elementos.length;i++){
            elementos[i].addEventListener("mouseenter",()=>{
                let r = Math.floor(Math.random()*255);
                let g = Math.floor(Math.random()*255);
                let b = Math.floor(Math.random()*255);
                elementos[i].style.backgroundColor=`rgb(${r},${g},${b})`;
                console.log(elementos);
            });
            elementos[i].addEventListener("mouseleave",()=>{
                elementos[i].style.backgroundColor="#FFFFFF";
            });
        }
    </script>
</body>
</html>
```



4. Eventos de teclado

key – objeto que almacena información de la tecla pulsada, independientemente de que se pulse sola o en combinación con [Mayús], [AltGr], [Ctrl], etc.

Podemos comprobar si se pulsó alguna de las teclas de combinación con **AltKey**, **CtrlKey**, **ShiftKey** o **MetaKey** (tecla especial de ordenadores Apple).

Cuando existen varias teclas con la misma función (Ctrl, Mayús) con **location** podemos saber qué tecla se pulsó. Los posibles valores de **location** son:

Valor	Funcionamiento	Constante asociada
0	Teclado estándar	DOM_KEY_LOCATION_STANDARD
1	Parte izquierda	DOM_KEY_LOCATION_LEFT
2	Parte derecha	DOM_KEY_LOCATION_RIGHT
3	Teclado numérico	DOM_KEY_LOCATION_NUMPAD

Los eventos existentes para gestionar el comportamiento del usuario respecto al teclado son:

Evento	Funcionamiento
keypress	El evento se lanza tras pulsar y soltar una tecla.
keydown	El evento se lanza tras pulsar y antes de soltar la tecla.
keyup	El evento se lanza tras soltar la tecla.

Códigos de teclas JS
https://www.freecodecamp.org/espanol/news/lista-de-codigos-de-teclas-en-javascript/

5. Eventos de arrastrar y soltar

Los objetos arrastrables son eventos que se lanzan al utilizar elementos definidos como arrastrables, es decir, que tengan establecido a true el atributo **draggable**.

Los eventos asociados a la acción arrastrar y soltar son:

Evento	Funcionamiento
Elemento que se arrastra	
drag	Cada vez que se arrastra una vez que se ha iniciado el arrastre.
dragstart	Al comenzar a arrastrar el elemento.
dragstop	Al finalizar el arrastre del elemento.
Elemento donde puede soltarse	
dragenter	Cuando el elemento que se arrastra entra en el elemento destino.
dragleave	Cuando el elemento que se arrastra sale del elemento destino.
dragover	Cada vez que continúa el arrastre sobre el elemento de destino.
drop	Cuando el elemento que se arrastra se suelta sobre el elemento destino.

6. Eventos de reproducción multimedia

Son eventos que pueden capturarse cuando se reproducen contenidos multimedia:

Evento	Momento en el que se activa
canplay	Cuando se detecta que el contenido se puede comenzar a reproducir.
canplaythrough	Cuando se estima que el contenido tiene cargados datos suficientes como para poder reproducirse.
durationchange	Cuando se modifica el atributo **duration** del medio.
emptied	Cuando se vacía de datos el medio.
ended	Cuando se detiene la reproducción, sin intervención del usuario.
loadeddata	Cuando se ha cargado la primera imagen de un vídeo.
loadedmetadata	Cuando se han cargado los metadatos del medio.
pause	Cuando se pausa el medio de reproducción.
play	Cuando se reanuda la reproducción tras una pausa.
playing	Cuando el medio está listo para reproducirse tras una parada.
ratechange	Cuando se modifica el **rate** del vídeo en reproducción.
seeked	Cuando finaliza la búsqueda en el medio.
seeking	Cuando se inicia la búsqueda en el medio.
stalled	Cuando se produce un fallo en la carga, pero la carga continúa.
suspend	Cuando se suspende la carga del medio.
timeupdate	Cuando se modifica el valor de **currentTime** del medio.
volumechange	Cuando se modifica el volumen.
waiting	Cuando la reproducción se detiene por falta de datos.

7. Otros eventos

Otros eventos que pueden ser interesantes son:

Evento	Momento en el que se activa
Carga de elementos	
abort	Cuando se cancela la carga de un elemento.
DOMContentLoaded	Cuando se ha cargado el documento HTML.
error	Cuando se produce un error en la carga.
load	Cuando se ha cargado el documento y los elementos externos que incorpora: imágenes, hojas de estilo…
progress	Cuando se está produciendo la carga.
readystatechange	Cuando se modifica el valor del atributo **readystate.**
Historial de la sesión	
popstate	Cuando se cambia el historial.
pagehide	Cuando se esconde la página actual mientras se visualizan las distintas páginas de la sesión.
pageshow	Cuando el navegador muestra el documento en la ventana.
Ventana	
scroll	Cuando se desplaza la ventana por medio de las barras de desplazamiento.
resize	Cuando se modifica el tamaño de la ventana.

7. Otros eventos

Evento	Momento en el que se activa
Animación	
animationend	Cuando finaliza la animación.
animationiteration	Cuando se repite la animación.
animationstart	Cuando comienza la animación.
Impresión	
afterprint	Cuando ha comenzado la impresión o se ha cerrado la previsualización de la impresión.
beforeprint	Cuando va a comenzar la impresión o se ha abierto la previsualización de la impresión.
Portapapeles	
copy	Cuando se va a copiar justo antes de ejecutar la acción.
cut	Cuando se va a cortar justo antes de ejecutar la acción.
paste	Cuando se va a pegar justo antes de ejecutar la acción.

Eventos en JS
https://developer.mozilla.org/es/docs/Web/Events#categor%-C3%ADas

Gracias