



# **Eventos, objeto window, y Timers**

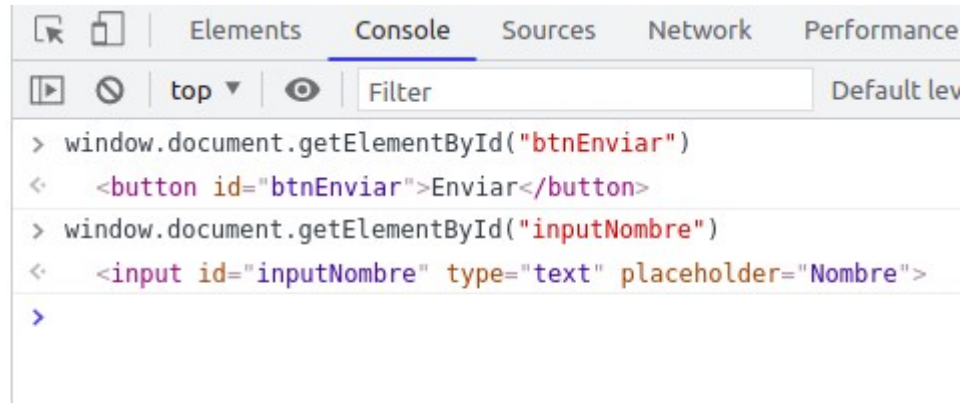
# Evento load

- Este evento se dispara al cargar por completo el documento.
- Es un evento del objeto window
- Al ser evento del objeto window, se puede omitir la declaración hacia dicho objeto
- Ejemplo:

```
addEventListener('load', (evt) => {  
  alert("La página se cargó completamente!");  
})
```

# Objeto window

- Este objeto representa la ventana que contiene un documento
- Es el objeto principal en la jerarquía y contiene las propiedades y métodos para controlar la ventana del navegador.
- document depende de window.
- Con el podemos controlar toda la ventana, modificar o obtener propiedades como el tamaño, el estado, las ventanas, etc.

```
> window.document.getElementById("btnEnviar")
< <button id="btnEnviar">Enviar</button>
> window.document.getElementById("inputNombre")
< <input id="inputNombre" type="text" placeholder="Nombre">
>
```

# alert()

- La función alert que venimos usando desde la primer clase, no es nada mas ni nada menos que un método del objeto window!

```
> window.alert('Hola!')  
> |
```

- En general cuando invoquemos métodos del objeto alert, no será necesario escribir `window.[metodo]`
- Lo mismo sucede con los eventos, como vimos recién

# Remover un evento

- Usaremos el método `removeEventListener("nombre_evento")`

```
const boton = document.getElementById('btnEnviar');

boton.addEventListener('click', buttonEventHandler);

function buttonEventHandler(evt) {
  evt.preventDefault();
  alert("No me volveras a hacer click :)");

  // eliminamos el evento:
  boton.removeEventListener('click', buttonEventHandler);
}
```

- En este ejemplo, removemos el evento click, una vez se haya disparado. Es decir, el evento se ejecutará una sola vez.

# Otra forma de ejecutar un evento una única vez

- Le pasaremos al `addEventListener` un tercer parámetro, indicándole que queremos que se ejecute una sola vez.

```
const boton = document.getElementById('btnEnviar');

boton.addEventListener('click', (evt) => {
  evt.preventDefault();
  console.log("Este mensaje no se volverá a mostrar :");
}, { once: true });
```

- El tercer parámetro será el objeto: `{ once: true }`
- (once = Una vez)

# Timers en JavaScript

- Los Timers son funciones que permiten establecer la ejecución de funciones en determinados momentos del tiempo.
- Dichos Timers son funciones predefinidas del objeto window
- `window.setInterval()`
- `window.setTimeout()`

# setInterval()

- Ejecuta una función de forma repetitiva cada cierto tiempo.
- El tiempo se expresa en milisegundos
- Ejecutamos unaFuncion cada 5 segundos:
- `setInterval(unaFuncion, 5000);`



# setTimeout()

- A diferencia de setInterval, este método establece un temporizador que ejecuta una función después de que transcurre un tiempo establecido.
- Es decir, ejecuta la función una sola vez, cuando pasen un tiempo específico de milisegundos

```
addEventListener('load', (e) => {  
  setTimeout(mostrarMensaje, 5000);  
})  
  
const mostrarMensaje = () => alert("Bienvenido/a");
```

# Eliminar los timers

- Usaremos las funciones `clearInterval()` o `clearTimeout()`
- Reciben como parámetro el timer

```
let timer = setTimeout(mostrarMensaje, 5000);

function detenerTimer() {
  clearInterval(timer);
}
```

# Element.innerHTML

- Con esta propiedad obtenemos o establecemos el contenido de un elemento HTML
- Es una alternativa al appendChild()
- Ejemplo:

```
const div = document.getElementById("div_datos");

div.innerHTML = `
  <h2>Datos del Usuario:</h2>
  <p>Nombre: ${usuario.nombre}</p>
  <p>Apellido: ${usuario.apellido}</p>
  <p>Fecha Ingreso: ${usuario.fechaIng}</p>`;
```