## **// Comentarios de una línea.**

## /\*

## Cometario

## en varias

## lienas

## \*/

## // Declaración de variable con var y asignación de valor

## var saludo = "Hola como estas?";

## console.log(saludo);

## var numero1 = 5;

## var numero2 = 10;

## var suma = numero1 + numero2;

## console.log("La suma de " + numero1 + " y " + numero2 + " es " + suma);

## var numero3 = "5";

## var numero4 = "10";

## var suma2 = numero3 + numero4;

## console.log("La suma de " + numero3 + " y " + numero4 + " es " + suma2);

## **/\* tipos de datos**

## **- cadenas de texto (String)**

## **- numéricos (Number) -> (Integer, Float, Real)**

## **- Booleanos (True, False)**

## **- Arreglos**

## **- Objetos**

## **- Funciones**

## **\*/**

## var esMayorDeEdad = true; // convención de nombre de variables en camelCase

## // var es\_mayor\_de\_edad = false; // convención de nombre de variables en snake\_case

## // var es mayor de edad = false; // no se puede usar espacios en los nombres de las variables

## // esMayorDeEdad no es igual a esMayorDeEdad

## console.log("Es mayor de edad: " + esMayorDeEdad);

## **/\* Tipos de operadores**

## - Asignacion (=)

## - Matematicos o Aritmeticos (+, -, \*, /, %)

## - Incremento y Decremento (++, --)

## - Operadores logicos (Negacion, and(y) &&, or(o) ||)

## - Operadores de comparacion (>, <, ==, !=, >=, <=) (===, !==)

## \*/

## var resta = numero1 - numero2;

## console.log("La resta de " + numero1 + " y " + numero2 + " es " + resta);

## var multiplicacion = numero1 \* numero2;

## console.log(

## "La multiplicacion de " + numero1 + " y " + numero2 + " es " + multiplicacion

## );

## var division = numero1 / numero2;

## console.log("La division de " + numero1 + " y " + numero2 + " es " + division);

## var modulo = numero2 % numero1;

## console.log("El modulo de " + numero2 + " y " + numero1 + " es " + modulo);

## /\*

## dividendo |\_\_\_\_ divisor

## resto cociente

## cociente = dividendo / divisor

## resto = dividendo % divisor

## \*/

## var incremento = numero1++; // numero1 = numero1 + 1;

## console.log("El incremento de " + numero1 + " es " + incremento);

## var decremento = numero1--; // numero1 = numero1 - 1;

## console.log("El decremento de " + numero1 + " es " + decremento);

## **// Condicionales**

## // if, else

## if (numero1 > numero2) {

## console.log("El numero 1 es mayor que el numero 2");

## } else {

## console.log("El numero 1 no es mayor que el numero 2");

## }

## **// do, while**

## /\*

## Características clave:

## El bloque de código dentro de do {} se ejecuta al menos una vez,

## incluso si la condición es falsa desde el principio.

## Después de ejecutar el bloque, se evalúa la condición.

## Si la condición es verdadera, se repite el ciclo; si es falsa, termina.

## \*/

## do {

## // Código a ejecutar al menos una vez

## } while (condición);

## **// Switch**

## /\*

## Claves:

## expresión se evalúa una vez y se compara estrictamente (===) con los valores de cada case.

## break evita que se ejecuten los siguientes casos (es opcional pero generalmente necesario).

## default es también opcional, y se ejecuta si no hay coincidencia.

## \*/

## switch (expresión) {

## case valor1:

## // Código a ejecutar si expresión === valor1

## break;

## case valor2:

## // Código a ejecutar si expresión === valor2

## break;

## // Puedes añadir más casos según sea necesario

## default:

## // Código a ejecutar si ningún caso coincide

## break;

## }

## // ciclo for

## /\*

## Componentes:

## inicialización: se ejecuta una sola vez al comienzo (por ejemplo, let i = 0).

## condición: se evalúa antes de cada iteración. Si es true, se ejecuta el bloque.

## incremento: se ejecuta al final de cada iteración.

## \*/

## for (inicialización; condición; incremento) {

## // Código a ejecutar en cada iteración

## }

## **// Arrow function (Funcion flecha)**

## // Arrow Function – Estructura básica

## const nombreFuncion = (parámetros) => {

## // Código a ejecutar

## };

## // 2. Sin parámetros (usa paréntesis vacíos):

## const decirHola = () => {

## console.log("Hola mundo");

## };

## // 3. Con retorno implícito (sin return, sin llaves):

## const cuadrado = (x) => x \* x;

## // Si usas llaves {}, necesitas return para devolver un valor.

## const divideByTwo = (Number) => number / 2;

## const multiByTwo = (Number) => number \* 2;

## console.log(divideByTwo(4));

## console.log(divideByTwo(8));

## console.log(divideByTwo(12));

## console.log(divideByTwo(6));

## console.log(multiByTwo(5));

## console.log(multiByTwo(8));

## console.log(multiByTwo(9));

## console.log(multiByTwo(4));

## **// Arrays**

## /\*

## Características clave:

## Los arrays almacenan listas ordenadas de elementos.

## Los elementos pueden ser de cualquier tipo: números, strings, objetos, otros arrays, etc.

## El primer índice es 0.

## Métodos integrados

## JavaScript proporciona muchos métodos útiles:

## .push(), .pop(), .shift(), .unshift(), .map(), .filter(), .reduce(), .forEach(), .find(), .includes(), etc.

## Aunque tiene una sintaxis especial, técnicamente un array es un objeto con claves numéricas.

## \*/

## // Array – Estructura básica

## const miArray = [elemento1, elemento2, elemento3];

## // Operaciones comunes (estructura simple)

## // Acceder a un elemento:

## miArray[0]; // Primer elemento

## // Modificar un elemento:

## miArray[1] = "nuevo valor";

## // Agregar un elemento al final:

## miArray.push("nuevo");

## // Quitar el último elemento:

## miArray.pop();

## // Agregar al inicio:

## miArray.unshift("inicio");

## // Quitar del inicio:

## miArray.shift();

## // Recorrer con for:

## for (let i = 0; i < miArray.length; i++) {

## console.log(miArray[i]);

## }

## // Recorrer con for...of:

## for (let elemento of miArray) {

## console.log(elemento);

## }

## //ejemplo

## var myArray = [, , ,];

## myArray.push(49);

## myArray.shift();

## myArray.push("I love JavaScript");

## myArray.shift();

## myArray.push(120);

## myArray.shift();

## console.log("Mi edad es de " + myArray[0] + " Años");

## console.log(myArray[1]);

## console.log("Mi direccion es Paraguay " + myArray[2] + " Depto 2");

## **// Objetos**

## /\*

## Características clave de los objetos en JavaScript

## Colección de pares clave-valor

## Cada propiedad del objeto está compuesta por una clave (key) y un valor (value).

## Ejemplo:

## "nombre": "Carlos"

## \*/

## // Estructura básica de un objeto

## const persona = {

## nombre: "Carlos",

## edad: 30,

## esEstudiante: true,

## };

## // 2 - Claves como strings

## // Las claves son cadenas (aunque puedes escribirlas sin comillas si no tienen

## // espacios ni símbolos especiales).

## // 3 - Acceso a propiedades

## persona.nombre;

## // Por corchetes (útil si la clave está en una variable o tiene caracteres especiales):

## persona["nombre"];

## /\* 4 - Pueden contener cualquier tipo de valor

## Los valores pueden ser:

## Números, strings, booleanos

## Arrays

## Funciones (métodos)

## Otros objetos (anidados)

## \*/

## // 5 - Modificables (mutables)

## // Puedes agregar, cambiar o eliminar propiedades después de crear el objeto:

## persona.apellido = "Pérez"; // Agrega

## persona.edad = 31; // Modifica

## delete persona.esEstudiante; // Elimina

## // Métodos (funciones dentro del objeto)

## // Puedes definir funciones dentro de un objeto:

## const persona = {

## nombre: "Ana",

## saludar: function () {

## console.log("Hola, soy " + this.nombre);

## },

## };

## // 7 - Iteración con for...in

## // Para recorrer las claves:

## for (let clave in persona) {

## console.log(clave + ": " + persona[clave]);

## }

## /\* 8 - No tienen orden garantizado

## A diferencia de los arrays, el orden de las propiedades en los

## objetos no está garantizado (aunque los navegadores suelen mantenerlo).

## \*/

## var persona = {

## nombre: "Christian",

## apellido: "Sanchez",

## edad: 49,

## empleado: true,

## };

## console.log("Nombre: " + persona.nombre);

## console.log("Apellido: " + persona.apellido);

## console.log("Edad: " + persona.edad);

## console.log("Trabaja: " + persona.empleado);