Statementes:

* **Conditionals**

If

switch

* **Looping**

While

For

Do while

Break

continue

* Declarations

Var

* Function

WHILE

While(expr) statm

DO WHILE

Al menos ejecuta una vez.

FOR

For (ini\_stmt; cond\_expr ; post\_expr) stmt;

Todas las partes del for que estan dentro de los parentesis son opcionales.

Conviene usar el for cuando hay algo asociado que debo incrementar o derementar. Tratar de mantener el for lo mas simple posible, por ejemplo si tengo mas de una condición es preferible usar un while, por ejemplo

Var i=0;

While(

I<10

|| i \*2 === 20

) {

Console.log(i);

I++

}

No recomendable

For (i=0; i< 10 || i\*2 === 20; i++) {

Console.log(i)

I ++;

}

Break, sale abruptamente del ciclo.

Continue, no se ejecutan las líneas que están debajo, sigue con el ciclo siguiente, con el ejemplo de abajo por ejemplo nunca saldría del ciclo, ya que nunca se incremente i porque la condición del if es true.

Var 1= 0;

While(i<10) {

If (i \* 2=== 10) {

Continue;

}

Console.log(i);

I++;

}

Las variables utilizadas como contadores siempre se inicializan en 0, ya que si incremento una variable sin valor da undefined.

**SWITCH**

switch(expr) {

case expr1:

stmtA;

break;

case expr2:

stmtB;

break;

// ...

default:

stmtC;

// opcional

break;

}

Funciones string:

**Str.length** – cuenta cantidad de caracteres.

**Str.indexOf()** – busca la cadena de texto que le pongo adentro y devuelve la posición donde esta esa cadena o -1 si no la encuentra, por ejemplo para buscar las vocales:

Var str= ‘ este es mi string’

Str.infexOf(‘te’);

Devuelve 2, ya que ahí comienza la cadena evaluada.

**Str.lastindexOf()**; devuelve la posición de la cadena pero comenzando desde atrás hacia adelante, muestra la posición de la primera que encuentra.

Str.lastIndexOf(‘es’) Devuelve 5

**Str.toUpperCase()** - convierte string a mayusculas.

**Str.toLowerCase()** – convierte string a minusculas.

**Str.substring(index A, index B)** – se queda con una parte del string original especificando desde donde y hasta donde. Si no se especifica hasta donde cuenta hasta el final. Ej:

**Str**.substring(11, 17) devuelve ‘ string’

**Str.substr()** – se especifica desde que posición y con cuantos caracteres me quedo, a su vez puedo poner índices negativo, por ejemplo si quiero quedarme con las ultimas tres letras puedo usar -3 ej:

**Str**.substr(-2) devuelve ‘ve’ osea las ultimas dos letras.

**Str**.substr(-2, 2) devuelve ‘v’ arranque desde atrás con dos lugares y desde ahí me quede con un carácter.

**ARRAYS**

En un mismo array puedo tener elementos de distintos tipos, aunque no es recomendable.

Generacion de arrays:

Var arr= []; array vacio

Var arr = [‘test’,’hola’] array con dos string

Var arr= [2]; array con un numero 2;

Agregar elementos a un arary:

Push(), agrega elementos al final del array.

Arr.push(‘elem3’);

En un array puedo modificar elementos, osea alguno de sus componentes, a diferencia de un string donde no puedo cambiar uno de sus componentes.

Los array con huecos se llaman arrays dispersos.

Para eliminar elementos de un array, el ultimo elemento uso la función pop, es el inverso de push, ambos métodos modifican el array, no es que crean otro array.

Arr.pop();

Para quitar un elemento del array, pero desde el inicio del array utilizo shift()

Arr.shift();

Para agregar un elemento del array al principio utilizo unshift()

Arr.unshift();

Los array permiten indexOf() y lastIndexOf() de la misma manera que un string, pudiendo identificar cualquier tipo de elemento, no solo strings.

El operador in nos dice si un índice esta definido o no en un array, por ejemplo

Var arr = new Array(9) – array con 9 huecos sin contenido.

1 in arr devuelve false porque en la posición 1 hay un hueco, es una manera de comprobar si hay algo