

JavaScript Avanzado

En breve comenzamos... 



Agenda de hoy

- Modelo orientado a eventos
 - qué es un evento
 - cómo manejar eventos en JS
 - tipos de eventos
 - eventos del mouse
 - eventos del teclado
 - eventos de formulario
 - eventos del navegador
 - información de eventos



qué es un evento



qué es un evento

Los eventos en JavaScript es la forma que utilizamos para controlar las diferentes acciones que los usuarios realizan sobre nuestras aplicaciones web.

JS nos brinda mecanismos para detectar cuándo ocurren diferentes eventos, y por lo tanto que podamos definir un código específico para actuar en consecuencia ante los mismos.



qué es un evento

Un ejemplo muy común de eventos, es cuando hacemos clic en un botón, o en un punto de menú, para enviar un formulario o navegar hacia otra sección de la aplicación web.

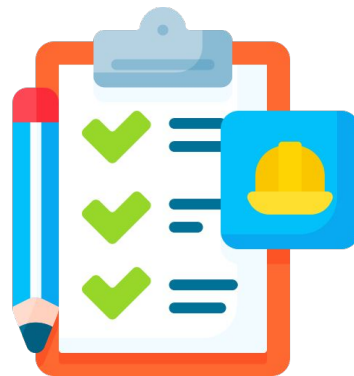
Ante el evento clic, nosotros responderemos con una acción asociada, para que el usuario pueda hacer efectiva su necesidad.



qué es un evento

El listado de eventos en JS son muchos, demasiados. Lo bueno es que la mayoría de estos son comunes a cualquier elemento HTML, por lo tanto es más fácil pensar en cómo aplicarlos.

Vamos a categorizar los eventos para darle un orden efectivo, y que así sean más fáciles de recordar.



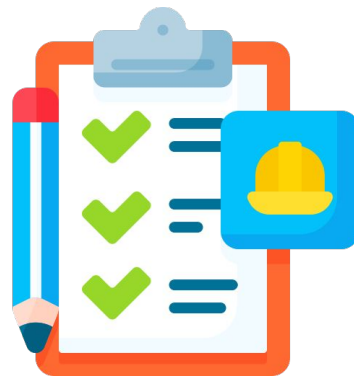
cómo manejar eventos en JS



cómo manejar eventos en JS

Lo primero que debemos identificar es que los eventos se aplican sobre un objeto JS el cual debe estar relacionado con un elemento HTML. Hay excepciones como la de detectar eventos del navegador web, pero profundizaremos estas hacia el final de la clase.

Una vez identificado el evento, asociamos una función al mismo. Así, cuando el evento ocurre, se ejecutará la función asociada.



cómo manejar eventos en JS

Esta forma de asignar una respuesta a un evento, se denomina “*manejar un evento*”; **event handling** en inglés.

Cada función asociada, puede ser una función convencional creada en JS, o puede ser una función anónima, creada y dedicada específicamente para dicho evento.



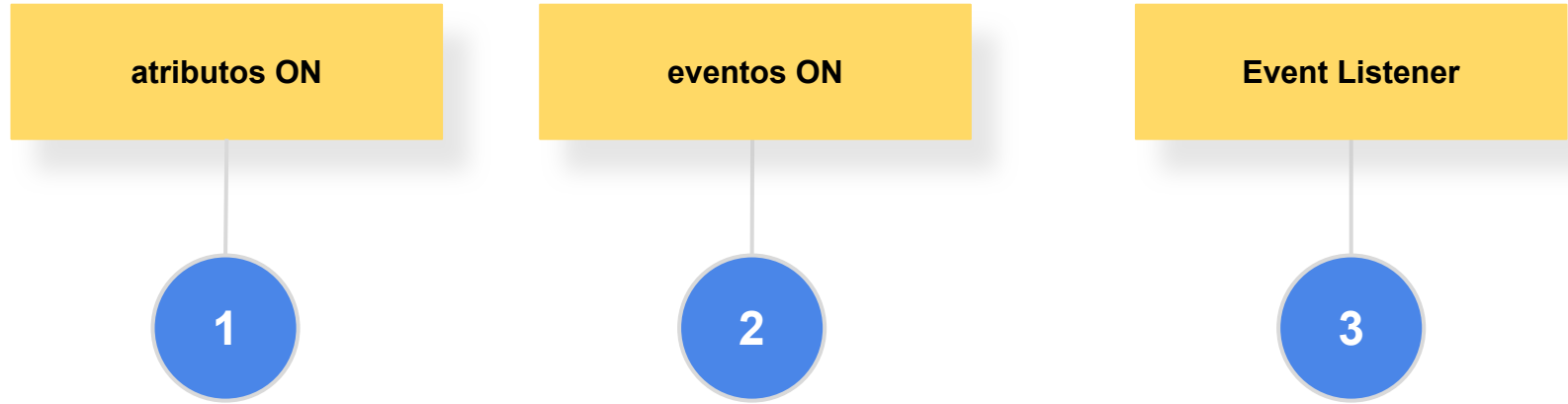
cómo manejar eventos en JS

Dentro del navegador web, los eventos ocurren de forma constante. El motor de JS se ocupa de activar lo que se denomina un escuchador de eventos.

El motor de JS escucha eventos todo el tiempo (**event listener**) y, cuando detecta uno en cuestión, revisa si hay alguna función asociada para ejecutar.



Manejar eventos en JS



cómo manejar eventos en JS

Los atributos on, son aquellos que forman parte de los elementos HTML. Se conocen con este nombre, porque en el nombre de cada evento, se antepone el término **on**.

atributos ON

Estos se definen dentro del elemento HTML donde deben ocurrir, y se les asocia una función la cual se ejecuta ante la ocurrencia de dicho evento.

cómo manejar eventos en JS

Ejemplo de un evento manejado mediante un **atributo on**.

```
atributos on

//JavaScript
function saludar() {
  console.log("Hola mundo!");
}

//HTML
<button class="btn btn-blue white-text" onclick="saludar();">MENSAJE</button>
```

cómo manejar eventos en JS

Estos eventos son manejados desde JavaScript. Una vez definida una constante la cual nos conecta con un elemento HTML mediante el objeto **document**, el evento **on** es accesible como si fuese una propiedad.

eventos ON

También se le asocia una función a dicho **evento on**, la cual es ejecutada cuando este evento ocurre. Como alternativa, se le puede definir una función anónima.

cómo manejar eventos en JS

Ejemplo de **evento on**, invocando una función existente.

```
eventos on

//HTML
<button class="btn btn-blue white-text">MENSAJE</button>

//JavaScript
const button = document.querySelector("button");

button.onClick = saludar; //nombre de la función a ejecutar
```

cómo manejar eventos en JS

Ejemplo de **evento on**, con una función anónima.

```
eventos on

//HTML
<button class="btn btn-blue white-text">MENSAJE</button>

//JavaScript
const button = document.querySelector("button");

button.onClick = function() {
  console.log("Hola mundo!");
}
```


cómo manejar eventos en JS

Los event listener se conforman mediante el uso de un método llamado **addEventListener()**, asociado a la constante que nos conecta con un elemento HTML mediante el objeto **document**.

Event Listener

Este método espera dos parámetros: el primero corresponde al evento en cuestión; el segundo, corresponde a qué función se debe ejecutar.

cómo manejar eventos en JS

Cuando definimos el método **addEventListener()**, el primer parámetro que corresponde al evento a escuchar, no lleva antepuesta la palabra **on**. Solo debemos definir de manera directa el nombre del evento.

Event Listener

Evento o Atributo “on”	addEventListener()
onmousemove	mousemove
onclick	click
ontimechange	timechange

cómo manejar eventos en JS

Ejemplo de manejo de evento utilizando el método **addEventListener()**.

```
Event Listener

//HTML
<button class="btn btn-blue white-text">MENSAJE</button>

//JavaScript
const button = document.querySelector("button");

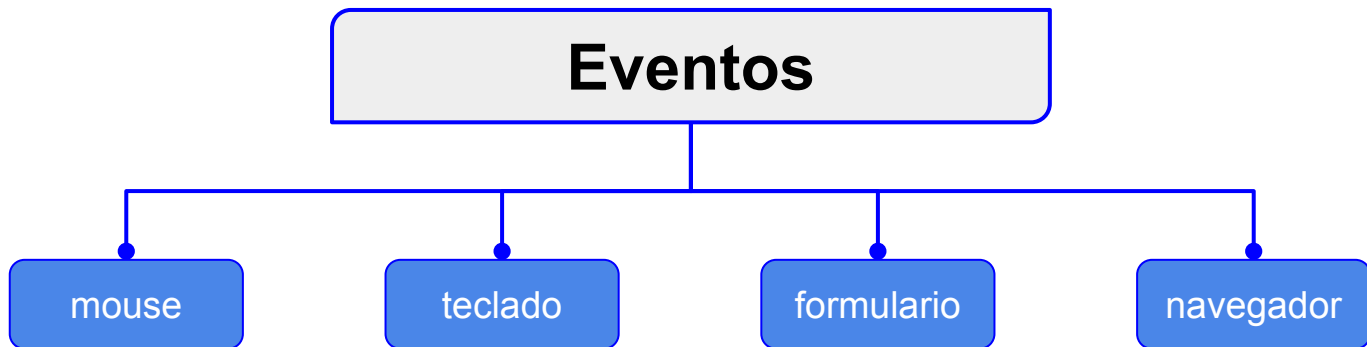
button.addEventListener("click", saludar);
```

tipos de eventos



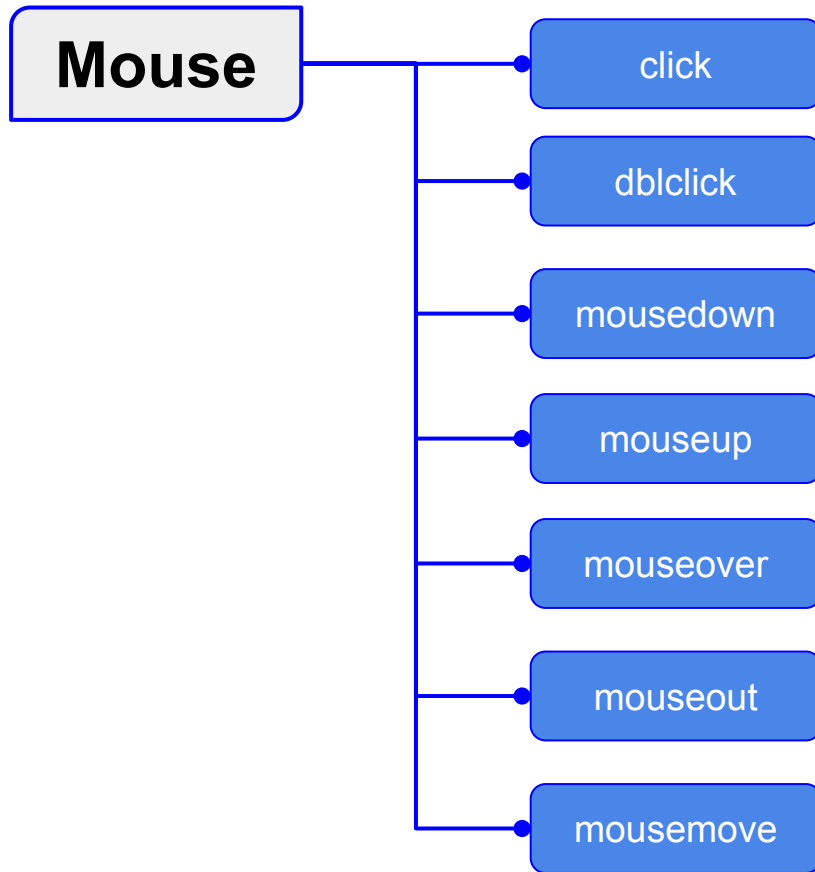
tipos de eventos

Categorizar los diferentes eventos ayuda a ordenar la forma en la cual recordaremos el uso de los mismos. Veamos a continuación una manera más óptima de ordenarlos.



tipos de eventos

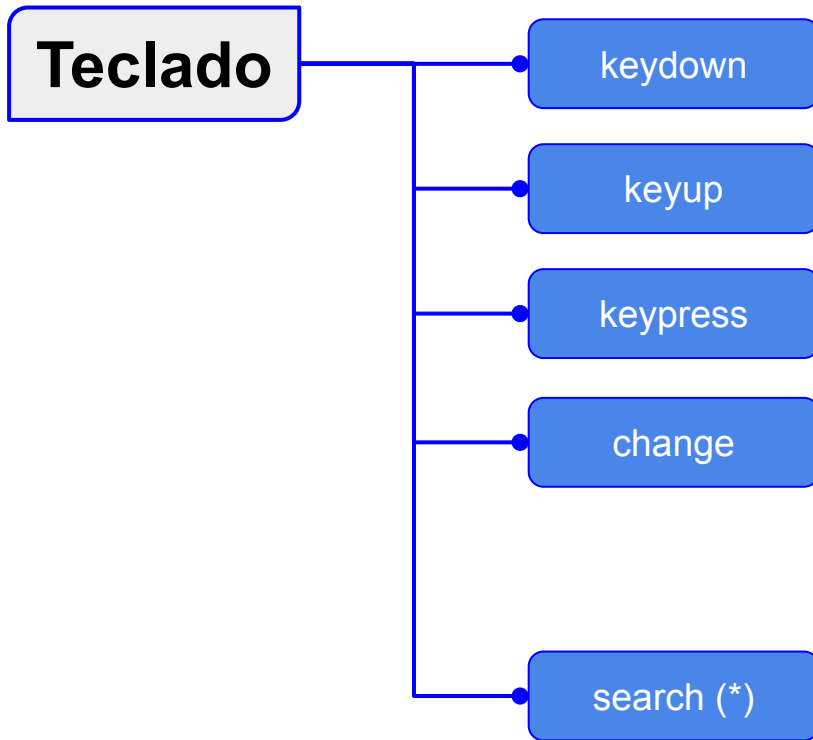
Los eventos del mouse están asociados a todo aquel evento disparado por el puntero del mouse, e incluso por el “*tap*” que realizamos sobre elementos HTML, a través de pantallas táctiles. Veamos entonces algunos de ellos:



tipos de eventos

Los eventos de teclado están regidos por todo lo que refiere a la pulsación de teclas.

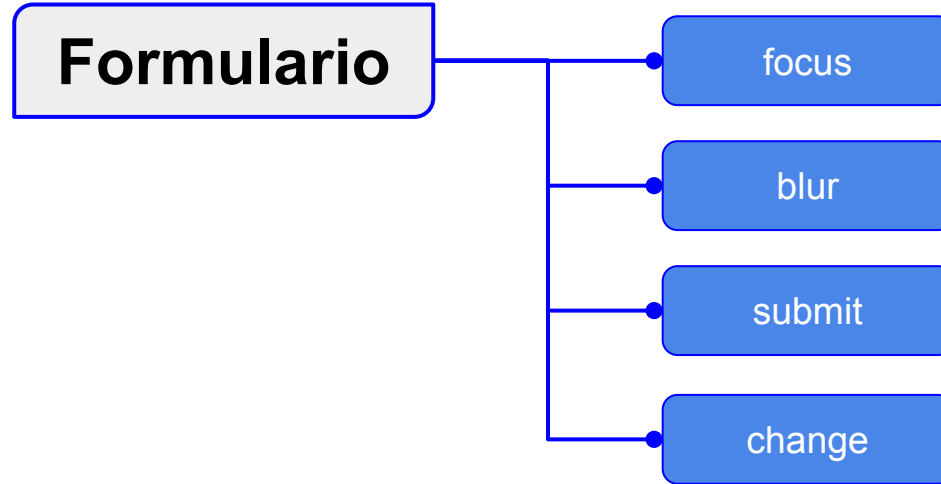
En particular, el evento *change*, aplica a poder detectar cuando se realizó un cambio en el contenido de algún elemento HTML.



(*) responde a detectar el evento equivalente a ENTER en un **input type "search"**, el cual inicia un evento asociado a una búsqueda.

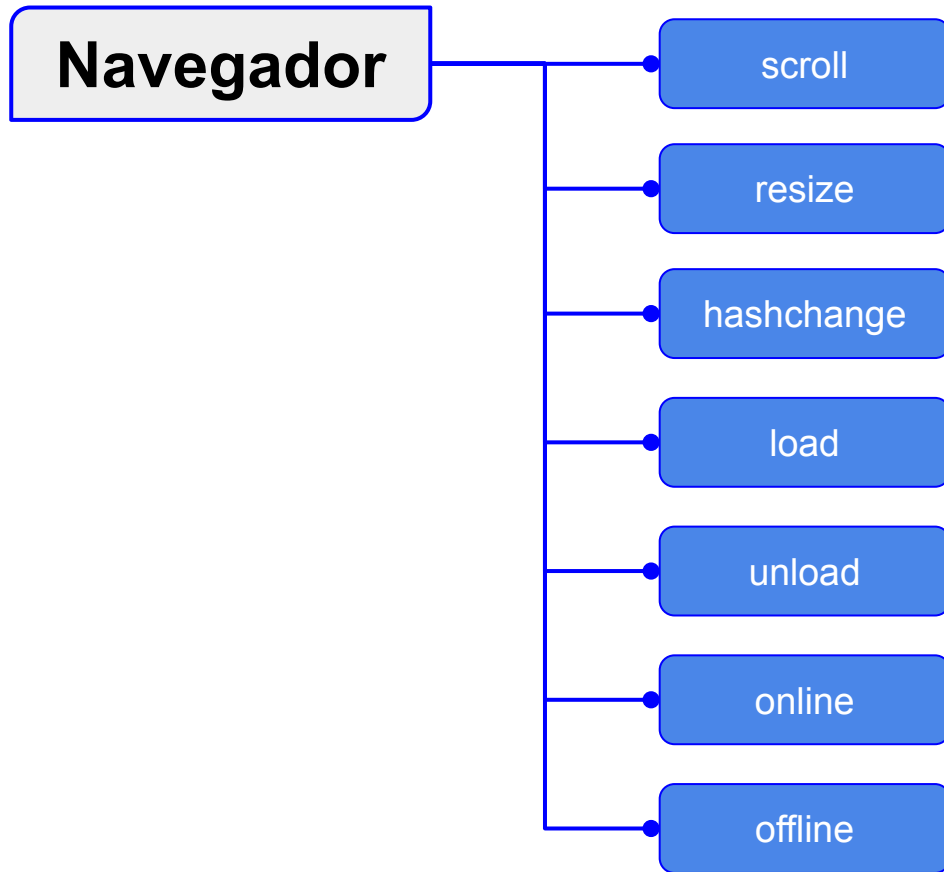
tipos de eventos

Estos eventos engloban la detección de acciones sobre elementos de un formulario web. Se incluye al tag **<form>** como también a los **<input type>** que se incluyan en el mismo.



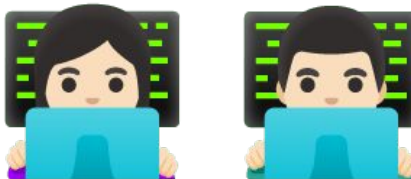
tipos de eventos

Los eventos del navegador ocurren en su mayoría sobre el objeto JS **window**.



tipos de eventos

Pongamos acción a toda esta información. Desarrollaremos a continuación un ejemplo funcional, donde aplicaremos todos los eventos posibles, según la categoría de cada uno.



información del evento



información del evento

En determinadas situaciones, es probable que necesitemos conocer información adicional del evento en cuestión como, por ejemplo, obtener datos del contexto que generó el mismo.

Luego, en base a esto, realizar acciones más concretas.



información del evento

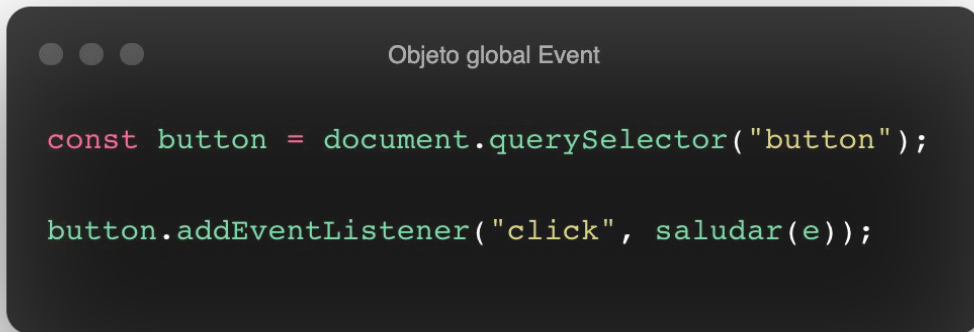
JavaScript cuenta para ello, con un objeto global denominado **Event**, el cual está siempre presente en cualquier parte de nuestra aplicación web donde ocurra un evento.

Su uso es opcional. En el caso de que deseemos incluirlo, lo podemos invocar como primer parámetro dentro de la función asociada al evento en cuestión.



información del evento

Para hacer uso del objeto global **Event**, podemos definir al mismo utilizando la letra **e**, **evt**, **event**, **ev**. Cualquiera de estos términos son los de uso más común que podrán encontrar en ejemplos de código a lo largo de la web.



```
Objeto global Event

const button = document.querySelector("button");

button.addEventListener("click", saludar(e));
```

información del evento

Este objeto acarrea información del contexto. Esto quiere decir que lleva consigo datos de qué elemento HTML u objeto generó el evento en cuestión. Al ser un objeto contenido dentro de Event, podemos navegar por propiedades y métodos asociados al elemento en sí, y aprovechar datos específicos de este.

```
Objeto global Event

//HTML
<input type="text" id="nombre">

//JavaScript
const inputNombre = document.querySelector("input#nombre");

inputNombre.addEventListener("keypress", ()=> detectarTeclaPulsada(e));
```



información del evento

Cuando escribimos en un **input type**, podemos automatizar la tarea de enviar el foco al siguiente campo cuando el usuario pulse **Enter**.

```
Objeto global Event

function detectarTeclaPulsada(e) {
  if (e.key === "Enter") {
    console.log("Has pulsado la tecla Enter.");
    //enviamos el foco al siguiente campo
  }
}
```

La información del contexto en el objeto **Event**, en este caso, corresponde a **KeyboardEvent** (*eventos del teclado*). Dentro de la misma, podemos detectar qué tecla se está pulsando y, en consecuencia, actuar de acuerdo a nuestra necesidad.

información del evento

```
Eventos de Formulario

<form method="POST" action="https://formularios.com/procesarForm">
  <input type="text" name="nombre" placeholder="Ingresa tu nombre">
  <input type="email" name="email" placeholder="Correo electrónico">
  <input type="tel" name="telefono" placeholder="Teléfono de contacto">

  <button type="submit" name="btnEnviar">ENVIAR</button>
</form>
```

Cuando trabajamos con formularios HTML, el mismo formulario posee el control de su contenido. Por defecto, cuando pulsamos Enter, aunque tengamos el foco del cursor en un input type, el formulario será enviado por defecto.

información del evento

Para evitar este comportamiento por defecto, podemos definir el evento **submit**, el cual escucha la acción de “*enviar/procesar el formulario*”.

Pasándole el objeto global Event como parámetro a la acción a este evento, podemos recurrir al método **preventDefault()**. Este método bloquea la acción por defecto de procesar el formulario de datos

```
Eventos de Formulario

const formulario = document.querySelector("form");

formulario.addEventListener("submit", (e) => {
  e.preventDefault();
  //aquí manejamos desde JS qué deseamos hacer
  //con el formulario cuando se pulse el botón
  //ENVIAR del mismo.
});
```

eventos del navegador



eventos del navegador

Los eventos del navegador son varios, pero nos vamos a concentrar en dos eventos específicos: **online** y **offline**.

Tal como sus nombres indican, estos eventos permiten detectar cuando el navegador web pierde conexión a internet, y cuando tiene nuevamente conexión a internet, por lo tanto, podremos actuar en consecuencia.



eventos del navegador

Estos eventos se activan sobre el objeto global window. Podemos, por ejemplo, desactivar el botón **SUBMIT** de un formulario HTML si se pierde conexión a internet, para que el usuario no envíe el Form y se pierdan los datos ingresados.

```
Events del navegador

const btnEnviar = document.querySelector("button type=['submit']");

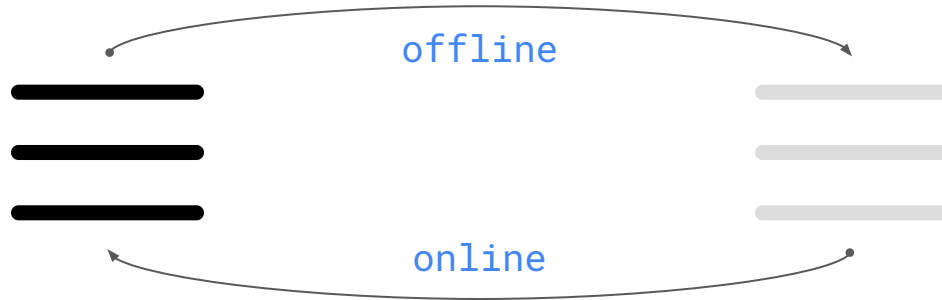
window.addEventListener("online", ()=> btnEnviar.disabled = false);

window.addEventListener("offline", ()=> btnEnviar.disabled = true);
```



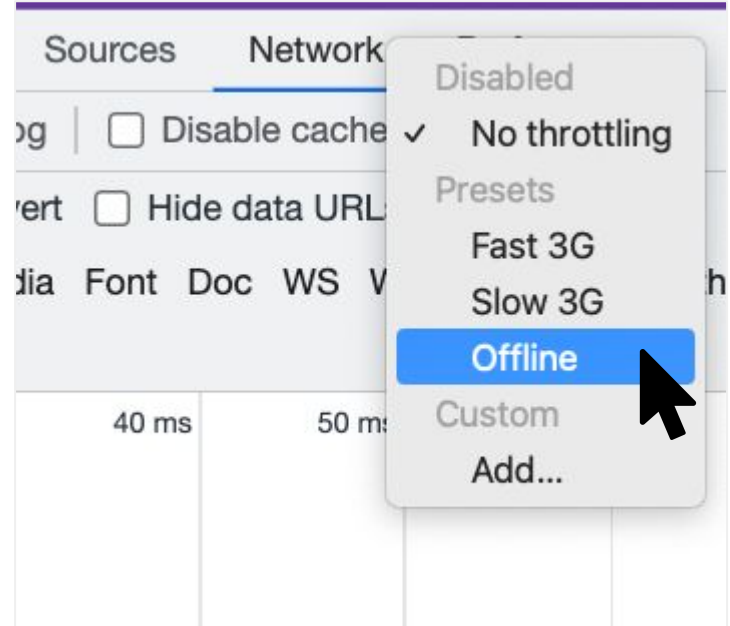
eventos del navegador

De igual forma, tanto en un sitio o aplicación web, podemos desactivar el menú de navegación si el usuario pierde conectividad, así evitamos que aparezca la famosa pantalla de error. Cuando retorne la conectividad, activamos nuevamente el menú.



eventos del navegador

En **DevTools > Network** encontraremos un menú desplegable desde donde podemos simular diferentes velocidades de navegación web ó la pérdida de conectividad, sin tener que desactivar el wifi, desconectar el cable de red, o apagar los datos móviles.





Mención Especial

Tanto el tag **<audio>** como **<video>**, poseen una serie de eventos específicos, los cuales permiten observar la reproducción del contenido, pausarlo, avanzar o retroceder contenido y cambiar el volumen, entre otras tantas opciones más y, con ello, actuar en consecuencia.

Tengamos esto presente siempre, porque ambos elementos HTML son totalmente controlables desde JS, a través de sus diferentes eventos.





Espacio de prácticas





Espacio de prácticas



20 minutos 

Trabajaremos con eventos JS vistos en esta clase

- Recupera
- Utiliza la propiedad **textContent** o **innerText**, para definir un texto nuevo en **título**, y un texto nuevo en **parrafo**
- Cambia la imagen predeterminada, referenciando la variable **imagenColor** en el atributo **src**
- define una constante **parrafos**, y crea en ella un array de elementos HTML que contengan la clase '**fuentes-italica**'
- recorre el array anterior, y agrega la clase '**estilos-parrafo**', a cada elemento HTML. Utiliza para ello **for...of**, la propiedad **classList** y su método **.add()**



Espacio de prácticas



Puesta en común del ejercicio

¡Gracias!

educación IT