**APUNTES 1 DE JAVASCRIPT**

QUE ES JAVASCRIPT?

- es un lenguaje de programacion dinamico

DIFERENCIA CON JAVA

- la manera que interpretan las clases(objetos) son distintas

NOTAS

- **no permite ni lectura ni escritura del lado del servidor**

**-**no podemos conocer la estrutura de datos del servidor

- **no puede utilizarlo para aplicaciones de red POR SI SOLO**

- **NO TIENE CAPACIDAD DE MULTIHILOS**

- cuando hay un proceso que se ejecuta fuera el hilo principal

- manda a este proceso al **stack de funciones** (este stack (es una pila) es de todos las funciones del programa

-en javascript es sensible a mayusculas y munisculas osea es distinto una “a” que una “A”

-**NO PONER NUESTRAS VARIABLE EN EL <head></head>**

**-mejor poner todos nuestro codigo javascript en un fichero .js y el html en otro**

**-**las variables tienen que ser definidas antes de usarlas

-var a = 1;

-si no lo hacemos y la usamos antes de definirla nos dara “undifined”

-para comentar en javascript es:

// para una linea

/\* \*/ para multilinea

- en javascript se puede cambiar el tipo de una variable despues d edeclararla

-como en python

-esto se llama estructura dinamica

-cuando se importa o se corre un programa en JS

-JS primero hace un barrido del todo el codigo

-esto **solo para reservar espacios de memoria y darles nombre a los espacios**

**-PERO NO ASIGNA VALORES**

**-**entonces se puede invocar una funcion antes de declararla

**-**JS despues del primer barrido, **recien ejecuta el programa como tal**

-para JS 3 es igual a “3”

-entero es igual a string en tanto sea el valor igual

**PROGRAMAR**

- se lo puede ejecutar en chrome

- el servidor va a buscar el archivo llamado “index.html” o .jsp o .php

- pero tiene que ser index

- en el navegador una vez corrido el programa .html

- vamos a los 3 puntitos > mas herramientas>herramientas de desarrollador>console

- clik derecho>inspeccionar>console

-en console escribiendo window vemos los objetos de la pagina

-window es el objeto principal (obejto global) en javascript

-aqui podemos ver, crear y modificar variables y objetos

-si cambio variables dentro de console

-son entiempo real durante la ejecucion del programa

-puedo escribir los comandos js aquí directamente

-una vez le dijimos al navegador que el fichero tiene un script crea el entorno (el objeto global)

-las variables, clases o objetos se alojan en el objeto principal

TIPOS DE INFORMACION DE CONSOLA

console.log(variable);

console.warn(variable);

-este es un warning

console.error(variable);

-este es un error pero que no detiene la ejecucion del programa

console.info(variable);

-este es uno de informacion

VARIABLES

var a = 1

var b = “texto”

-asi se crean las variables

FUNCIONES

-asi se declara funciones

function nombre(parametro1 , parametro2){

var objeto = {parametro1 : parametro1,parametro2:parametro2}//al hacer esto el parametro se guardara como valor de la llave respectiva

}

funcion(valor1 , valor2);

-asi se invoca a las funciones

-**asi importamos los scripts javascript en el HTML**

**-**<script src=”path”></script>

-las variables que estan dentro de las funciones son variables locales

-solo toman el valor asignado dentro de la funcion

-por mas que exista otra variable global con el mismo nombre

-las funciones pueden acceder a las variables **locales Y GLOBALES**

**-igual que python**

**-**pero el programa principal no puede acceder a las variables locales de una funcion

-si no hacemos que la funcion no retorne nada

-este retornara “undefined”

-la variable de retorno puede ser cualquier cosa

-primitivos,objetos,funciones

-para usar la funcion retornda tenemos que instanciarla

-las funciones retornadas pueden retornar otra funciones

-en caso de no darle un valor a un parametro este solo se guardara con “undefined”

-gracias al primer barrido que hace JS

-se puede mandar una funcion a otra como parametro

-**podemos crear variables de instancia**

**-**estas variables son las que anadimos a una funcion despues de instanciar la misma

-this es como self en python,

-**pero hay que tener cuidado a que objeto apunta cada THIS**

**-**poq dependiendo el objeto puede cambiar sus atributos

**PALABRA RESERVADA THIS**

**-el valor de this depende del objeto que lo invoca**

**-**cuando se

VARIABLES PREDEFINIDAS D ELAS FUNCIONES

nombre\_funcion.name :esto nos devuelve un string con el nombre de la fucnion

**TIPOS DE DATOS**

-existen los datos primitivos y los objetos

-las variables primirtivas cuando se asigna una variable a otra, tiene espacios de memoria distintos

-las variables objetos cuando se asigna una variable a otra comparten el mismo espacio de memoria

-solo tiene nombres distintos (referencias distintas al mismo espacio de memoria)

PRIMITIVOS

var num = 10; //son tipo entero

var str = “texto” // tipo string

var bol = true; // tipo looleano

var und = indefined; // tiop indefinido, cuando no se le asigna valor a la variable

var nul = null; //tipo null

OBJETOS

var obj = {numero:10 , letra:’hola’ , “rodri”:nombre : funcion1 : function(){ } };

-un objeto es una **colección de datos primitivos o de otros objetos**

**-para acceder a un valor del objeto**

console.log(obj.numero)//asi solo para cuando la llave es un string

console.log(obj[‘numero’])//asi para cualquier llave

-para agregar un valor al objeto

obj.nueva\_llave = nuevo\_valor

-el valor de una llave en los objetos pueden ser funciones

**OPERACIONES EN JAVASCRIPT**

a == b : asi preguntamos si ‘a’ es igual a ‘b’

-esto es igualdad simple

a != b : es la misma logica con este operador

3 == “3”

-esto nos dara “true”

a === b : esta es igualdad estricta

3 === “3”

-esto nos dara “false”

a !== b : asi seria desigualdades estrictas

-las **operaciones matematicas** son iguales que en PYTHON

++a : incrementa 1 a “a” **y devuelve “a” incrementado**

a++ : **devuelve “a” original y recien incrementa 1 a “a”**

-el resto de **comapradores** son igual que en python

-los **operadores bit a bit** son iguales que en python

-(en python los operadores logicos de bit a bit son los mismos)

-los **operadores logicos** son asi:

a && b : asi decimos “a and b”

a || b : asi decimos “a or b”

-esta es **condicion ternario:**

condicion ? Resultado1 : resultado2

var estado = (edad >= 18) ? “adulto” : “menor”

-pregunta si edad mayor de 18 si es igual asigna “adulto” a estado

-si es falso asigna “menor”

**CONDICION IF**

if (condicion){

}

else if{

}

else{

}

-**tenemos que poner si o si los parentesis en la condicion**

**CICLO FOR**

-asi es el siclo for

for (var i=0; i<100; i++){

}

**MENSAJE EMERGENTE DE ALERTA**

alert(“mensaje de alerta”);

-con este comando sale una pesatana de alerta con el mesnaje que pongamos

**PALABRA RESERVADA NEW**

function funcion(){

}

var funcion2 = funcion();

-cuando hacemos de esta manera ‘funcion2’ solo resive el valor retornado por la ‘funcion’

-en caso de no especificar un retorno en ‘funcion’ retornara “undefined”

var funcion3 = new funcion()

-usando la palabra reservada “new” estamos haciendo que ‘funcion3’ sea una instancia de ‘funcion’

-**cuando en la funcion querramos poner otra funcion hacerlo de la siguiente manera**

function funcion(){

this.funcion2 = function(){console.log(‘hola)};

}

var instancia = new funcion();

instancia.funcion2();

-asi podremos acceder a todos los atributos de la funcion principal y la secundaria

**MODO ESTRICTO**

**-**para declarar el modo estricto:

“use strict”;

-poniendo esto al inicio del script hace que todo el script se ejecute en modo estricto

-se lo puede poner dentro de una funcion para que solo la funcion trabaje asi

-**solo la funcion estara en modo estricto**

-el modo estricto es un modo de ejecucion del script

-en este modo no se puede dar un valor a una variable que no esta declarada

**-**en este modo no se puede borrar variables u objetos

delete variable;

-esto no esta permitido en modo estricto

-dara error

-no se puede cambiar el valor de variables fijas

“use strict”;

var obj ={};

Object.defineProperty(obj, “x”, {value : 0 , writable : flase});

obj.x = 3.1;

-esto dara error

**PROTOTIPOS:**

-los prototipos sirven para cuando queremos agregar un atributo a una funcion despues de haber declarado la funcion1

function funcion(){

this.atributo =valor

}

funcion.prototype.atributo\_nuevo = valor nuevo;

funcion.prototype.funcion2 = funciont() { }

-los prototipos pueden ser de cualquier tipo, incluso una funcion

**FUNCIONES ANONIMAS**

(function() {

} );

();

-la funcion anonima no tiene que tener nombre

-es como los lmbdas en python

-para declarar la funcion anonima se la pone dentro de aprentesis

-para invocarla solo se pone 2 parentesis

PREGUNTAR TIPO DE OBJETO

typeof(objeto)

-nos devuelve el tipo de objeto, variable o lo que sea

PREGUNTAR SI ES UNA INSTANCIA DE UNA FUNCION

instancia instanceof funcion

-devuelve true si instancia de funcion

ARREGLOS

var arreglo = [];

-asi se declara los arreglos

-funcionan como las listas en python

arreglo.reverse();

-invierte el orden del arreglo en la misma variable

nuevo\_arreglo = arreglo.map(funcion(elem){

elem \*= 2;

return elem;

});

nuevo\_arreglo2 = arreglo.map(funcion)

-map ejecuta la funcion con cada elemento del arreglo y devuelve un nuevo arregle

string\_nuevo = arreglo.join(“|”)

-join convierte el arreglo en un string con los elementos seperados por comas por defecto

-si ponemos un parametro, este sera el separador

arreglo nuevo3 = string.split(‘|’)

-recive un string y lo convierte tomando el parametro como referencia para separar cada elemento

arreglo.push(‘6’)

-aumenta al final del arreglo el parametro que le mandemos, y lo guarda en el mismo arreglo

arreglo.unshift(‘0’)

-aumenta el parametro al inicio del arreglo y lo guarda en el mismo arreglo

arreglo.pop()

-elimina el **ultimo valor del arreglo**

arreglo.splice(posicion\_inicio , cuantos\_borrar, sustituto1 , sustituto2,…..)

-splice eliminara los delementos desde las **posicion** inicio hasta la posicion final, **incluido el final**

-si le pasamos sustitutos pondra estos den lugar de los valores eliminados

-si en posicion final ponemos ‘0’ (cero), no borrara nada y solo pondra los sustitutos

arreglo\_nuevo4 = arreglo.slice(posicion\_inicio , posicion\_final)

-slice deja solo desde la posicion de inicio hasta 1 antes de la posicion final

-y los guarda en un nuevo arreglo

PREGUNTAR EL TAMANO

console.log(lista.length)

-esto imprimira el tamano de la lista

**ARGUMENTOS**

**-**dentro de una funcion esta hace referencia a los **arguemntos realmetne mandados a la funcion cuando se invocar**

**-**esto devuelve una lista con los argumentos

-por mas que no se le especifique parametros, si al invocar la funcion se le pasa argumentos este los recivira

funtion funcion(a,b,c){

console.log(arguments);//de esta manera se usa la variable

}

funcion(1,2);

>>[1,2]