Intro a Javasscript

Algoritmo – conjunto de instrucciones o reglas definidas y no ambiguas que te permite solucionar un problema.

Javascript

Lenguaje de programación interpretado, se utiliza en sitios web para ejecutar acciones en el lado del cliente, estando embebido en el código fuente de la página web.

**Consola del navegador, uso de Javascript en HTML, Entradas/Salidas y Comentarios**

Salidas y entradas

Console.log(‘ Hola mundo’)

=== para validar que un valor sea igual a otro

Document.write()

Alert = mostrar mensajes al usuario

Prompt = pide al usuario ingresar algo

Comentar el código = /\* INICIO - FIN \*/ o // o //

**ETIQUETA SCRIPT**

Se debe colocar el script en el **head** con el atributo defer src

Defer src=

**Tipos de datos, Variables y operadores matemáticos**

**Tipos de datos**

**Strings** = se utiliza para representar datos textuales, se presentan encerrando el valor entre comillas.

“soy un string”

**Numéricos**

Son valores numéricos, números con punto flotante (decimales)

**NaN Not a number**

**Booleans**

Es un dato lógico que solo puede tener los valores true o false

Podemos establecer acciones, dependiendo de si se cumple o no cierta condición

**Null**

Es un valor nulo, indefinido o no inicializado.

**Undefined**

Una variable a al cual no se le haya asignado valor

Ej. Var saludo;

**Symbol**

**Object**

La representación de un objeto real en código,

Var persona = {

Nombre=

Edad=

Cuidad=

}

**Operadores matemáticos**

Suma +

Resta -

Multiplicaciones \*

División /

Residuo %

Exponencial \*\*

**Variables**

Es un símbolo que te permite identificar una variable “x”

Se utilizan para guardar y almacenar información

Var nombre = “Pedro”; – tiene que ser descriptiva

\*Es importante dar un nombre descriptios nuestras variables

\*utilizamos **camelCase**

**Condicionales**

Nos vana ayudar a decir qué acción vamos a ejecutar. La evaluación de condiciones, sólo puede arrojar 1 de 2 resultados, verdadero o falso.

**Operadores relacionales**

**Símbolos que se usan para comparar dos valores**

Mayor que >

Menor que <

Mayor igual a que >=

Menor igual a que <=

Igual que ===

Diferente que ¡==

**Ojo null y undefined no son iguales**

Null = vacio

undenfined =indefinido

**Operadores lógicos**

**&& (and**) solo si ambas condiciones se complen (son verdaderas), el resultado es verdadero

True y true = true

**Or (||)** si una sola condición se cumple (es verdadera), es igual a verdadero

**Declaraciones IF / ELSE**

Ejecuta una sentencia su una condición especificada es evaluada como verdadera, si es evaluada como falsa, otra sentencia puede ser ejecutada como else.

**Arrays**

Manera ordenada de almacenar una lista de elementos de datos bajo un solo nombre de variable. Con la capacidad de acceder a cada elemento individual de la lista.

Se representa con [] puede ser vacio o también contener elementos

Var verduras = []

Var frutas = [‘Pera’, ‘Manzana´]

Los elementos pueden ser de tipo diferente: string número booleanos

Puedes acceder a los conenidos por medio de ***índice o index***

Frutas [ 0,1,2,3, etc…]

Frutas [3] = “Uva” reemplaza el valor del elemento en el array

**Lenght,** esta propiedad permite saber la longitud del arreglo.

***Iterar (recorrer) el arreglo***

Frutas.lenght

**Métodos de arrays**

**Push y pop – trabajan al final de array**

Push agrega in item al final de la lista

Var.push ( )los métodos requieren paréntesis-

Pop elimina el item al final de la lista

Var.pop( )

**Unshift y shift**

Unshift agrega in item al principio de la lista

Shift elimina un elemnto al principio

**Split**

Divide una matriz en subcadena, convierte un string que vienen con comas o puntos en un array para poder iterar

Var arregloVerduras = Var.split(‘ ,’);

**Slice**

Crea un nuevo array a partir de la posición en la que se los indiquemos.

Devuleve parte de una cadena

Var.slice

Slice cuenta 0,1,2,3

(start) =1, 2 saca dos elementos. slice (2)

(end) = (1,3)

**Splice**

Agregsa o borra elementos de un arreglo original, modifica el original

Agregar Var. Splice (index, numero- a borrar, “elementos”)

Var.Splice(index -desde donde empieza, y numero – loq ue va a borrar, + elementos)

**SORT**

Ordena la lista de forma ascendente (A-Z) por defecto

Var.sort();

.sort(function (a,b) {return a-b});

**Reverse**

**Pone el array al revés**

Var.reverse

**Concat**

Unir dos arreglos en uno solo

Crear una variable para asigaar la **variable.concat(variable)**

**Estructuras de control**

Nos permite repetir un loque de instrucciones, hasta que se compre cierta condición.

Cada repetición, se llama iteración.

**For**

Repite el bloque de instrucciones un número determinado de veces.

*Se utilizan contadores y acumuladores*

*Contador =: una variable que lleva la cuenta del npumero de veves que se*

Acumulador = acumula el resultado de una operación = precios los sumas, string se vuelve más largo

**While**

Mientras se cumpla condición realizar estas acciones. Cuando la condición deje de cumplirse continua el flujo del programa.

Do while

**Funciones**

Porción de código reusable que puede ser llamada en cualquier momento desde nuestro programa.

Receta

Parámetros o ingresar parámetros

Ingresar instrucciones

Llamara la función que declaramos

Function **MyFunction** (Parámetros){

}

Return = “some text here” + variable

Devuelve el valor de una operación. Esto sirve para asignar ese valor a una varable.

Función Pura

Función impura

solo trabaja con valores positivos

**Objetos literales**

Tienen un valor

Las propiedades están declaradas textualmente en el código

Para acceder alas propiedades de un objeto dentro deun método es necesario usar la palabra reservada this

**Desfragmentación de objetos**

**CONST { NOMBRE, EDAD, PESO, ETC} = CONST;**

DOM JS