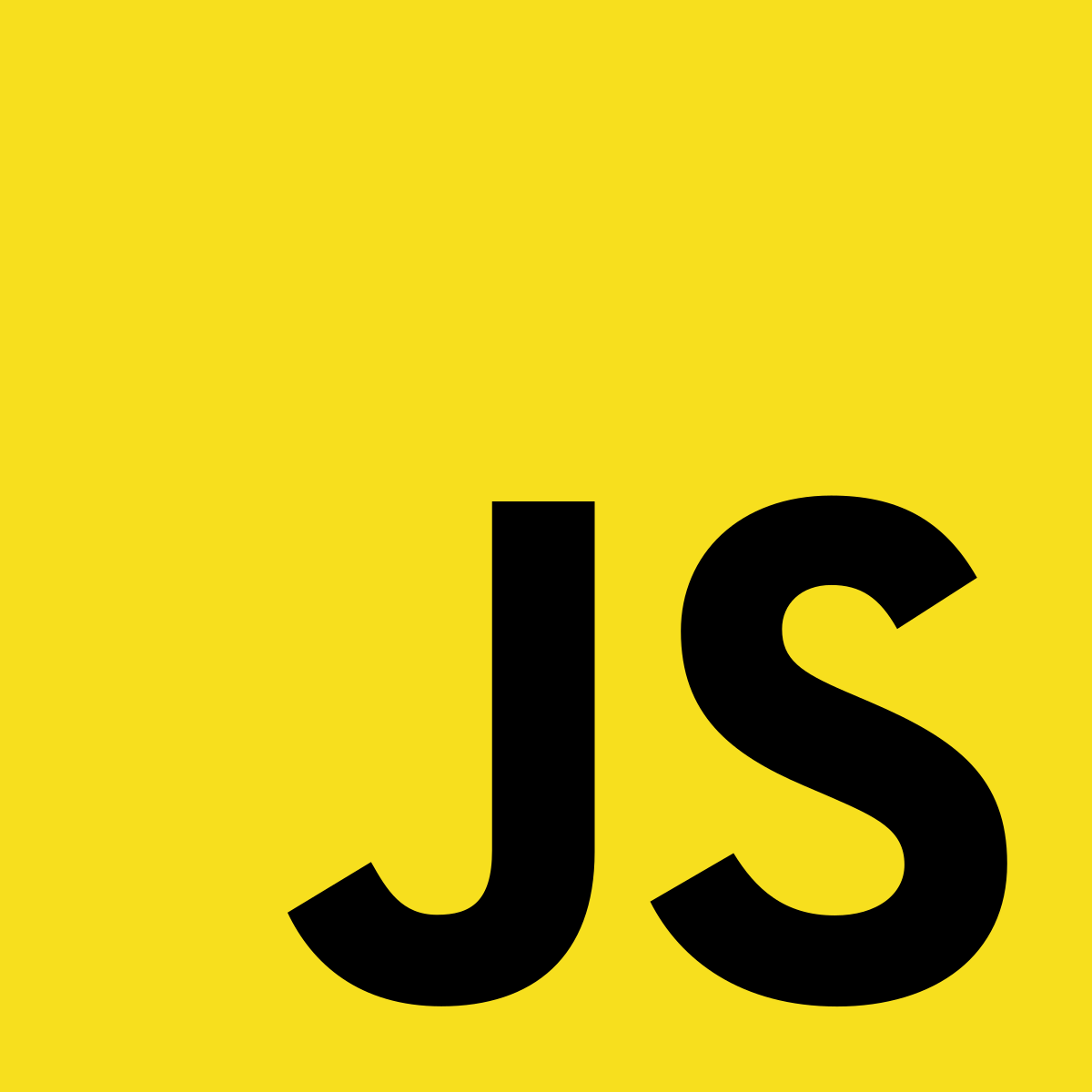
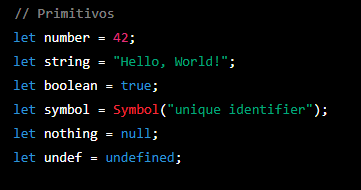
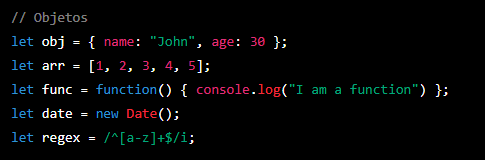
****

**A P U N T E S**

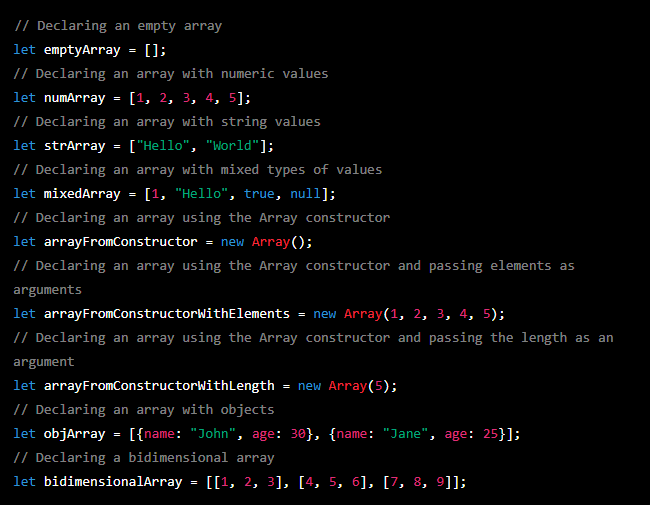
**PRIMITIVE DATA TYPES**



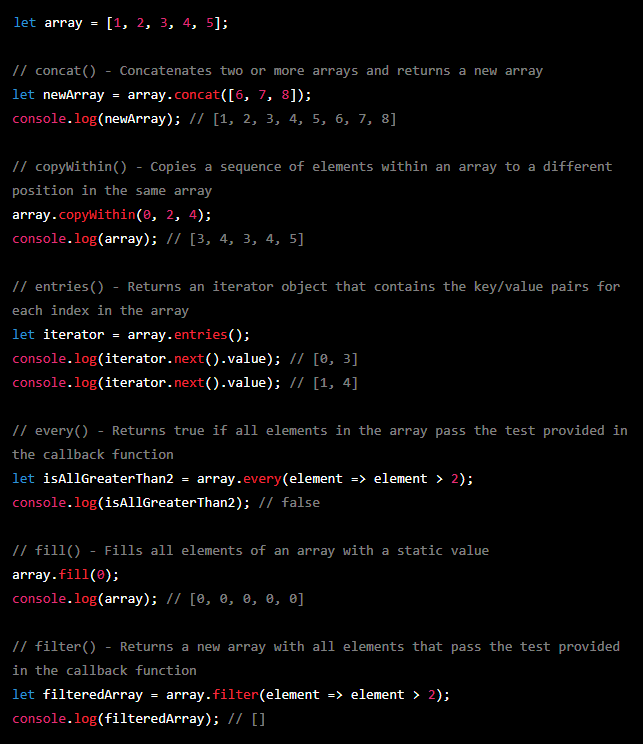
**OBJECT DATA TYPES**

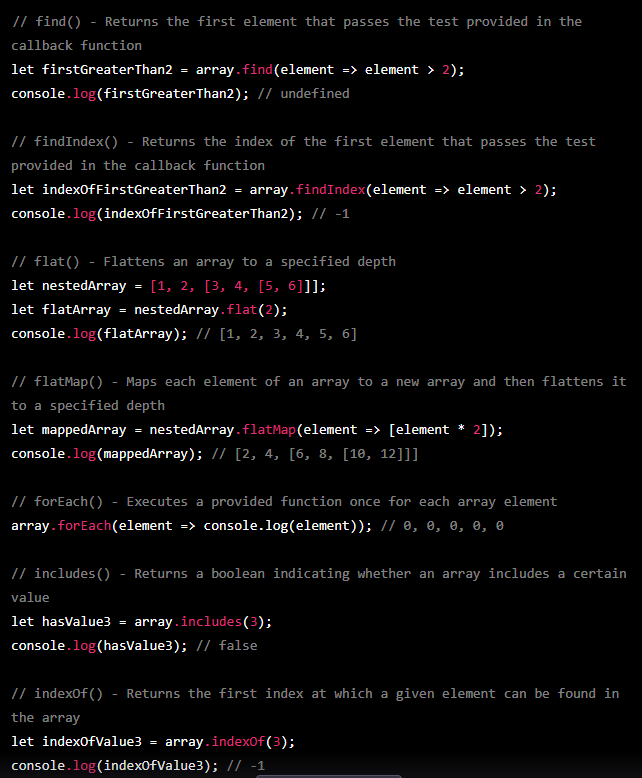


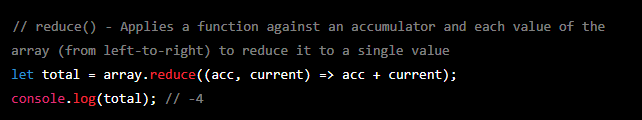
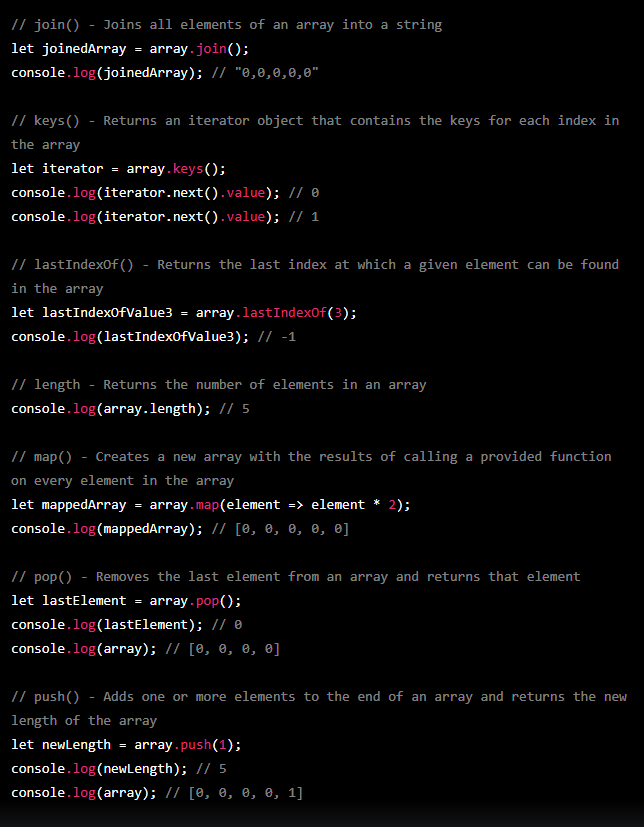
**CREACION DE ARRAYS**

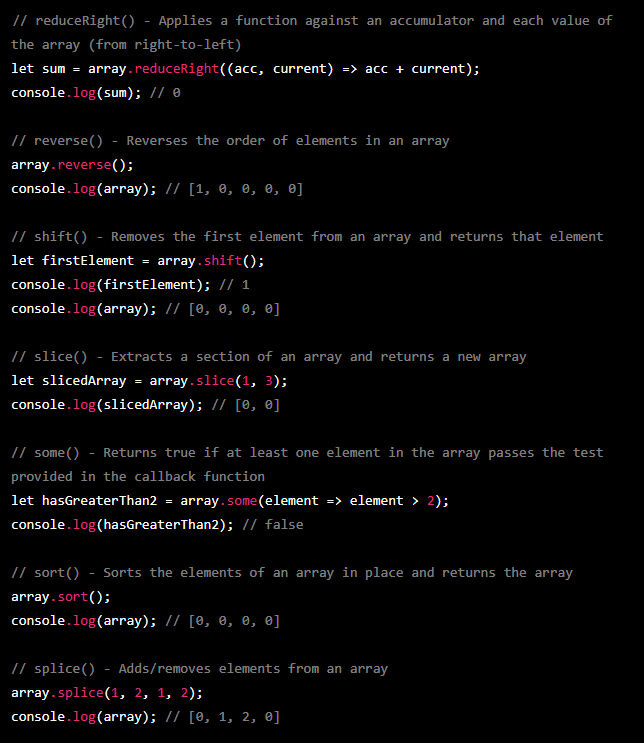


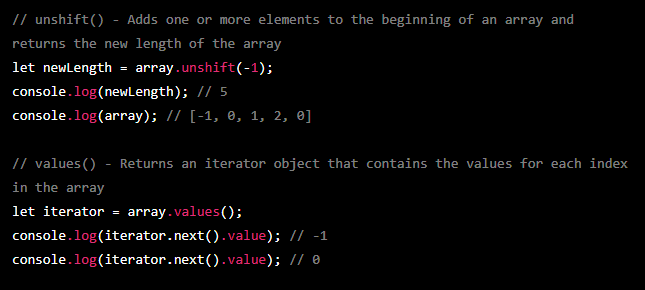
**ARRAY METHODS**

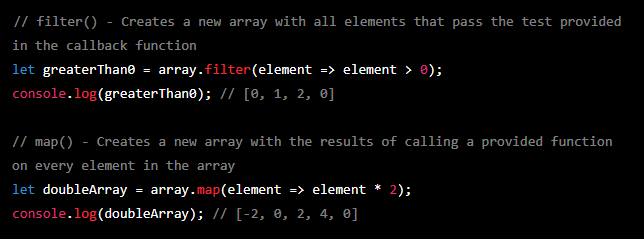
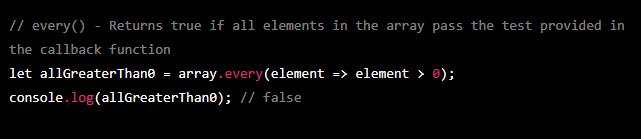
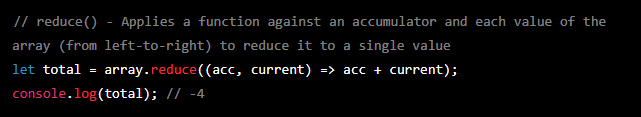




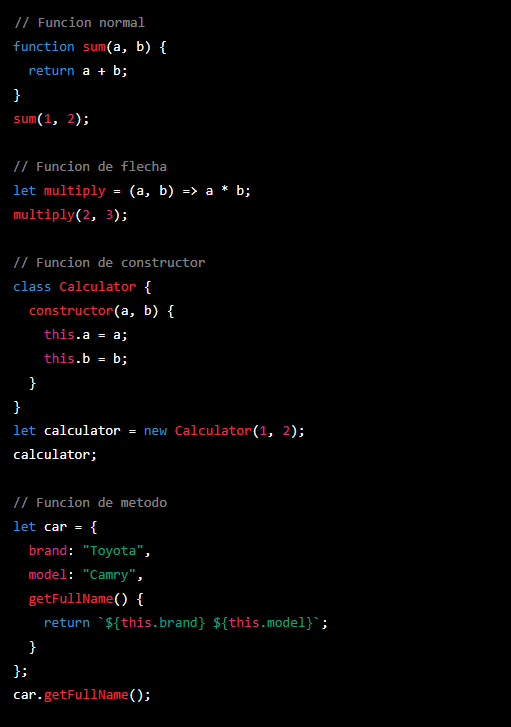
 

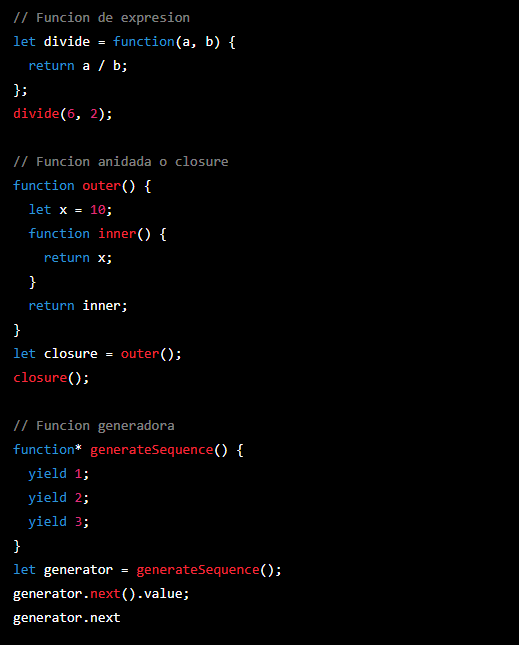




**FUNCIONES**



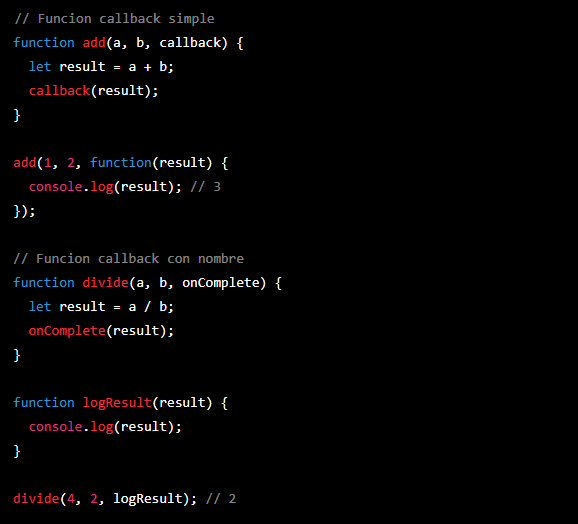


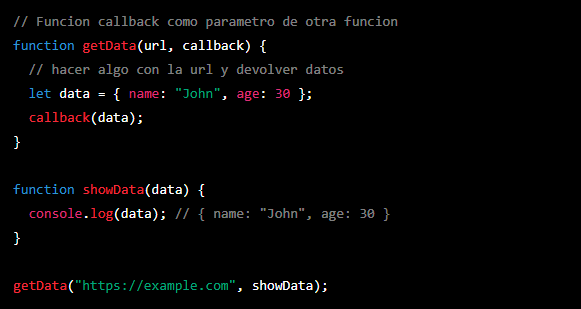
**FUNCIONES CALL-BACK**

Una función Call-Back es una función que se pasa a otra función como un argumento y es invocada dentro de la función externa para completar alguna tarea.

La función Call-Back se ejecuta cuando se produce un evento específico o cuando se completa una tarea asíncrona.

Es útil para realizar operaciones asíncronas de manera ordenada y organizada en JavaScript.

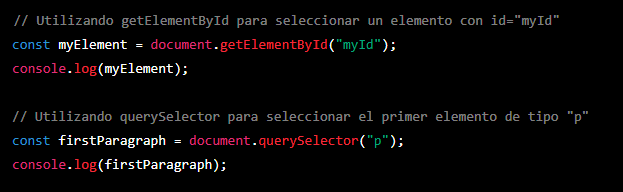




**DOM – SELECTORS (SINGLE ELEMENTS)**

Los selectores de elementos únicos del DOM son utilizados para seleccionar un único elemento en una página web.

Es importante tener en cuenta que estos selectores solo seleccionarán el primer elemento que coincida con la consulta en caso de haber varios elementos con el mismo nombre.



**DOM – SELECTORS (MULTIPLE ELEMENTS)**

**getElementsByTagName()** selecciona todos los elementos con un nombre de etiqueta específico.

  
**getElementsByClassName()** selecciona todos los elementos con un nombre de clase específico.

  
**getElementsByName()** selecciona todos los elementos con un atributo "name" específico.

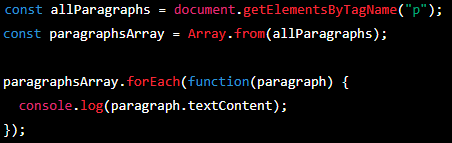
  
**querySelectorAll()** selecciona todos los elementos que coinciden con una consulta de CSS.

Es importante tener en cuenta que estos selectores devuelven una lista de elementos, no un solo elemento.

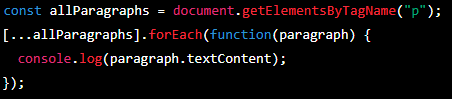
También es importante tener en cuenta que todos los selectores de elementos múltiples **devuelven un objeto HTMLCollection**, no un array, por lo que **no se pueden usar** todas las funciones de un array como **forEach() o map().**

Sin embargo, es posible **convertir los objetos HTMLCollection en un array** utilizando **from() o el operador spread "[...]"**, y luego utilizar forEach() en el array.

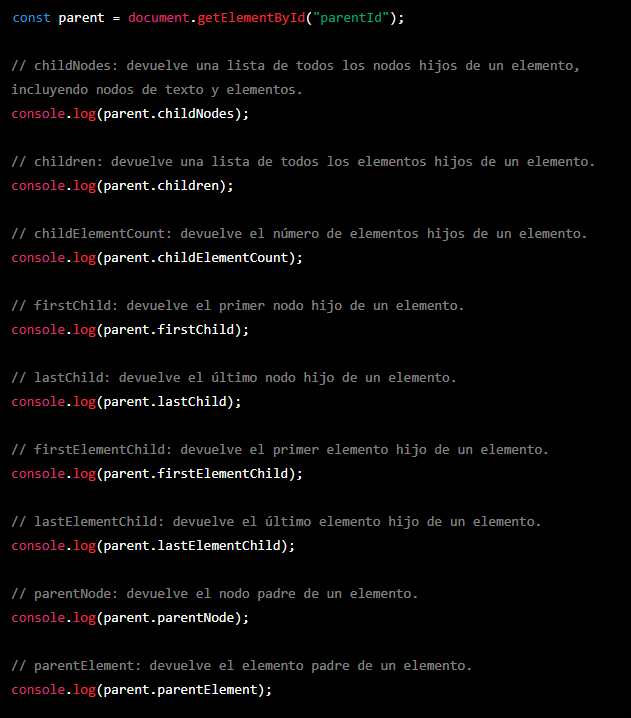
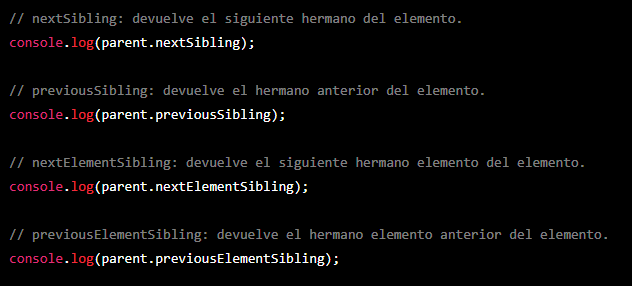
**· from()**



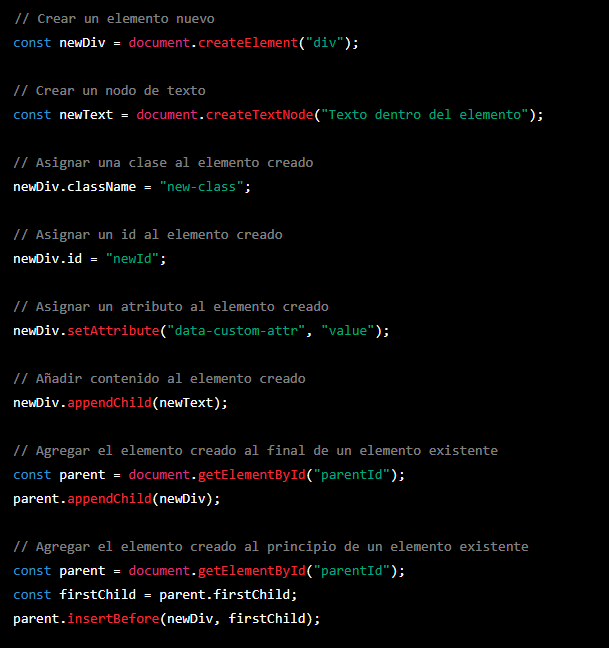
**· Operador spread […]**



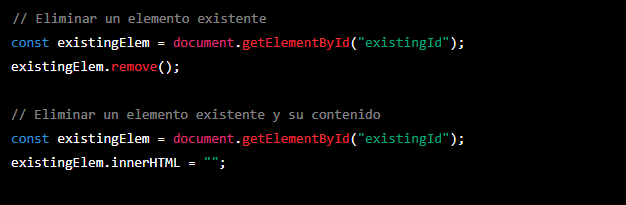
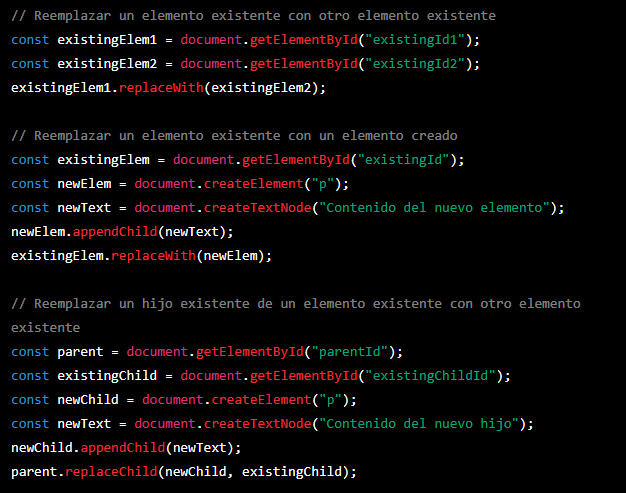
**DOM – TRAVERSING**



**DOM – CREATING ELEMENTS**

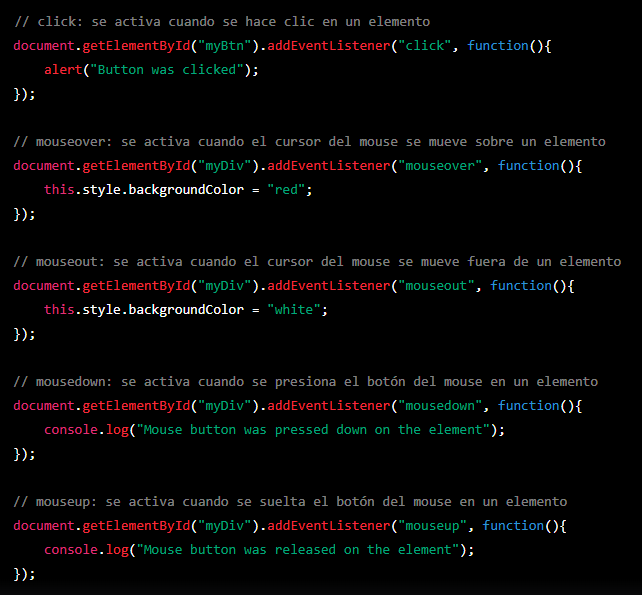


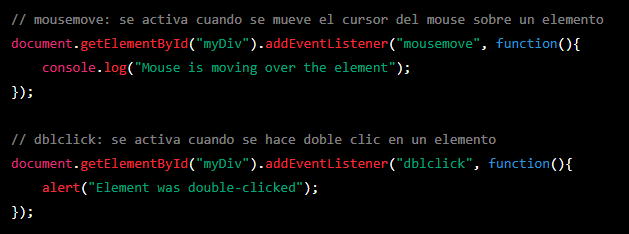
**DOM – REMOVING & REPLACE ELEMENTS**



**EVENTOS – MOUSE**

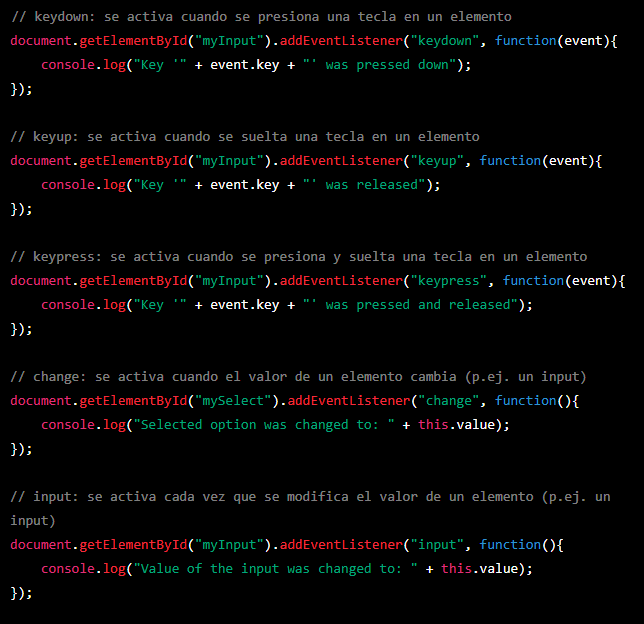
Eventos que ocurren cuando se interactúa con el cursor del mouse, como hacer clic, mover el cursor, presionar y soltar botones, etc.





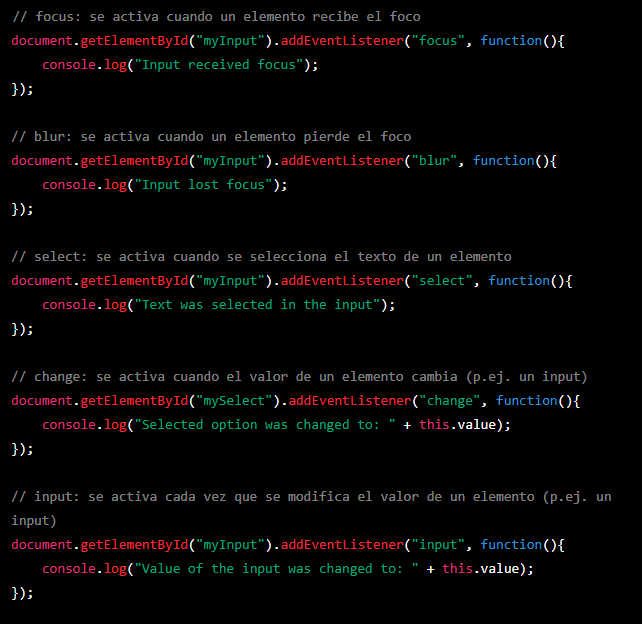
**EVENTOS – KEYBOARD**

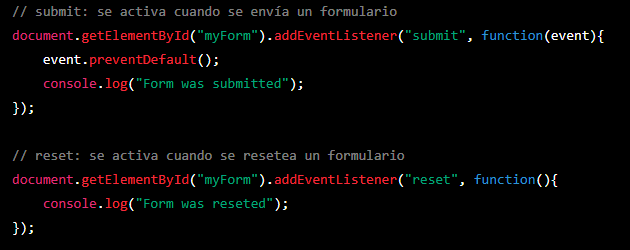
Eventos que ocurren cuando se interactúa con el teclado, como presionar y soltar teclas, ingresar texto, etc.



**EVENTOS – FORMULARIO**

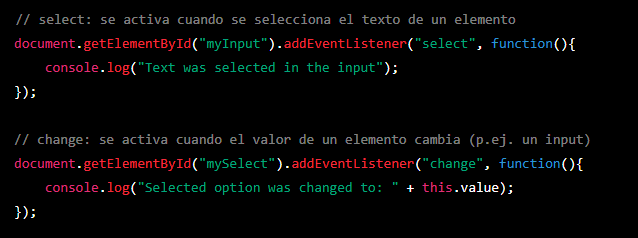
Eventos que ocurren cuando se interactúa con elementos de formulario, como botones, inputs, etc.





**EVENTOS – SELECCIÓN**

Eventos que ocurren cuando se selecciona texto o elementos en una página.

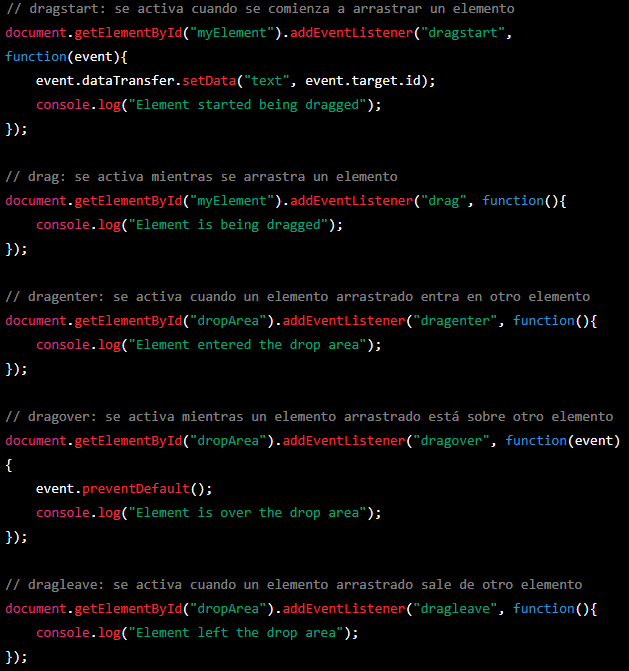
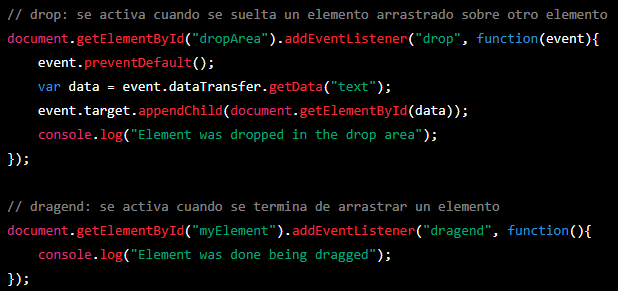


El evento "select" se activa cuando el usuario selecciona el texto de un elemento (como un input o un textarea).

El evento "change" se activa cuando el usuario cambia el valor de un elemento (como un select o un input con tipo "radio" o "checkbox").

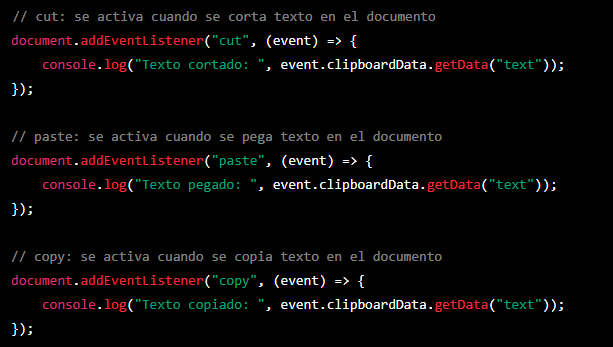
**EVENTOS – DRAG & DROP**

Eventos que ocurren cuando se arrastra y suelta elementos en una página.



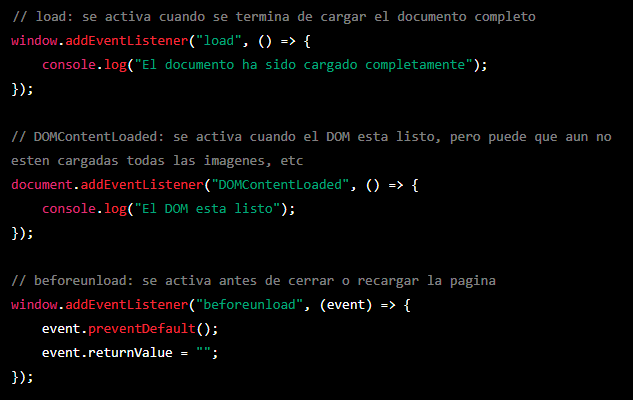
**EVENTOS – COPY & PASTE**

Eventos que ocurren cuando se realiza alguna acción de cortar, copiar y pegar en una página



**EVENTOS – CARGA**

Eventos que ocurren cuando se carga una página o un recurso, como imágenes, scripts, etc.



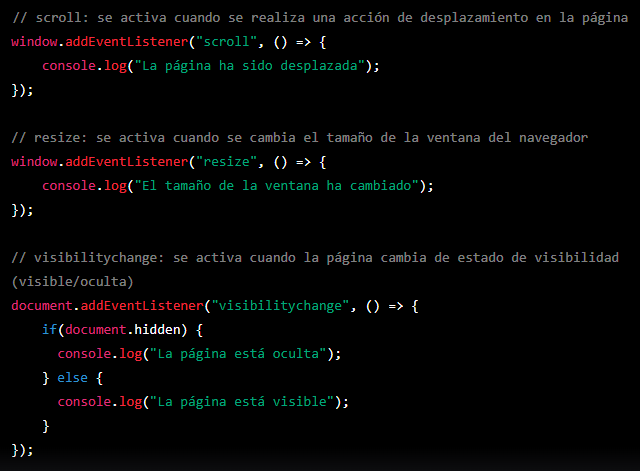
El evento "load" se activa cuando se ha terminado de cargar completamente el documento.

El evento "DOMContentLoaded" se activa cuando el DOM está listo, pero puede que aún no estén cargadas todas las imágenes, etc.

El evento "beforeunload" se activa antes de cerrar o recargar la página, en este caso se esta utilizando event.preventDefault() y event.returnValue = "" para evitar el comportamiento por defecto de mostrar una alerta de confirmación al salir de la página.

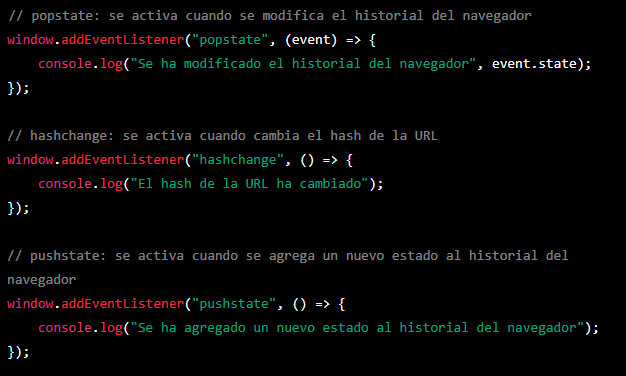
**EVENTOS – VISTA**

Eventos que ocurren cuando se interactúa con la ventana del navegador, como cambiar el tamaño, desplazarse, etc.



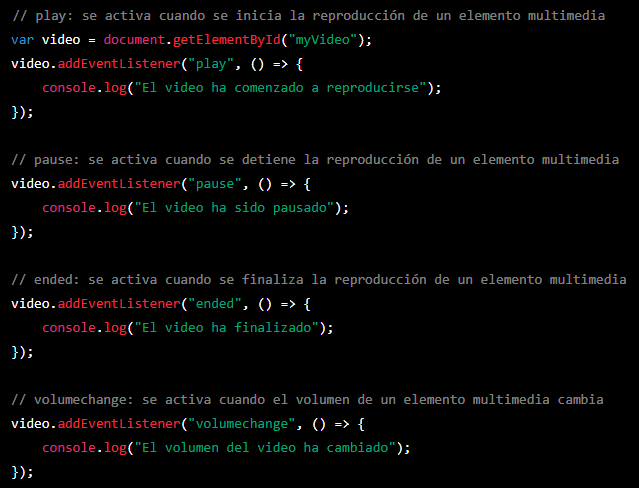
**EVENTOS – NAVEGACIÓN**

Eventos relacionados con la navegación por la página, como hacer clic en enlaces, cambios en la URL, historial, etc.



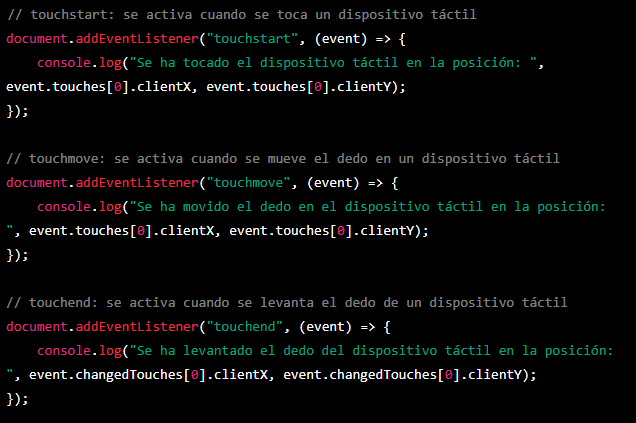
**EVENTOS – MULTIMEDIA**

Eventos relacionados con los medios multimedia, como reproducción de audio y video, cambios en el volumen, etc.



**EVENTOS – DISPOSITIVOS TÁCTILES**

Eventos que ocurren cuando se interactúa con un dispositivo táctil, como tocar y deslizar en una pantalla táctil.



**PROMISES**

Las promesas en JavaScript son objetos que representan la eventual terminación o el fallo de una operación asíncrona.

Una promesa tiene tres estados posibles: pendiente, cumplida y rechazada.

Inicialmente, una promesa se encuentra en un estado pendiente, pero eventualmente transiciona a un estado cumplido o rechazado.

Una vez que se ha cumplido o rechazado, no puede volver a un estado pendiente.

Las promesas tienen dos métodos principales: .then() y .catch().

El método .then() se utiliza para especificar qué acción se debe realizar cuando una promesa se cumple.

El método .catch() se utiliza para especificar qué acción se debe realizar cuando una promesa se rechaza.

Ambas funciones toman como parámetros las funciones que se deben ejecutar en caso de cumplimiento o fallo.

