**Variables**

Recomendación, darles los nombres en inglés.En formato camelcase, la primera letra en minúscula cuando venga la siguiente palabra, la primera letra en mayúcula.

Se declara con var (alcance mayor), let(alcance menor, mas usado, el que se debe usar, consume menos recursos) y const (constante, no cambia, consume menos recursos). Las variables pueden cambiar su valor y tipo.

**Tipos de datos (primitivos: no almacenan referencias de datos)**

Número, string, lógicos(verdadero/falso).En los string Podemos acceder a los caracteres dentro de las cadenas con su posición numérica, teniendo en cuenta que la primera casilla es 0. Para ello usaremos los corchetes []

var jediName = "Luke";

console.log(jediName[0]); // => L

console.log(jediName[3]); // => e

console.log(jediName[8]); // => undefined

console.log(jediName[-1]); // => undefined

Valor null es distinto a undefined, el null hay que asignarlo.

Tabla de asignaciones con operadores, imagen1

Precedencia del operador, imagen2

**Asignación de booleano:**

**Const age = 18;**

**Const age2 = 20;**

**Const vanALaMismaClase = age == age2;**

**Console.log(vanALaMismaClase); imprime true o false si la edad es la misma comparada por el ==, compara el valor**

**Para comparar valor y tipo, se usa ===**

**Const vanALaMismaClase = age === age2;**

**Console.log(vanALaMismaClase); imprime false en este caso**

**Const vanALaMismaClase = age == age2;**

Const claseContraria = !vanALaMismaClase; ! operador de negación o not

Console.log(calseContraria); imprime true

También se puede usar el operador !directamente en la declaración de la primera constante

**Const vanALaMismaClase = age != age2; el valor de la constante sería true**

**No solo se compara el valor, también el tipo se puede con !==**

**Const puedeSacarElCarnet = age > 18; sería false**

**Const puedeSacarElCarnet = age >= 18; sería true**

**Igualmente con menor y menor o igual**

**Bucles con cadenas de texto**

Para obtener la longitud de una cadena de texto, usaremos la propiedad **.length** que devuelve el número de caracteres dentro de un **string**

Para obtener una letra de una cadena de texto dentro de un bucle, podemos utilizar los corchetes seguidos de la posición de dicha letra texto[i]. Por ejemplo, para el string **hola** podremosobtener la letra **o** si entramos en la posición 1 (**¡en programación empezamos contando desde 0 siempre!), hola[1].**

**ARRAYS, dato no primitivo**

Los datos dentro del listado pueden ser de cualquier tipo, es decir, es posible crear un array que tenga una cadena (o string) en la primera posición, un número en el segundo, un valor booleano en el tercero y así sucesivamente.Tmabién puede tener objetos, array de objetos:

Var list0fObjs = {

{ id: 1, name: ‘Tim’ },

{ id: 4, name: ‘Bob’ },

};

var ejemplo = ['string', 10, true];

**Propiedad length:**

Si queremos conocer el número de elementos del array avengers se puede consultar mediante la propiedad length al igual que hacíamos con los strings. Os dejamos un ejemplo:

var avengers = ["Hulk", "SpiderMan", "AntMan"]; avengers.length; // Devuelve 3 porque hay 3 elementos

**Método Push en Array:**

Al ser una pila de elementos, se tiende a añadir o eliminar elementos por el final (arriba del todo):

Para añadir elementos en un array, usamos el método push, que inserta un nuevo elemento:

var avengers = ["Hulk", "Thor", "Iron-Man"];

avengers.push("SpiderMan");

console.log(avengers);

// Imprime ["Hulk", "Thor", "Iron-Man", "SpiderMan"]

avengers.push("AntMan", "Black Widow")

console.log(avengers);

// Imprime ["Hulk", "Thor", "Iron-Man", "SpiderMan", "AntMan", "Black Widow"]

**Método Pop en Array:**

Para eliminar elementos de un array, usamos el método pop, que elimina el último elemento:

var avengers = ["Hulk", "Thor", "Iron-Man", "SpiderMan", "AntMan"];

avengers.pop();

console.log(avengers);

// Imprime ["Hulk", "Thor", "Iron-Man", "SpiderMan"]

**Reverse** invierte el orden del array

Avengers.reverse();

**Shift** es como el pop pero extrae el primer elemento, no el último

Avengers.shift();

**Sort** para ordenar el array, si es de string, lo hace alfabéticamente, número de izq a der, de menor a mayor.

**Splice** elimina cualquier elemento del array, utiliza 2 parámetros, el primero es desde dónde voy a eliminar y el segundo cuántos elemntos quiero eliminar:

Avengers.splice(2,2)quitaría Ironman y Spiderman

Si hay un mas parámetros son los que sustituyen a los eliminados.

**Unshift** puedo añadir elementos, varios o uno pero al principio.

**Métodos que no modifican la matriz**

**Concat** concatena los 2 arrays, sus elementos

Console.log(moderatII.concat(moderat));

**Join** separa los elementos del array por medio del signo que utilice para ello, al final al concatenar obtenemos un string:

ModeratII.join(‘-’);

Console.log(moderatII);

**Slice** extrae una copia de una sección especificada del array, pero no la modifica como el splice, igual que el splice:

Avengers.splice(2,2)muestra Ironman y Spiderman

**Tostring** parecido al Join pero sin parámetros, transforma el array a string directamente:

moderatII.toString();

**Localizar un valor en un array**

**Indexof**

Devuelve el índice del primer elemento que coincide con el parámetro proporcionado

Avengers.indexOf(“Thor”)devuelve 1

Si no da coincidencia, devuelve -1

Importante cuando queremos hacer un Splice, primero localizamos el índice con éste y luego usamos el splice.

**Lastindexof**

Cuando un elemento se repite en un array, devuelve el índice de la última repetición.

Útil cuando se comprueba la no existencia de un elemento en un array.

**Includes**

Devuelve el valor booleano si el elemento que se busca existe o no en el array

Avengers.includes(“Thor”)devuelve true