**Let y Var:**

Let: variables ligadas al bloque en el que aparece. no se pueden volver a declarar en el interior del mismo alcance, sino solamente en bloques diferentes.

Var: puede ser de nuevo declarada en el interior del mismo alcance. En este caso el primer valor será simplemente reemplazado.

**Operadores lógicos**

(true && true); // true

(true && false); // false

(false && true); // false

(false && false); // false

**Condicionales:**

If…else

If…else if…

Switch case

**Ternario**

let result = condition ? value1 : value2;

**Métodos de String**

* **length**: permite conocer la longitud de un string, es decir, el número de caracteres que lo conforman. *console.log(country.length).*
* **includes:** permite conocer si un caracter o una porción de caracteres se encuentran dentro de un string, devolviendo un valor booleano según se cumpla o no. *console.log(quote.includes(word))*
* **replace:** devuelve un string con el string insertado por argumento sustituido por el segundo argumento. *movie.replace("Trek", "Wars")*
* **slice:** devuelve una nueva cadena con la porción delimitada entre la posición del primer argumento y la posición del segundo argumento.

*const album = "Master of Puppets";*

*console.log(album.slice(10, 13)); // Retorna "Pup"*

* **Split:** genera un array de tantos elementos como se indique en el segundo argumento indicándole el elemento separador en el primer argumento.

*const phrase = "Buenos días, ¿cómo estás?";*

*console.log(phrase.split(" ", 3)); // Retorna ["Buenos","días",",¿cómo" ]*

*Para convertir un array en un string, por ejemplo.*

* **toLowerCase/toUpperCase**: devuelve el string en min o may. name.toUpperCase()
* **trim**: elimina los espacios en blanco.

**Métodos de Number:**

**isNaN,** si es un número o no (true o false). **isInteger** número entero o no

**parseInt**: de string.a entero; **parseFloat**: de string a decimal. **toString**: a número con la base indicada.

**Función**: function name (parámetros) {acciones de la función; return}

**Arrow**: const name = (parámetros) => {acciones de la función; return}

* No tienen la palabra clave this.
* No se pueden usar como constructores.
* No tienen una palabra clave arguments.
* No se pueden usar como métodos.

**Bucles**:

For (let i=0; i<=array.length; i++) { let element=array[] }

Arrays:

forEach (solo arrays); for of (objeto iterable, devuelve los valores); for in (cualquier objeto, devuelve claves)

studentList.**forEach**(element => console.log(element))

**for** (var word **of** name) { console.log(word); } // **Array**, **Map**, **String**

**Array.**

**Acceso:**

Array = arrays[0] (posición)

Pop(): elimina el último elemento.

Shift(): elimina el primer elemento.

Push(): añade un elemento en la última posición.

Reverse(): invierte el orden del array.

Sort(): ordena por orden alfabético.

Splice(). Param1-índice para empezar a modificar el array. Param2- n elementos eliminar/sustituir. 0=nada.

2 parámetros: elimina el n elementos de param2 desde param1.

1 parámetro: rellena los huecos y elimina los que faltan.

+ elementos: elimina los indicados e inserta los elementos desde el anterior y el siguiente a la sección eliminada. No devuelve la parte secciona sino la nueva longitud.

Unshift(): los elementos se añaden al principio de la matriz y devuelve la nueva longitud.

**Métodos que no modifican**:

Concat(), concatena arrays, o elementos separados por comas, no altera el array.

Join(): devuelve valores del array separados por comas, o especificar por parámetro el separador.

Slice(): extrae una copia de una sección de un array pero no lo modifica. 1 param: índice empezar copia hasta el final. 2 param: índice donde comenzar (incluido) e índice donde terminar (no incluido).

indexOf(): devuelve el índice del primer elemento que coincide con el parámetro o -1 si no lo hay.

lastIndexOf(): devuelve el índice del último elemento que coincide con el parámetro o -1.

Includes(): si el array incluye un elemento: true o false.

**Objetos.** Colección de pares clave-valor.

Obj = {}

Acceso propiedades:

Obj.propiedad

Obj[propiedad]

myObject[propiedad]

objet.keys(0): devuelve array con las propiedades (claves) propias del elemento 0.

Función constructora: fuction Car(model, year); this.model=model; this.year=year;

This para asignar valores a las propiedades del objeto en función de los valores pasados a la función.

This: dentro de un método para referiste al objeto actual.

Obj.entries(): genera array cuyos elementos son arrays correspondientes a las claves y valores que tenga cada propiedad del objeto.

Object.entries(objeto): clave y valor.

**JSON**: objeto que representa una serie de pares clave-valor separados por comas.

JSON.parse(): convierte un JSON en un objeto.

JSON.stringify(): convierte un objeto o valor de JS en un string JSON. Para almacenar datos o pintar info.

**Hoisting**: permite declarar variables y funciones al comienzo del ámbito en el que estemos. Esto hace que aunque se declare una variable o función en cualquier parte del código, JS la tratará como si estuviera principio del ámbito en el que se ha declarado. Permite usar una función antes de declararla.

**Destructuring**: permite extraer datos de arrays u objetos en variables. Útil cuando queremos acceder a los valores fácilmente.

**Spread**: expandir o copiar array u objetos. Copiar, concatenar.