

Esta clase va a ser

- grabada

Clase 10. JAVASCRIPT

Eventos

CLASE N°8

Glosario

DOM o Modelo de Objetos del Documento:

es lo que permite interactuar a JS con los diferentes elementos HTML de una web, como también poder operar sobre ellos y modificarlos.

JavaScript Object Notation (JSON):

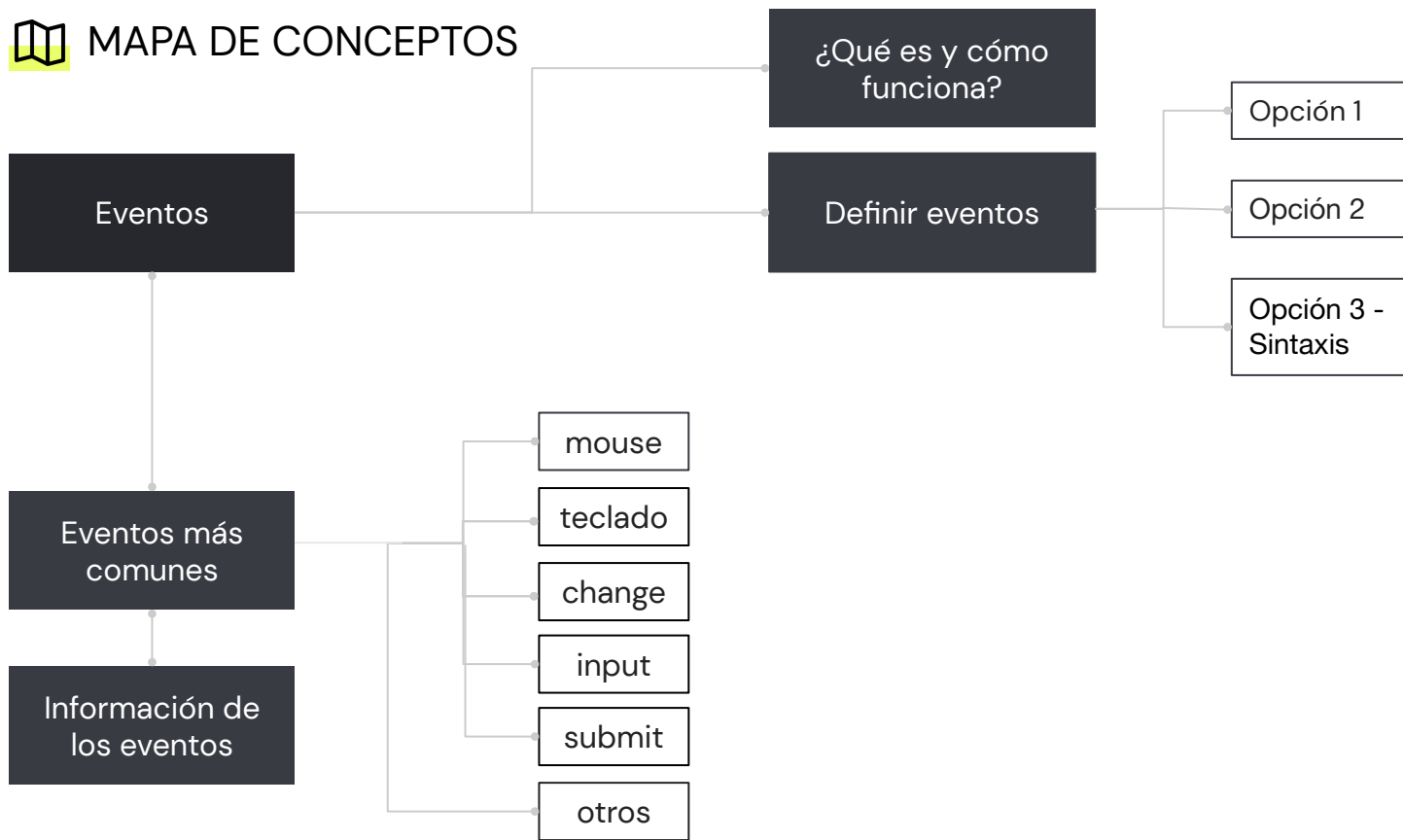
es un formato basado en texto plano, para representar datos estructurados en la sintaxis de objetos de JavaScript. Es comúnmente utilizado para transmitir datos en aplicaciones web.

Objetivos de la clase

- **Comprender** qué son los eventos y para qué sirven.
- **Aprender** cuáles son las tres opciones para definir eventos.
- **Entender** cómo escuchar un evento sobre el DOM.
- **Conocer** los eventos más comunes.
- **Entender** cómo escuchar un evento sobre el DOM.



MAPA DE CONCEPTOS



¡Vamos a la clase!



Eventos en JS



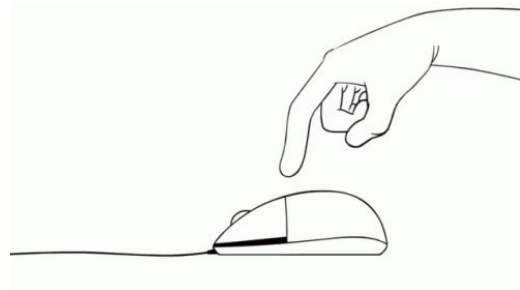
¿Qué es un evento?

Los **eventos** son la manera que tenemos en Javascript de controlar las acciones de los usuarios, y definir un comportamiento de la página o aplicación cuando se produzcan.

Con Javascript es posible definir qué sucede cuando se produce un evento, por ejemplo, cuando se realiza un clic en cierto elemento o se inserta un texto en un determinado campo.

¿Cómo funciona?

JavaScript permite **asignar una función a cada uno de los eventos**. Reciben el nombre de **event handlers** o **manejadores de eventos**. Así, ante cada evento, JavaScript asigna y ejecuta la **función asociada** al mismo.



¿Cómo funciona?

Hay que entender que los eventos **sucedan** constantemente en el navegador.

JavaScript lo que nos permite hacer es **escuchar** eventos sobre elementos seleccionados.

Cuando escuchamos el evento que esperamos, se ejecuta la función que definimos en **respuesta**.

A esta escucha se la denomina **event listener**.

¿Cómo definir eventos en JS?

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi primer App</title>
  </head>
  <body>
    <h2>Coder House</h2>
    <button id="btnPrincipal">CLICK</button>
    <script>
      let boton =
document.getElementById ("btnPrincipal")
      boton.addEventListener ("click", respuestaClick)
      function respuestaClick () {
        console.log ("Respuesta evento");
      }
    </script>
  </body>
</html>
```

Opción 1

El método **addEventListener()** permite definir qué evento escuchar sobre cualquier elemento seleccionado. El primer parámetro corresponde al nombre del evento y el segundo a la función de respuesta.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi primer App</title>
  </head>
  <body>
    <h2>Coder House</h2>
    <button id="btnPrincipal">CLICK</button>
    <script>
      let boton =
document.getElementById("btnPrincipal")
      boton.onclick = () =>{console.log("Respuesta 2")}
    </script>
  </body>
</html>
```

Opción 2

Emplear una propiedad del **nodo** para definir la respuesta al evento. Las propiedades se identifican con el **nombre del evento** y el **prefijo on**. También es posible emplear funciones anónimas para definir los manejadores de eventos.

Opción 3: Sintaxis

Determinar el evento especificando el **manejador de evento** en el atributo de una etiqueta HTML. La denominación del atributo es idéntica al de la propiedad de la opción 2 (prefijo on).

La función puede declararse entre la comillas o bien tomarse una referencia existen en el script.

```
<input type="button" value="CLICK2" onclick="alert('Respuesta 3');"  
/>
```



¿Y cuál conviene usar?

Las opciones 1 y 2 son las recomendadas.

Si bien se pueden presentar casos de aplicación específicos (por ejemplo, en la opción 1 el nombre del evento puede venir de una variable al usar la propiedad, y esto no puede hacerse en la 2), se identifican como formas de definición de **eventos equivalentes**.

¿Y cuál conviene usar?

La opción 3, aunque es de fácil implementación, no es recomendada para proyectos en producción.

No se considera **buena práctica** declarar funciones y código JavaScript dentro del HTML.





Ejemplo en vivo

¡Vamos a practicar lo visto!

Duración: **10 minutos**

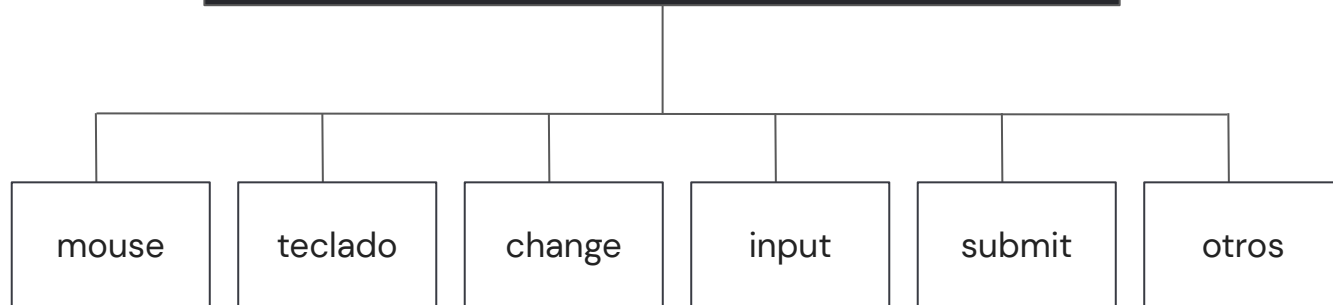


Break

¡10 minutos y volvemos!

Eventos más comunes

Eventos más comunes



Eventos del mouse

Se producen por la interacción del usuario con el mouse.

Entre ellos se destacarán los que se encuentran a continuación.



Eventos del mouse

- ✓ **mousedown/mouseup**: Se oprime/suelta el botón del ratón sobre un elemento.
- ✓ **mousemove**: El movimiento del mouse sobre el elemento activa el evento.
- ✓ **mouseover/mouseout**: El puntero del mouse se mueve sobre/sale del elemento.
- ✓ **click**: Se activa después de mousedown o mouseup sobre un elemento válido.

```
//CODIGO HTML DE REFERENCIA
<button id="btnMain">CLICK</button>

//CODIGO JS
let boton = document.getElementById("btnMain")
boton.onclick = () => {console.log("Click")}
boton.onmousemove = () => {console.log("Move")}
```



Eventos del teclado

Se producen por la interacción del usuario con el teclado.

Entre ellos se destacarán los que se encuentran a continuación.

Eventos de teclado

```
//CODIGO HTML DE REFERENCIA
<input id = "nombre" type="text">
<input id = "edad" type="number">

//CODIGO JS
let input1 = document.getElementById("nombre")
let input2 = document.getElementById("edad")
input1.onkeyup = () => {console.log("keyUp")}
input2.onkeydown = () =>
{console.log("keyDown")}
```

- ✓ **keydown:** Cuando se presiona.
- ✓ **keyup:** Cuando se suelta una tecla.

Evento change

```
//CODIGO HTML DE REFERENCIA
<input id = "nombre" type="text">
<input id = "edad" type="number">

//CODIGO JS
let input1 = document.getElementById("nombre");
let input2 = document.getElementById("edad");
input1.onchange = () => {console.log("valor1")};
input2.onchange = () => {console.log("valor2")};
```

El evento **change** se activa cuando se detecta un cambio en el valor del elemento.

Por ejemplo, mientras se escribe en un input de tipo texto **no hay evento change**, pero cuando se pasa a otra sección de la aplicación entonces sí ocurre.

Elemento input

```
//CODIGO HTML DE REFERENCIA
<input id = "nombre" type="text">

//CODIGO JS
let input1 = document.getElementById("nombre")
input1.addEventListener('input', () => {
    console.log(input1.value)
})
```

Si queremos ejecutar una función cada vez que se tipea sobre el campo, conviene trabajar directamente con el evento **input**.

Evento submit

```
//CODIGO HTML DE REFERENCIA
<form id="formulario">
  <input type="text">
  <input type="number">
  <input type="submit" value="Enviar">
</form>

//CODIGO JS
let miFormulario =
document.getElementById("formulario");
miFormulario.addEventListener("submit", validarFormulario);

function validarFormulario(e) {
  e.preventDefault();
  console.log("Formulario Enviado");
}
```

El evento **submit** se activa cuando el formulario es enviado. Normalmente se utiliza para validar el formulario antes de ser enviado al servidor o bien para abortar el envío y procesarlo con JavaScript.



Otros eventos

Existen otros eventos que podemos utilizar.

Algunos son **eventos estándar** definidos en las especificaciones oficiales, mientras que otros son eventos usados internamente por **navegadores específicos**.

Otros eventos

La forma de declararlos es similar a lo abordado en esta clase, lo que necesitamos aprender es **bajo qué condición se disparan los eventos que buscamos implementar.**

Para conocer más eventos se recomienda verificar la [referencia de eventos](#) en la documentación.

Información del evento

Información del evento

En algunos casos, necesitamos obtener **información contextual** del evento para poder realizar acciones.

Por ejemplo, ante el evento submit necesitamos prevenir el comportamiento por defecto para operar correctamente.

Para esto existe en JavaScript el **objeto event**.

Información del evento

En todos los navegadores modernos se crea de forma automática un parámetro que se pasa a la función manejadora, por lo que no es necesario incluirlo en la llamada.

Ese parámetro puede o no usarse en el handler, **pero siempre estará disponible en la llamada.**


```
//CODIGO HTML DE REFERENCIA
<form id="formulario">
  <input type="text">
  <input type="number">
  <input type="submit" value="Enviar">
</form>

//CODIGO JS
let miFormulario =
document.getElementById("formulario");
miFormulario.addEventListener("submit", validarFormulario);

function validarFormulario(e){
  //Cancelamos el comportamiento del evento
  e.preventDefault();
  //Obtenemos el elemento desde el cual se disparó el
evento
  let formulario = e.target
  //Obtengo el valor del primero hijo <input type="text">
  console.log(formulario.children[0].value);
  //Obtengo el valor del segundo hijo <input type="number">
  console.log(formulario.children[1].value);
}
```

Ejemplo aplicado:

Datos del formulario usando event

¡Vamos a practicar lo visto!





#CoderAlert

Encontrarás en la [Guía de Actividades](#) del curso un ejercicio para aplicar todo lo aprendido hoy sobre **Eventos** a tu Proyecto. ¡Será fundamental al momento de realizar tu primera pre entrega en clase N°**13**!



Interactuar con HTML

Consigna

Traslada al proyecto integrador el concepto de objetos, visto en la clase de hoy. En función del tipo de simulador que hayas elegido, deberás:

- ✓ Crear elementos manipulando el DOM a partir de la información de tus objetos.
- ✓ Modificar etiquetas existentes en función del resultado de operaciones.

Formato

- ✓ Página HTML y código fuente en JavaScript.

Podrás encontrar un ejemplo en la carpeta de clase.



Interactuar con HTML

Aspectos a incluir

- ✓ Archivo HTML y Archivo JS, referenciado en el HTML por etiqueta `<script src="js/miarchivo.js"></script>`, que incluya la definición de un algoritmo en JavaScript que opere sobre el DOM, modificando, agregando o eliminado elementos.

Sugerencias

- ✓ Generalmente, identificamos a un único elemento del DOM con el atributo `id` y a un conjunto asociado por `class`.

Ejemplo

- ✓ Podemos crear elementos HTML en función del listado de nuestros objetos identificados en la clase 6.
- ✓ Establecer un mensaje de bienvenida aleatorio usando un array de mensajes.
- ✓ Capturar una o más entradas por `prompt()` y mostrarlas en el HTML, modificando el DOM

Podrás encontrar un ejemplo en la carpeta de clase.

¿Preguntas?



Para pensar

¿Te gustaría comprobar tus conocimientos de la clase?

Te compartimos a través del chat de Zoom el enlace a un breve quiz de tarea.

Para el profesor:

Acceder a la carpeta "Quizzes" de la camada.

Ingresar al formulario de la clase.

Pulsar el botón "Invitar".

Copiar el enlace.

Compartir el enlace a los alumnos a través del chat.



MATERIAL AMPLIADO

Recursos multimedia

Ejemplos interactivos: Eventos

- ✓ [Introducción a eventos del navegador](#)
- ✓ [Acciones predeterminadas del navegador](#)
- ✓ [Eventos change, input, cut, copy, paste](#)
- ✓ [Formularios: evento y método submit](#)

Documentación

- ✓ [Documentación Eventos](#)
- ✓ [Referencia de Eventos](#)

Resumen de la clase hoy

- ✓ Eventos: qué son y para qué sirven.
- ✓ Tipos de eventos.
- ✓ Parámetro del evento y eventos más comunes.
- ✓ Información del evento.

Opina y valora
esta clase

Muchas gracias.

#DemocratizandoLaEducación