***¿Qué es una variable?***

Un espacio de memoria donde se almacena información que podemos reutilizar en un futuro.

Lo más común para generar una nueva variable es **“var”**

***Estándar del lenguaje (la manera correcta de escribir código):***

Existen varios formatos de nomenclatura de variables:

* Camel Case: unaMayusculaPorPalabra
* Snake Case: un\_guion\_bajo\_por\_palabra
* Kebab Case: ahora-con-guion-medio

var nombre = ‘Sofia’ ;

El signo “=” es un operador de asignación

Las comillas simples son para envolver un string (no es ningún valor, sino simplemente un texto)

el “;” lo pongo cada vez que finalizo una línea de código

El console.log() nos permite mostrar datos en la terminal

DUDA: Cuando utilice el Terminal, no me encontraba el archivo porque estaba en una carpeta dentro de una carpeta. Hice el cambio y guardé el archivo en donde veía que estaba buscando la terminal, pero cómo elijo yo dónde tiene que buscar la terminal mi archivo? No recuerdo bien eso.

Un bloque de ejecución es todo aquel código que se encuentra dentro de las llaves

La variable “**let**” funciona para crear variables, al igual que “var”. Pero la diferencia es que la variable “let” funciona dentro de un mismo código de ejecución. De esta manera, esa variable que creamos con “let” deja de existir cuando pasamos a un bloque distinto, y hasta puede haber dos variables “let” con el mismo nombre y nunca sobreescribirse; esto no ocurre así con la variable “var”.

Las tres formas de crear una variable:

* var
* let
* const

Estas tres existen para que puedas crear las variables según tus propias necesidades

**“const”**, a diferencia de las otras dos, una vez declarada no se puede redefinir bajo ninguna circunstancia. Pero, al igual que la variable let, estará disponible solo en el bloque de código en el que se haya declarado. Se mantiene constante, por eso su nombres. Las constantes se las debe declarar con palabras totalmente en mayuscula. por ej: const EDAD = 22;

DUDA: NO COMPRENDI BIEN LA DIFERENCIA ENTRE LET Y CONST

YA QUE CON LA VARIABLE LET, TAMPOCO ME DEJABA REDEFINIRLA DENTRO DE UN MISMO BLOQUE, PERO SI EN BLOQUES DIFERENTES, AL IGUAL QUE PUDE CON LA VARIABLE CONST

***Tipos de datos:***

-numéricos:

let precio = 19.95;

-cadenas de caracteres (string):

let nombre = ‘Sofia’;

-lógicos o booleanos:

let laCharlaEstaGenial = true;

let hayAsadoAlFinal = false;

-objeto literal

let persona {

nombre: ‘Sofia’, (edad es el atributo y el string el valor)

edad: 22

}

-array

let persona [ ‘Sofía´, 22, 1.60 ] (‘Sofia’ está en la ubicación 0 y 1.60 en la ubicación 2, por si luego quiero acceder a estos valores)

***Tipos de datos especiales:***

-NaN (Not a Number):

let malaDivision = ‘35’ / 2 (‘35’ es un string y no un número, por lo tanto no se puede dividir por 2, y no puede ser parseado como número)

-Null (valor nulo):

let temperatura = null; (lo asignamos nosotros para indicar un valor vacío o desconocido)

-Undefined:

let saludo; (no le asignamos ningún valor, no tiene nada definido)

saludo = ‘hola’ (ahora sí está definido)

***Para escribir comentarios en el archivo que no queremos que se ejecuten:***

Comienza con //

Y a continuación escribimos el comentario, que servirá para el equipo con el que trabajemos o nuestro yo futuro

***Operadores aritméticos:*** =, +, -, /, \*, %, ++, --

Los operadores ++ y -- se escriben antes del valor en el que quiero realizar la operación, por ej:

let numero = 5;

let numeroMenor = --numero; (sin espacio entre operador y

variable)

***Otros operadores:***

Comparación simple (para preguntar si un valor es igual a otro): ==

Comparación estricta (Compara el valor, pero además pregunta si el tipo de dato es el mismo): ===

Desigualdad simple: !=

Desigualdad estricta: !==

< (menor que)

> (mayor que)

<= (menor o igual que)

>= (mayor o igual que)

Estos operadores nos van a dar como resultado datos booleanos (true y false). A menos que le demos alguna instrucción aparte.

***Operadores lógicos:***

And (chequea que todas las condiciones de la operación sean verdaderas. Con que solo una sea falsa, el resultado será false): &&

Or (Chequea que ALGUNA de todas las condiciones sean verdaderas. Con que una sea verdadera, el resultado ya será true): ||

Negación (va a cambiar al valor opuesto al que tenía antes, ej: si antes era falso ahora será negativo): !

***Operador de concatenación:***

Concatenación (Une dos o más cadenas de texto (string) en una sola: +