1. En un archivo de texto separado que debes crear, escribe explicaciones de los siguientes conceptos como si se lo estuvieras explicando a un niño de 12 años. Hacer esto te ayudará a descubrir rápidamente cualquier agujero en tu comprensión.
   * Variables
   * Strings
   * Funciones

* Argumentos de funciones
* return
  + Declaraciones if
  + Valores booleanos (true, false)
* for
* &&, ||, !

1.- Qué es una variable? Cuáles son los tipos de variables?

Una variable es una forma de poder almacenar el valor de algo para luego poder usarlo.

En el programa de Javascript para crear una variable utilizamos la palabra:

**var** seguida de un espacio, más el nombre de la variable, con el cual luego la vamos a identificar. Para poder asignar un valor a la variable debemos colocar el signo =. Por ej:

var nombre = ‘Carlos’

Aunque existen tres formas de declarar una variable:

**var** es la forma de declarar una variable en la version anterior de JavaScript (ES5)

**let** es una palabra clave nueva de la version actual ES6 de JavaScript. Es muy similar a **var,** aunque en algunos aspectos se comporta distinto. Lo mas notable, es que difiere al crear un “nivel de scope”.

**const**  es otra palabra clave nueva de la version actual de JS6. Pero const es una variable cuyo valor no se podra cambiar. Cons proviene de la abreviatura de constante.

Otro concepto, importante es **console.log**  es una forma simple de imprimir en la consola todo aquello que le pongamos dentro del parentesis. Por ejemplo:

var nombre = ‘Carlos’

>console.log(nombre);

>Carlos

2.- Que son las string?

**Las “strings”** son los valores que se definen como bloques de texto entre comillas simples o dobles.

**var** nombreGato **=** ‘Garfield’;

A diferencia de **Los “numeros”**, estos valores **NO** se encierran entre comillas.

**var** numeroPositivo= 87;

**var** numeroNegativo= -87;

**Los “Booleans”**, se los utiliza para daatos binarios porque solo pueden arrojar dos valores posibles en los datos**: True or False.**

**var** meEncantaprogramar **= true;**

**var** nomeGustaprogramar **= false;**

Por ultimo, **Las “Strings”**, **los Numeros** y **los Booleans, son tres tipos de datos en javascript.**

Pero existen dos objetos en Javascript que no entran en ningun tipo de datos. Estos son los valores: **UNDEFINED y NULL**.

Por un lado, obtendremos UNDEFINED -configurado por JS- cuando busquemos algo que no existe, algo que aun no ha sido definido; Es decir una variable que aun no tiene un valor asignado.

Por otro, obtendremos NULL -definido por los desarrolladores-, cuando los desarrolladores quieren comunicar a otros desarrolladores que el elemento buscado existe pero que aun no hay ningun valor asociado al mismo.

IMPORTANTE: ni **undefined**, ni **null**, ni los **booleanos** -true or false- son strings, por ende ninguno de ellos NUNCA llevara comillas.

3.- Que son las funciones?

Son una parte esencial en todo lenguaje de programacion y sobre todo en JS. Son tipo particulares de OBJETOS, llamados objetos invocables (callable objets), por lo que tienen iguales propiedades que cualquier objeto.

Son formulas para realizar determinadas operaciones.

En Javascript vamos a realizar operaciones a traves de los OPERADORES. Estos son los signos que ya conocemos: (**+** ; **-** ; **\***; **/**) que indican a JS que clase de operaciones debe ejecutar.

Por ej: Si JS ve el operador +, sabe que debe ejecutar la operacion o funcion suma.

var a = (b+c)

console.log (a)

// b+c

Existen diferentes formas de escribir estas funciones:

* Expresiones INFIX

A+B

A+B\*C

A-C/B

Es decir los operadores (+-\*/), siempre se ubicaran entre los operandos.

* Expresiones PreFIX

El operador (+-\*/) se ubica antes de los operandos.

+A\*BC

+AB

* Expresiones PostFIX

El operador (+-\*/) se ubica despues de los operadores.

AB+

ABC\*+

Es importante comprender los conceptos de PRECEDENCIA DE OPERADORES y ASOCIATIVIDAD porque cuando tenemos mas de un operador (+-\*/) en la misma expresion de la funcion, nos permitira entender como interpreta JS que debe ser el orden de ejecucion de las operaciones.

Entonces, la PRECEDENCIA DE OPERADORES es basicamente el orden en que se van a ejecutar las funciones de los operadores. Estas funciones son llamadas en orden de precedencia cuando tenemos mas de un operador, el interprete de JS llamara a ejecutar al operador de mayor precedencia primero, de acuerdo a la tabla de precedencia.

Cuando tenemos mas de un operador en la expresion del mismo rango de precedencia, se da el fenomeno de ASOCIATIVIDAD DE OPERADORES. Donde su orden de ejecucion sera de derecha a izquierda o de Izquierda a derecha, de acuerdo lo indique la tabla de precedencia.

Por ej:

Para resolver una expresion y saber que resultado nos traera el interprete de JS, es necesario conocer la precedencia de los operadores para intuir en que orden ejecutara las operaciones.

console.log (3+5/5)

Al ver la tabla, vemos que la division tiene una precedencia de 15 y la suma de 14. Por ende, el interprete de JS primero ejecutara la division y luego la suma.

console.log(3+1)

console.log(4)

El otro caso es cuando los operadores tienen el mismo rango de precedencia.

var a=1, b=2, c=3;

a=b=c

El = tiene un rango de precedencia 3 y dice que se ejecutara del siguiente modo: Right-to-Left

Entonces, C=3 B=3 A=3, si la asociatividad hubiera sido al reves Left-to-Right, todos los valores serian igual a 1.

OBJETOS GLOBALES Y METODOS

JS tiene varios OBJETOS como CONSOLE, MATH, etc y cada OBJETO tiene su METODO. Por ejemplo:

CONSOLE.LOG donde console es el objeto y log es el metodo.

MATH.POW donde MATH es el objeto y POW es el metodo.

Donde Math.pow sirve para elevar un numero a un exponente

Math.pow (3,2) =9;

Donde Math tambien tiene metodos que redondearan los numeros por nosotros, por ejemplo:

ROUND, redondeara el numero al mas cercano

Math.round (7.5) = 8

Math.round (6.4) = 6

FLOOR, redondeara el numero, al numero entero mas cercano hacia abajo.

Math.FLOOR (5.9) = 5

CEIL, redondeara el numero, al numero entero mas cercano hacia arriba.

Math.CEIL (5.1) = 6

**INTRODUCCION A LAS FUNCIONES**

Cuando tenemos un conjunto de variables, necesitamos funciones para calcularlas, cambiarlas, hacer algo con ellas.

**TRES FORMAS DE CONSTRUIR UNA FUNCION**

**function** miFuncion **( ) { } Usaremos esta por ahora**

var otraFuncion = function ( ) { };

var yOtra = 9 ( ) => { };

**ANATOMIA DE UNA FUNCION**

**function** miFunction ( ) { }

Primero debemos declarar la **function** a JS para que la pueda tratar como tal. Luego viene el nombre de la funcion que debe relacionarse con la funcion que hacen. Luego viene un parentesis abierto y uno cerrado. Finalmente, se abren y cierran corchetes para ejecutarse.

function logHola ( ) {console.log(‘hola!’);}