**RESUMEN GENERAL DE JAVASCRIPT A PROFUNDIDAD**

**Importante:**

* Mirar el archivo de Glosario General Entendimiento World Programming para visualizar el significado de ciertos conceptos utilizados en este documento.
* Mirar

**¿Qué es JavaScript?**

* JavaScript es un lenguaje de scripting multiplataforma y orientado a objetos
* Es un lenguaje de interpretación, se hace funcionar de arriba hacia abajo y el resultado de leerlo de esta manera hace que inmediatamente responda. No se tiene que transformar tu código en algo diferente antes de que el navegador lo haga por ti, en comparación de los lenguajes compilados que son transformados antes de que sean leídos por la computadora.
* JavaScript es un lenguaje de un único subproceso ( solo se ejecuta una cosa a la vez)

**+**

* Javascript esta estanderizado en Ecma International ECMAScript
* Javascript es un lenguaje de tipado débil.

**¿Cómo funciona JavaScript?**

* **Mirar en el siguiente link** <http://www.solocodigoweb.com/blog/2018/01/11/como-funciona-javascript/>

**Tipos de Definición Datos**

* **Importante:**  Las variables en javascript son dependientes del contexto de ejecución, el scope ( alcance ) de una variable dentro de una función será solo de la función, si se declara por fuera deberá ser con un scope global.
* **Var:** Actúa para la declaración de variables globales
* **Let:** Actúa para la declaración y uso de variables por bloque, su scope solo será del bloque en donde está ubicado.
* **Const:** Actúa para la declaración de una variable la cual nunca será modificado su valor.

**Tipo de Datos en Javascript**

* **Number**
* **Strings**
* **Booleanos**
* **Arrays**
* **Objetos**

**Operadores**

* **+** SUMA, **-** RESTA, **\*** MULTIPLICACIÓN, **/** DIVISIÓN, **%** MODULO (Valor restante, residuo de una división)
* **Operadores de Incremento:**
  + Incremento++ **(Esto significa incremento = incremento + 1)**
  + Disminucion-- **(Esto significa disminucion = disminucion - 1)**
* **Operadores de Comparación:**
  + **===**  igual
  + **!==**  diferente

**Condicionales**

* Son estructuras de control que nos van a permitir comparar algo. Con esto se realizará una debida acción o otra.
* **IF (CONDICIONAL DE COMPARACIÓN)**
  + Esta condicional se usa cuando se desea comparar valores.
  + If( valor === valor2) {….} \***si valor es igual a valor2\***
  + **NOTA:** Se da uso de los operadores de comparación.
  + **Formas de construir un IF:**
    - **Normal:**
      * If(condicion){ …….. }
    - **Resumida:**
      * if(condicion) return valorRetorno; \***se usa cuando se debe retornar un valor inmediato\***
    - **IF Ternario**
      * Variable = (condicion) ? valorSiPasaCondicional : valorSiNoPasa \***se usa cuando debes darle valor aa alguna variable y para ello necesitas una condicional\***

**Funciones**

**Función por expresión averiguar**

**Importante, javascript ignora las inicializaciones, muy diferente a las declaraciones, las declaraciones son las que sufren el hoisting y dependen del scope, AVERIGUAR BIEN ESTE TEMA**

* **Creación de Funciones:**
* **Forma Normal:**
  + Function nameOfFunction(){ …. }
* **Forma función Anonima:**
  + Var anónima = function(){….}
  + Se puede definir también de esta siguiente manera:
  + **Forma función de Flecha:**
    - var funcionFlecha = (param1,param2) =>{………..}
    - Var anónima = () =>{……………} \***Con esta forma se recomienda usarla cuando se usan más de un parámetro\* si es solo un parámetro no se necesita los paréntesis**
* **Forma con Parámetros:**
  + Function nameFunctionParams( param1, param2) { ………}
  + Los parámetros se pueden crear de manera no obligatoria de pasar así:
  + Function name( param1, param2, paraNoRequest = “”) {……}
  + **También** sirve para darle valores predeterminados
  + **Funcion en parámetros**
  + Se pueden enviar funciones como parámetros, esto se hace cuando se da la necesidad de hacer una función para algo especial dentro de una función, solamente si se usará en este único lugar, osea si se necesita en otras partes, es mejor crear una función normal aparte, pero si no es necesario entonces puedes optar por este método.
  + var valorRetorno = nombreFuncion1(param1, param2, (param1, param2)=>{
    - param1+ param2
  + });
  + Function nombreFuncion1(param1, param2, nombreFunction2){ …… nombreFunction2(param1, param2);}
  + \*  **RECORDAR: es importante que en el lugar donde se invoca la primera función se declare la segunda función dentro de los parámetros que se envirará a la primer afunción, una vez estes en la primera función, debes invocar el parámetro que contiene la función para usar la función2, en el ejemplo donde se invoca la primera función, en el tercer parámetro se le envía una función como parámetro haceindo uso de funciones de flecha \***
  + **Parametro REST and SPREAD**
  + **REST:**
    - Es cuando no se sabe la cantidad de parámetros que se recibirá
    - nameFunction(param1, param2, param3, param4)
    - function nameFunction(…arrayParametrosDesconocidos);
  + **SPREAD:**
    - Es cuando se tiene un array y se envía a la funcióncomo parámetro spread, entonces la función cogerá y llenará con los parámetros en el orden de llegada del array
    - nameFunction(…array);
    - function(param1, param2, param3){…}
* Atrapar Errores recomendado para funciones:
* **TRYCATCH**
  + TryCatch en podría identificarse como “inténtalo” y si falla lo atrapas, es recomendable hacer esto si se sabe que le función fallará.
  + Try{ // Código que podría fallar }catch(error){ alert(“Atrapar el error y un mensaje para el suaurio” + error); }

**--- IMPORTANTE PASO A MANEJO DE ARRAYS/MATRICES**

**¿Qué es un array?**

* Las matrices se describen como “objetos tipo lista”, básicamente son objetos individuales que contienen múltiples valores almacenados en una lista.
  + arrayMuestra = [“prueba”,”array”,”hahaha”];
* **Array multidimensional**
  + Es una array dentro de una array, a esto se le define como una array multidimensional.
  + Array1 = [“marvel”,”oihyeah”];
  + Array2 = [“sibebe”,”siempre”,”jijij”,Array1]
  + \* **Realmente** un array multidimensional / matriz es un array de arrays, osea dentro de un array hay otro array
* **¿Cómo recorrer un array?**
  + **Forma clandestina:**
    - **For(let i = 0;i<array.length();i++){** Esto nos indica que la i será la posición del array en la que este **}**
  + Si es un array normal, se recomienda usar la siguiente función:
  + arrayNormal.forEach( (elemento,indice) => { \***el elemento vendría siendo el valor del índice actual, y el índice es el la posición en la que se encuentra del array** } )
* **¿Cómo recorrer un array multidimensional / matriz o objeto?**
  + **Forma Clandestina:**
    - **For(let i=0;i<array.length();i++){** aquí estarías en la primera posición del array si tienens mas arrays dentro de este harías lo siguiente **for(lei j=0;i<array[i].length();j++){** de esta forma estarías ya recorriendo el array interno del primer array **} }**
  + Si es un array multidimensional se recomienda usar la siguiente función:
  + For(let índice in arrayRecorrer){ **\*de esta forma se puede recorrer de forma eficaz un array multidimensional o un objeto, donde índice viene siendo la posición en la que se encuentre dentro de este array, osea si se tiene un array dentor de un arra podrías acceder a la info de el de la siguiente forma, arrayRecorrer[0][1] donde estas llendo en la posición cero se encuntra el otor array y la posición 1 de ese otro array, si es un objeto es mejor usar el arrayRecorrer[indice].NombreDato\*** }
* **Funciones para Arrays**
  + **Array.push(“valor”) =**  funciona para agregar un elemento al array.
  + **Array.pop() =** Funciona para eliminar el último elemento del array.
  + **StringVariable = Array.join() =** Funciona para volver un array en string
  + **Array.split(“,”) =** Funciona para volver un string en array \***cada nuevo elemento del array se determina por el delimitador que este dentro de la función Split\***
  + **Array.sort() =** Ordenará array por orden alfabetico.
  + **Array.reverse() =** Funciona para ordenar el array de forma al revés
  + **Buscar en un array:**
    - Var lenguaje = arrayPrueba.find(elemento =>{
    - Return elemento == “valorABuscar”;
    - })