**Curso de Javascript Platzi**

**Elementos en javascript:**

**Data**: Basado en tipos de datos

**Tareas**: Acciones que se realizan en base a esos datos.

**Tipos de datos:**

**Valores primitivos:**

**Numerico** (integer)

**String**: Cadena de caracteres

**Booleanos**: true o false, verdadero o falso para casos de decisión

**Null, undefined**: empty values, valores placeholder, cuando queda un valor faltante debe ser gurdado en memoria

**Valores tipo objetos:**

**Array**:[1,2,3,4] Es un arreglo de tipos primitivos

**Object** { “nombre”: “Gianni”, “apellido”:”santucci”}

typeof: Es una palabra reservada que permite ver el tipo de dato de la variable.

**Que son las variables y como utilizarlas:**

Es la representación de un lugar en memoria, que vamos a utilizar:

**Var nombre = “Gianni”;**

**Var es una palabra reservada que indica que el elemento es una variable, debe apartarse un espacio en memoria para guardar ese valor.**

**Estados de una variable:**

**Declarar:**

Var edad;

**Inicializar:** Dar un valor inicial

edad=20

//arreglos

var elementos = [“computadora”,”celular”];

//objeto

var persona = {

nombre : “Gianni”,

edad:34,}

Para acceder a la variable de objetos.

persona.nombre

Funciones:

Son sentencias, para generar cálculos o pasos sobre las variables.

**Declarativas:**

Function miFuncion(){

Return 3;

}

**Expresión**: A través de variables, o funciones anónimas ya que no se le genera un nombre

var miFuncion = function(a,b){

return a+b;

}

Asi la llamo: Se le pasan los parámetros necesarios

miFuncion(1,2)

**Template string**:

Console.log(`Hola ${estudiante}`)

**return**: permite obtener resultado de una función

ejem: return a+b;

A las funciones declarativas se les aplica hoisting, y a la expresión de función, no. Ya que el hoisting solo se aplica en las palabras reservadas var y function.

**Scope**: Alcance de elementos en js

**Scope global**: Se genera al ejecutar el archivo js en el navegador. Todo el entorno js de la aplicación, el global no puede acceder a las variables del scope local

**Scope Local:** Solo lo que esta dentro de la función va a tener acceso a sus elementos. Este scope se genera al crear una función.

**Hoisting**: Que podemos llamar a la función antes de ser creada.

Cuando hablamos de funciones en JavaScript, tenemos dos tipos de funciones: Funciones Declarativas (function declaration / function statement) y Expresiones de función (function expression / funciones anónimas).

**Funciones Declarativas:**

En las funciones declarativas, utilizamos la palabra reservada function al inicio para poder declarar la función:

**function** **saludar**(nombre) {

console.log(`Hola ${nombre}`);

}

saludar('Diego');

**Expresión de función:**

En la expresión de función, la declaración se inicia con la palabra reservada var, donde se generará una variable que guardará una función anónima.

**var** nombre = **function**(nombre){

console.log(`Hola ${nombre}`)

}

nombre(‘Diego’);

En la expresión de función, la función podría o no llevar nombre, aunque es más común que se hagan anónimas.

**Diferencias:**

A las funciones declarativas se les aplica hoisting, y a la expresión de función, no. Ya que el hoisting solo se aplica en las palabras reservadas var y function.

Lo que quiere decir que con las funciones declarativas, podemos mandar llamar la función antes de que ésta sea declarada, y con la expresión de función, no, tendríamos que declararla primero, y después mandarla llamar.

El hoisting no sucede de ecmascript6 en adelante

Con let y const

Console.log(): Es una función del navegador que nos permite imprimir ciertas cosas

**Ejemplo de hoisting:**

hey();

function hey(){

console.log("Este es mi nombre"+ miNombre);

}

var miNombre = "Gianni";

/\*

El resultado del console.log es undefined, debido a que la variable la declare al final y se aplica el hoisting para la función pero no para la variable, las buenas practicas indican que las variables deben estar declaradas primero, luego las funciones y luego los llamados

\*/

**Typeof(a);**

Devuelve el tipo de dato

**Coercion**-.Permite cambiar de un valor a otro en una valor en una variable

**Implicita**: Javascript nos ayuda a hacer el cambio

Ejemplo: 4+”7” devuelve “47”

**Explicita**: Obnligamos que un valor pase de un tipo a otro tipo

Ejemplos

Var a =20;

var c = String(a);

var d = Number( c );

# Valores: Truthy y Falsy

Boolean() : Sino se tiene un valor e resultado es falso

**Valores falsos**

Boolean(0);

Boolean(null);

Boolean(NaN);

Boolean(undefined);

Boolean(false);

Boolean(“”);

Valores verdaderos

Boolean(“a”);

Boolean(1);

Boolean([]); //aunque el array este vacio

Boolean({]); //aunque el objeto este en vacio

Boolean(function(){});

Boolean(true);

# Operadores: Asignación, Comparación y Aritméticos.

**De operación (binarios):**

**+, - , \* , /**

3+2

50-10

10\*20

20/2

“Gianni ”+” Santucci”;

**De asignación**

Var a = 1;

**De comparación:**

**== Doble igual**

3 == 3

=== Triple igual, valida el contenido y el valor sean los mismos

1. === “3”

**>, < , <=, >=**

**&& And**

If ( a && b)

**|| or** : Si alguna de las dos es verdad

If(a || b)

**Para incrementar valores**

++ //Suma uno

A += 2; //Suma dos

# Condicionales: If, Else, else if

var edad = 18;

if(edad==18){

console.log("Puedes votar por primera vez");

}else if(edad >18){

console.log("YA puedes votar de nuevo")

}else{

console.log("aun no puedes votar")

}

//operador ternario: SE hace un condicional en una sola linea

var numero = 1;

var resultado = numero == 1 ? "Soy un uno " : "No soy un uno";

**Switch**

var numero = 1;

switch(numero){

case '1':

console.log("soy uno");

break;

case '2':

console.log("soy un dos");

break;

default:

console.log("soy otro numero");

}

**Arrays**:

Estructura de datos, tipo objetos.

**Métodos que nos permiten mutar el contenido de un array**

var frutas = ["Manazana","Plátano","Cereza","fresa"];

console.log(frutas.length); //Para ver la longitud del array

//para acceder a cada uno de los elementos

console.log(frutas[0]);//Elemento de la lista

//agregar elementos al final del array

var masFrutas = frutas.push("Naranja");

//quitar elementos: Elimina un elemento del array

var ultimo = frutas.pop("Naranja");

//Agregar valor al inicio del array

var nuevaLongitud = frutas.unshift("Pera");

//Elimina un valor que esta al inicio de la posicion

var nuevaLongitud = frutas.shift()

//Obtener la posicion del elemento

var posicion = frutas.indexOf("cereza");

//Elimina el un valor al final del array

Var nuevaLongitud2 = frutas.pop();

# Loops: For y For...of

var estudiantes = ["maria","sergio","rosa","daniel"];

function saludarEstudiantes(estudiante){

console.log(`Hola, ${estudiante}`);

}

//Loops

//For

for(var i=0; i < estudiantes.length; i++){

console.log(estudiantes[i]);

saludarEstudiantes(estudiantes[i]);

}

//For of

//Segundo ejemplo de for: parecido a un foreach, por cada uno.

for(var estudiante of estudiantes){

saludarEstudiantes(estudiante);

}

var estudiantes = ["maria","sergio","rosa","daniel"];

function saludarEstudiantes(estudiante){

console.log(`Hola, ${estudiante}`);

}

# Loops: While

//while: mientras hay elementos en el array estudiantes se ejecuta el ciclo

while(estudiantes.length > 0){

//Usamos el metodo de mutacion shift

//saco del arreglo de estudiantes e ingreso el nombre a la variable estudiante

var estudiante = estudiantes.shift();

saludarEstudiantes(estudiante);

}

# Objects

var miAuto = {

marca: "Toyota",

modelo: "Corola",

annio: 2020,

//Un atributo le objeto puedo ser una funcion del objeto

detalleDelAuto: function (){

//This es la forma de obtener valores del objeto global

console.log(`Auto ${this.modelo} ${this.annio}`);

}

};

//Asi accedo al objeto

console.log(miAuto.marca);

miAuto.detalleDelAuto();

//Me va a mostrar Auto Corola 2020

# Objects: Función constructora

//Funcion constructora

//Se le pasan los prametros que generaran el objeto

function auto(marca,modelo,annio){

this.marca = marca;

this.modelo = modelo;

this.annio = annio;

}

var autoNuevo = new auto("Tesla","Model 3",2020);

console.log(autoNuevo);

//Me muestra el objeto con sus valrores

var autoNuevo2 = new auto("Ford","Fiesta",2000);

var autoNuevo3 = new auto("Toyota","Corola",2020);

# Métodos de recorridos de Arrays

var articulos = [

{

nombre: "bici",

costo: 3000

},

{

nombre: "Tv",

costo: 2500

},

{

nombre: "Libro",

costo: 320

},

{

nombre: "Celular",

costo: 10000

},

{

nombre: "bici",

costo: 3000

},

{

nombre: "laptop",

costo: 20000

},

{

nombre: "Teclado",

costo: 500

},

{

nombre: "audifonos",

costo: 1700

},

];

//Filter: filtrar elementos, valida si es cierto o falso y genera un nuevo array

//EL parametro articulo es el que va a permitir manipular al objeto

var articulosFiltrados = articulos.filter(function(articulo){

return articulo.costo <= 500;

});

console.log(articulosFiltrados)

//Map: Permite m,apear ciertos objetos, genera un nuevo array

// Permite mapear usar un serctor del objeto en este caso nombre

//Asi no necesito hacwr un ciclo para recorrer el array sino que en un array nuevo obtengo eñl campo que se desea

var nombreArticulos = articulos.map(function(articulo){

return articulo.nombre;

})

console.log(nombreArticulos);

# Recorriendo Arrays con .find(), .forEach() y .some()

var articulos = [

{

nombre: "bici",

costo: 3000

},

{

nombre: "Tv",

costo: 2500

},

{

nombre: "Libro",

costo: 320

},

{

nombre: "Celular",

costo: 10000

},

{

nombre: "bici",

costo: 3000

},

{

nombre: "laptop",

costo: 20000

},

{

nombre: "Teclado",

costo: 500

},

{

nombre: "audifonos",

costo: 1700

},

];

//Find: devuelve true o false de la busqueda, solo devuelve un objeto segun lo señalado a diferencia del filter que devuelve una matriz de regitros coincidentes

//Devuelve dentro de un nuevo arreglo no midifica ni muta el original

var encuentraArticulo = articulos.find(function(articulo){

return articulo.nombre ==="laptop";

});

console.log(encuentraArticulo);

//Foreach: filtrar sobre el array original sin modificarlo, lo recorre

//No muta el arreglo original

articulos.forEach(function(articulo){

console.log(articulo.nombre)

})

//Some: retorna true o false para los objetos que cumplan con una condicion

//Trae un nuevo array

// Existen articulos baratos el dice si o no

var articulosBaratos = articulos.some(function(articulo){

return articulo.costo <= 700;

});

console.log(articulosBaratos);