

**JS**



# JAVASCRIPT – parte 2

# Índice

- **Introducción a Eventos**
- **Trabajando con Eventos**
- **Formularios y expresiones regulares**
- **Objeto Date**
- **Temporizadores**
- **Prototipos**
- **Symbols**
- **Maps**
- **Sets**
- **APIs HTML5**
- **Prácticas finales**



# EVENTOS

Un evento es cualquier cosa que sucede en nuestro documento.

- El contenido se ha leído
- El contenido se ha cargado
- El usuario mueve el ratón
- El usuario pulsa una tecla
- La ventana se ha cerrado
- Y un largo etc.

~~`<p onclick="saludo()">Soy un párrafo</p>`~~

`<p (click)="saludo()">Soy un párrafo</p>`

No hacerlo por varios motivos:

- Para no mezclar HTML y Javascript.
- Difícil ejecutar más de un evento de esta forma.

➡ **ANGULAR**

`Element.addEventListener('event', callback)` ← **CORRECTO**

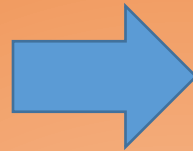
# LISTADO DE EVENTOS POSIBLES

<https://developer.mozilla.org/es/docs/Web/Events>

- Los eventos que están en color azul son aquellos que funcionan y no tienen ningún error, por ejemplo “click”.
- Los de color rojo son aquellos que presentan algún error que no se ha corregido
- Los del pulgar hacia abajo están obsoletos y no se utilizan.

index.html

```
<button id="button">Click Me!</button>  
  
<script src="scripts.js"></script>
```



scripts.js

```
const button = document.getElementById('button')  
  
button.addEventListener('click', () => {  
  console.log('CLICK');  
})
```

Función de flecha para indicar lo que queremos que ocurra cuando suceda el evento

# EVENTOS QUE VAMOS A TRABAJAR

## Eventos de ratón:

click - cuando pulsamos el botón izquierdo del ratón

dblclick - cuando pulsamos dos veces seguidas el botón izquierdo del ratón

mouseenter - cuando entramos en la zona que tiene el evento

mouseleave - cuando salimos de la zona que tiene el evento

mousedown - cuando pulsamos el botón izquierdo del ratón

mouseup - cuando soltamos el botón izquierdo del ratón

mousemove - cuando movemos el ratón

## Eventos de teclado:

keydown - cuando pulsamos una tecla

keyup - cuando soltamos una tecla

keypress - cuando pulsamos una tecla y no la soltamos



# EJERCICIO BÁSICO 1

- Crea una página **index.html** que llame a un archivo javascript cuyo nombre sea **scripts.js** y una hoja de estilo cuyo nombre sea **styles.css**
- En el archivo index.html crea un div que tenga una clase que se llame box y cuyo id se llame box. En la hoja de estilo defina esta clase con los siguientes valores:
  - width: 100px
  - height: 100px
  - background: red
- Cuando el ratón entre dentro de la caja, se cambiará el color a verde.
- Cuando el ratón salga de la caja el color deberá cambiar a rojo.
- Cuando pulsemos el botón izquierdo del ratón estando situados sobre la caja, aparecerá por consola el mensaje “Has pulsado la caja.
- Al soltar el botón izquierdo del ratón en la caja, aparecerá por consola el mensaje “Has soltado el botón izquierdo dentro de la caja”.

# EJERCICIO BÁSICO 2

- Sobre el ejercicio anterior añade un input de tipo texto.
  - Al pulsar una tecla deberá aparecer el mensaje por consola “Has pulsado una tecla”
  - Al soltar la tecla deberá aparecer el mensaje por consola “Has soltado una tecla”

**¿Cómo podemos saber qué tecla concreta se ha pulsado?, investiga que tendrías que utilizar para detectar la tecla concreta pulsada y muéstrala en el mensaje por consola.**

# EJERCICIO BÁSICO 3

- Crea un formulario con un input de tipo texto y un botón “Enviar”. Al pulsar el botón. Crea un evento para que al soltar una tecla se lance una función que vaya mostrando por consola todo lo que se escribe en el input.

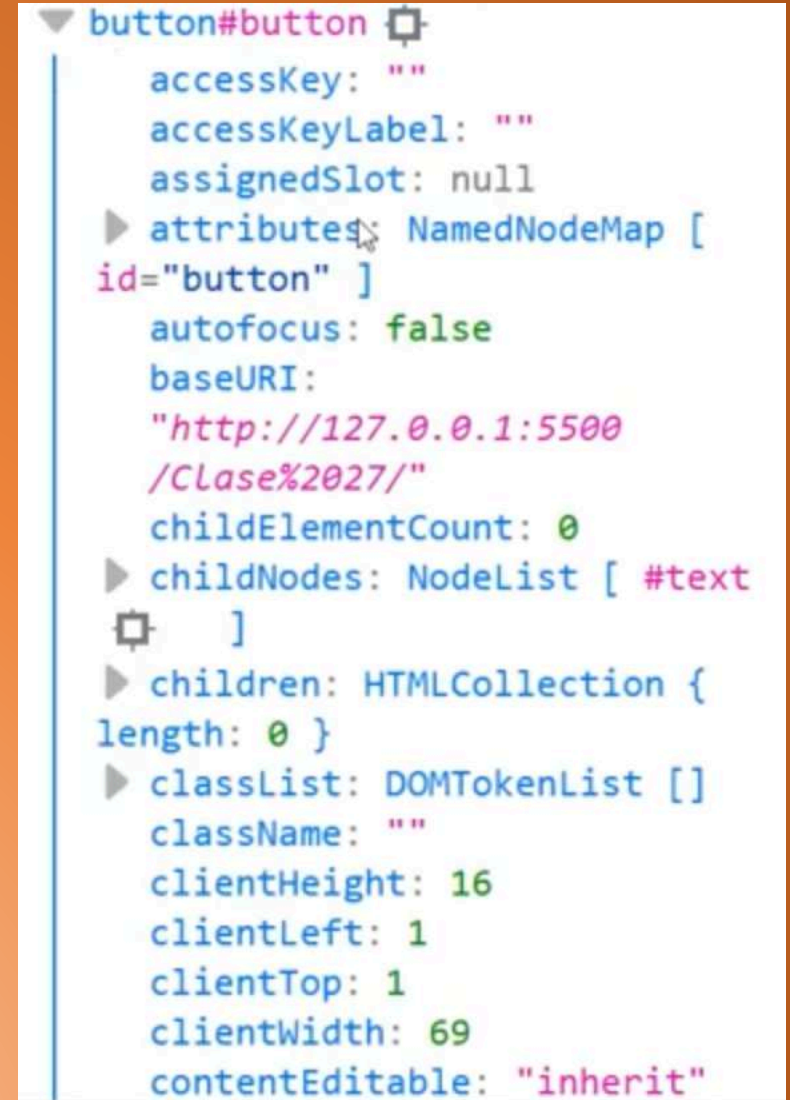
**¿cómo podemos sacar información del evento? Averigua como podemos obtener la tecla concreta que se ha pulsado cada vez.**



# ¿Cómo accedemos a la información del evento?

## Ejemplo de acceso al target del evento:

```
button.addEventListener('click', (e) => {  
  console.log(e.target);  
})
```

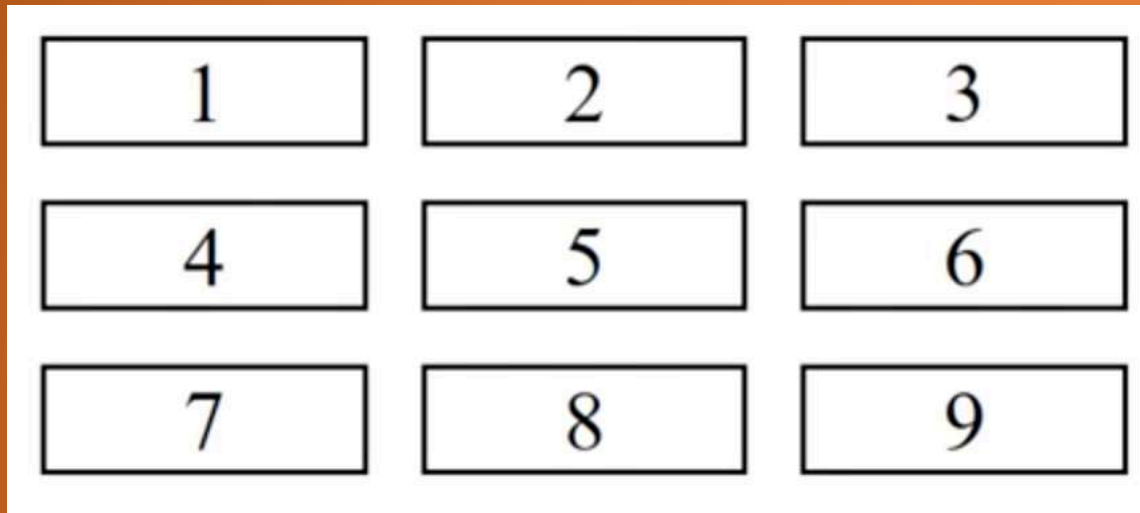


Se puede encontrar mucha información del evento dentro del target  
Por ejemplo:

- Clases que tiene el botón
- Posición del botón
- Si está activado o no
- Si es arrastrable o no
- Tipo de nodo
- Quién es su padre e información del su nodo padre
- Propiedades CSS que tiene el botón
- Tipo de botón que es
- Etc

# Delegación de eventos. Optimización de recursos

¿Dada la siguiente galería, como harías para añadir un evento a cada uno de los div y saber en cual se ha hecho click?



```
<div id="gallery" class="gallery">
  <div class="gallery__item">1</div>
  <div class="gallery__item">2</div>
  <div class="gallery__item">3</div>
  <div class="gallery__item">4</div>
  <div class="gallery__item">5</div>
  <div class="gallery__item">6</div>
  <div class="gallery__item">7</div>
  <div class="gallery__item">8</div>
  <div class="gallery__item">9</div>
</div>
```

# Delegación de eventos. Optimización de recursos

Una opción podría ser localizar la galería con `document.getElementById("gallery")` y hacer un bucle `for` y a cada uno de los botones le añadimos un evento.

Esto consume muchos recursos del navegador.

Para solucionarlo ponemos una escucha al gallery container y localizar en cual de los hijos se ha hecho el click. A continuación podemos ver como hacerlo.


```
const gallery = document.getElementById('gallery')

gallery.addEventListener('click', (e) => {
  console.log(e.target.textContent);
})
```

Esto además de utilizarse en galerías, se puede utilizar también en formularios, cuando hay muchos campos es mucho más óptimo hacerlo de esta forma

# Eventos y Formularios

¿Cómo puedo hacer para que al pulsar un botón de un formulario de tipo **submit** no se recargue la página?



```
form.addEventListener('submit', (e) => {  
  e.preventDefault()  
  console.log('El formulario se ha enviado');  
})
```

Con **preventDefault** conseguimos que no se ejecute el comportamiento por defecto. No sirve solo para formularios sino para cualquier elemento de HTML que tenga un comportamiento determinado.

```
const link = document.getElementById('link')  
  
link.addEventListener('click', (e) => {  
  e.preventDefault()  
})
```



# Eventos y Formularios. Disparar eventos

Podemos disparar eventos sin necesidad de interacción con el usuario.

Para hacerlo pongo el `elemento.eventoadisparar()`. Por ejemplo si tengo una constante `button`, para lanzar el evento `click` sin que el usuario tenga que hacer click en ningún sitio, sería suficiente escribiendo en mi código `button.click()`

