Introducción JavaScript 

Objetivos

Aprender los conceptos básicos de JavaScript

Conocer las variables y sus tipos

Porque no usar var

Buenas prácticas en el uso de variables

Que son las funciones

Que es la coerción

Manipular el DOM mediante eventos

¿Qué es JavaScript?

JavaScript es un lenguaje de programación muy popular que se utiliza para crear interactividad en sitios web y aplicaciones. Es una herramienta fundamental en el desarrollo web y te permite hacer que tus páginas sean más dinámicas y atractivas para los usuarios.

Algunas características de JavaScript son:

Interacción con HTML y CSS:
Puedes usar JavaScript para manipular el contenido HTML y el estilo CSS de una página web.

Eventos:
JavaScript permite reaccionar a eventos del usuario, como hacer clic en un botón, desplazarse por la página, ingresar datos en un formulario, entre otros. Cuando ocurre un evento, puedes definir qué acción deseas que realice el sitio web.

JavaScript es un lenguaje interpretado:
por lo que no necesita compilarse para ejecutarse. Como otros lenguajes como java o c++. En cambio, el código JavaScript se ejecuta directamente línea por línea en el navegador web o en un entorno que soporte la interpretación de JavaScript, como Node.js para el lado del servidor.

¿Qué son las variables?

Las variables son contenedores que se utilizan para almacenar datos. Estos datos pueden ser números, texto, valores booleanos (verdadero o falso) y otros tipos de información. En JavaScript, declaramos variables usando la palabra clave var, let o const. La diferencia entre var, let y const es el alcance de la variable.

Las variables son dinámicas, lo que significa que puedes cambiar el tipo de dato almacenado en una variable en cualquier momento

Ejemplo de cambios en una variable:

```
var a = 10;
console.log(a); <-- 10
a = "hola";
console.log(a); <-- misma variable cambiamos el dato resultado "hola"
```

Las variables se almacenan en la memoria del navegador o del entorno de ejecución en el que se está ejecutando el código. Cuando declaras una variable y le asignas un valor, se reserva un espacio en la memoria para almacenar ese valor.

Tipos de variables

Variables globales:
Si declaras una variable fuera de cualquier función o bloque de código, se convierte en una variable global. Estas variables se almacenan en el objeto global del entorno de ejecución. y pueden ser accedidas desde cualquier parte del código y permanecen en la memoria durante toda la vida de la página web o la ejecución del programa.

Variables locales:
Si declaras una variable dentro de una función o un bloque de código, se convierte en una variable local. Estas variables se almacenan en la pila de llamadas, que es una parte de la memoria reservada para gestionar las funciones en ejecución. Las variables locales solo son accesibles dentro del ámbito de la función o el bloque donde se declaran y se eliminan de la memoria una vez que la función termina de ejecutarse.

Ejemplo de variables globales y locales:

```
var globalVariable = 10; Variable global

function exampleFunction() {
  var localVariable = 20; Variable local dentro de la función.
  console.log("global-->", globalVariable); Acceso a la variable global
                                            desde dentro de la función.
  console.log("local-->", localVariable); Acceso a la variable local desde
                                          dentro de la función
}

exampleFunction();
console.log("global afuera-->>", globalVariable); Acceso a la variable global
                                                  fuera de la función
// console.log("local afuera-->>>", localVariable); Error, la variable local
                                                    no está definida fuera
                                                    de la función.
```

Uso de let y const en JavaScript moderno...!!!

En JavaScript moderno se recomienda utilizar "let" y "const" en lugar de "var" para declarar variables.

let:
Se utiliza para declarar variables que pueden cambiar su valor a lo largo del tiempo. Las variables declaradas con "let" tienen un alcance limitado al bloque de código (como una función o un bloque de declaración) donde se definen.

const:
Se utiliza para declarar constantes, es decir, variables cuyo valor no puede cambiar después de haber sido inicializado. Las variables declaradas con "const" también tienen un alcance de bloque.

Ejemplo de let const y sus alcances:

```
let x = 10;
console.log("x antes de cambiar-->", x);
x = 20; Cambiar el valor de la variable "x"
console.log("x despues de cambiar-->", x);

if (true) {
  let y = 30; Variable "y" solo disponible dentro de este bloque
  console.log(y); Muestra 30
}

console.log(y); Error, "y" no está definida fuera del bloque if

const price = 3.1416;
// price = 4;  Error, no se puede cambiar el valor de una constante

if (true) {
  const nombre = "Juan"; Constante "nombre" solo disponible dentro
                           de este bloque
  console.log(nombre); Muestra "Juan"
}

// console.log(nombre); Error, "nombre" no está definida fuera
                        del bloque if
```

¿Por qué no usar var?

una de las tantas razones para no usar usar var:

Reasignación y bloqueo:
Las variables declaradas con "var" pueden ser reasignadas en cualquier momento, incluso después de su declaración inicial. Esto puede dificultar el seguimiento de los cambios en el valor de una variable a lo largo del tiempo. En contraste, "const" se utiliza para declarar constantes que no pueden ser reasignadas una vez que se les ha asignado un valor inicial. Esto hace que el código sea más seguro y fácil de entender, ya que se puede confiar en que el valor de una constante no cambiará accidentalmente.

¿Cuál es el estándar para declarar variables en JavaScript?

La convención más común para nombrar variables en JavaScript es el estilo CamelCase. En este estilo, el nombre de la variable comienza con una letra minúscula, y cada palabra subsiguiente comienza con una letra mayúscula, sin espacios ni guiones y el idioma debe ser ingles.

Ejemplo de camelCase:

```
ejemplo de camelCase

let nombreCompleto = "Juan Perez";
let edadUsuario = 30;

ejemplo declaracion de variable descriptiva

let fullName = "Juan Perez";
let userAge = 30;
```

Buenas prácticas al declarar variables en JavaScript

ignificado claro:
Es importante que los nombres de las variables sean descriptivos y reflejen su propósito o contenido. Evita utilizar nombres ambiguos o abreviaciones confusas. Un buen nombre de variable debe ser autoexplicativo para que otros programadores puedan entender rápidamente su función.

Empiezan con letras o guión bajo:
Los nombres de variables deben comenzar con una letra (mayúscula o minúscula) o un guión bajo. No deben comenzar con números o caracteres especiales.

Evitar palabras reservadas:
No utilices palabras reservadas de JavaScript (como "if", "else", "while", "for", "function", etc.) como nombres de variables, ya que pueden causar errores en el código.

Algunas palabras reservadas

abstract	else	int	switch
arguments	export	interface	synchronized
await	extends	let	this
boolean	false	long	throw
break	final	native	throws
byte	finally	new	transient
case	float	null	true
catch	for	package	try
char	function	private	typeof
class	goto	protected	var
const	if	public	void
continue	implements	return	volatile
debugger	import	short	while
default	in	static	with
delete	instanceof	super	yield
do			
double			

¿Dónde se agrega la etiqueta script?

Se puede agregar dentro del head, o en el body como cualquier etiqueta de HTML

Buenas prácticas

Ubicación del enlace:
Siempre que sea posible, coloca las etiquetas "script" cerca del final del cuerpo (body) en lugar de la sección head. Esto asegura que el contenido principal de la página se cargue

primero antes de ejecutar el código JavaScript, lo que puede mejorar la velocidad de carga percibida por los usuarios.

Evitar código JavaScript inline:
Siempre que sea posible, evita agregar código JavaScript directamente dentro del HTML utilizando la etiqueta script incorporada. Esto puede dificultar el mantenimiento del código y afectar la legibilidad de tu HTML. Es preferible tener el código JavaScript separado en archivos externos.

¿Que son las funciones?

Son herramientas que podemos utilizar para realizar diversos tipos de acciones. Sabemos que son funciones o métodos porque vienen acompañados de paréntesis. Los elementos que agreguemos dentro de los paréntesis reciben el nombre de argumentos.

Algunas funciones más usadas son:

console.log():
Esta función se utiliza para imprimir mensajes o valores en la consola del navegador o del entorno de ejecución. Es una herramienta útil para depurar y ver el contenido de variables y objetos durante el desarrollo.

alert():
Muestra un mensaje emergente con el contenido proporcionado. Es útil para mostrar mensajes informativos o solicitar la interacción del usuario.

document.getElementById():
Permite acceder a un elemento del DOM a través de su atributo id. Es ampliamente utilizado para manipular el contenido de elementos HTML.

Ejemplo de getElementById:

```
const test = document.getElementById("probando");
console.log("getElementById -->>", test);
```

querySelector() y querySelectorAll():
Estas funciones permiten seleccionar elementos del DOM usando selectores CSS. querySelector() devuelve el primer elemento que cumpla el selector, mientras que querySelectorAll() devuelve una lista de todos los elementos que lo cumplan.

Ejemplo de querySelector y querySelectorAll:

```
const lista = document.querySelector("ul");Selecciona el primer
                                           ul encontrado
console.log("querySelector -->>", lista);
const elementos = document.querySelectorAll("li");
Selecciona todos los li
console.log("querySelectorAll -->>", elementos);
const probandoQuerySelector = document.querySelector(".probando2");
Seleccionar por clase
console.log("querySelector con clases -->>", probandoQuerySelector);
```

innerHTML:
Permite acceder o cambiar el contenido HTML de un elemento. Es muy útil para insertar contenido dinámicamente.

Ejemplo de innerHTML:

```
const div = document.getElementById("miDiv");
div.innerHTML = "Texto modificado";Cambia el contenido del div
```

parseInt() y parseFloat():
Estas funciones convierten una cadena en un número entero o de punto flotante, respectivamente.

Ejemplo de parseInt y parseFloat:

```
const numeroEntero = parseInt("10");
console.log("parseInt -->>", numeroEntero);
const numeroDecimal = parseFloat("3.14");
console.log("parseFloat -->>", numeroDecimal);
```

Number():
se utiliza para convertir un valor en un tipo de dato numérico Si el valor no se puede convertir a un número válido, el resultado será NaN (Not a Number).

También se puede usar este método para convertir valores de otros tipos de datos, como cadenas (strings) y booleanos, a números.

Ejemplos de Number:

```
Convertir una cadena a un número
const numero1 = Number("10"); resultado: 10 (número)
console.log("Number -->>", numero1);

Convertir un booleano a un número
const numero2 = Number(true); resultado: 1 (verdadero se convierte a 1)
const numero3 = Number(false);resultado: 0 (falso se convierte a 0)
console.log("Number booleano -->>", numero2, numero3);

Convertir valores que no pueden ser convertidos a números
const numero4 = Number("Hola");resultado: NaN (no es un número válido)
const numero5 = Number(null);resultado: 0 (null se convierte a 0)
```

¿Qué es un argumento?

Cuando se llama a una función o método, se pueden pasar uno o varios valores entre paréntesis separados por comas. Estos valores que pasamos reciben el nombre de argumentos.

¿Qué es la coerción?

La coerción en JavaScript se refiere a la conversión automática de un tipo de dato a otro durante una operación o una comparación. JavaScript es un lenguaje de tipado dinámico, lo que significa que las variables no están asociadas a un tipo de dato específico, y la coerción ocurre cuando se realizan operaciones con diferentes tipos de datos.

Ejemplo de coerciónt:

```
let b = 10;x es de tipo número
let y = "5";y es de tipo cadena (string)

Coerción implícita
let suma = b + y;  Se convierte "y" a número y se realiza la suma:
                    10 + 5 = 105
console.log("Coercion suma -->>", suma, typeof suma);

En lugar de realizar una suma aritmética, la operación x + y realiza una
de cadenas porque uno de los operandos (y) es una cadena.
```

Manipulando el dom con Eventos

En JavaScript, los eventos son acciones o sucesos que ocurren en la página web y que pueden ser detectados y manejados por el código. Los eventos pueden ser generados por el usuario (como hacer clic en un botón) o por el propio navegador (como cargar una página).

algunos eventos que se pueden detectar con JavaScript son:

Eventos de ratón:
click: Se activa cuando se hace clic en un elemento. dblclick: Se activa cuando se hace doble clic en un elemento. mouseover: Se activa cuando el ratón pasa sobre un elemento. mouseout: Se activa cuando el ratón sale de un elemento.

Eventos de teclado:
keydown: Se activa cuando se presiona una tecla. keyup: Se activa cuando se suelta una tecla. keypress: Se activa cuando se presiona una tecla y se mantiene pulsada.

Eventos de formulario:
submit: Se activa cuando se envía un formulario. reset: Se activa cuando se resetea un formulario. change: Se activa cuando el valor de un campo de formulario cambia (por ejemplo, una casilla de verificación o un menú desplegable).

Eventos de arrastre y soltar:
dragstart: Se activa cuando se inicia el arrastre de un elemento. dragover: Se activa cuando un elemento se está arrastrando sobre otro. drop: Se activa cuando se suelta un elemento arrastrado sobre otro.

Eventos de ventana:
resize: Se activa cuando se cambia el tamaño de la ventana del navegador. scroll: Se activa cuando se desplaza el contenido de la página.

La funcion addEventListener.

es un método en JavaScript que se utiliza para escuchar eventos en elementos HTML y ejecutar funciones en respuesta a esos eventos. Es una forma común de manejar interacciones del usuario y hacer que una página web sea más interactiva.

Ejemplo de sintaxis addEventListener:

```
elemento.addEventListener(evento, =>{
  Código a ejecutar cuando se dispare el evento
});
```

Analizando estructura de addEventListener.

Elemento:
Es el elemento HTML al que deseas agregar el evento. Puede ser un elemento específico del DOM (Document Object Model) seleccionado a través de document.getElementById, document.querySelector, u otros métodos de selección de elementos.

Evento:
Es una cadena que representa el nombre del evento que deseas escuchar. Por ejemplo, "click" para detectar un clic del ratón, "submit" para detectar el envío de un formulario, "keyup" para detectar cuando una tecla del teclado es liberada, etc.

Código a ejecutar cuando se dispare el evento:
Sección al interior de la función para escribir el código que deseamos ejecutar cuando se dispare el evento.

Ahora que ya sabemos las bases de JavaScript vamos a hacer algo de código, para eso te invito a desarrollar algunos ejercicios que prepararé para ti mucha suerte...!!