**Metodo para saber si existe una cadena de texto:**

.includes( ) - te marca false porque tablet no existe en la cadena de texto

**Metodos para eliminar el espacio en blanco al inicio y al final de una cadena de texto:**

.trimStart( ) – Elimina espacios en blanco del inicio

.trimEnd( ) - Elimina espacios en blanco del final

.trim( ) – Elimina espacios en blanco al principio y al final

**Metodos para reemplazar el texto de una cadena de texto**

.slice(0, 10) - Le indicas la posicion que quieres que corte

.slice(8) - Lo que hace esto es ur de la posicion 8 al final si es que no le pasas el segundo valor

slice(2, 1) - Si el primer numero es mayor que el primero, entonces este codigo no va a hacer nada

.substring(0,10) - Se comporta igual que el slice

charAt(0) - Forma para obtener la primer letra mucho mas facil, pero aqui solamente le indicas cual quieres

**Metodos para repetir una cadena de texto:**

.repeat(3) - Si intentas engañarlo y decir que quieres que se repita 2.4 veces, js lo redondea

**Metodos para dividir una cadena de texto:**

.split(" ") - Le pasas que v a abuscar en ese string para dividirlo y crear diferentes partes de ese string

**Metodos string para convertir a mayusculas o minúsculas**

.toUpperCase( ) - Transforma todo a mayusculas

.toLowerCase( ) - Transforma todo a minúsculas

**Metodo string para convertir un numero a un string**

.toString( ) - Convertir un numero a un string

**Metodos math**

resultado = Math.PI; - PI

//Redondear

resultado = Math.round(2.8); - redondea hacia arriba

resultado = Math.round(2.2); - redondea hacia abajo

//Redondear hacia arriba

resultado = Math.ceil(2.1);

//Redondea hacia abajo

resultado = Math.floor(2.9);

//Raiz cuadrada

resultado = Math.sqrt(144);

//Valor absoluto (ignora el menos)

resultado = Math.abs(-500);

//Potencia

resultado = Math.pow(8, 3);

//Minimo

resultado = Math.min(3,4,5,6);

//Maximo

resultado = Math.max(3,4,5,6);

//Aleatorio

resultado = Math.random();

//Aleatorio mezclado

resultado = Math.random() \* 20;

//Aleatorio dentro de un rango

resultado = Math.floor(Math.random() \* 30); //Aleatorios del 0 al 30

**Metodos para convertir strings a números**

Number.parseInt(numero1) - Esto sirve para convertir un string a numero, ademas de eso tambien lo convierte a entero

Number.parseFloat(numero2) - Esto sirve para convertir un string a numero flotante

Number.isInteger(numero4) - Revisar si un numero es entero o no

isNaN( ) – Intenta convertir lo que ingreso a numero, si no puede regresa ‘true’, si puede regresa ‘false’

Number( ) – convierte un string a numero

**Metodo para prevenir que un objeto sea modificado:**

Object.freeze(producto)

**Metodos saber que un objeto este congelado:**

Object.isFrozen(producto)

**Metodo para sellar un objeto:**

Object.seal(producto);

**Metodo para copiar 2 objetos:**Object.assign(objeto1,objeto2)

**Metodo para saber las llaves y credenciales de un objeto**

Object.keys(producto) - Retorna arreglo con los keys del objeto, tambien se uiliza para saber si un objeto esta vacio

Object.values(producto) - Retorna los valores que tienen las propiedades de los objetos

Object.entries(producto) - Este te retorna todo en pares

**Metodo para** **agregar en un arreglo**

.push( ) - Agregar al final del arreglo

.unshift( ) - Colocar al inicio del arreglo

**Metodo para eliminar en un arreglo**

.pop( ) - Eliminar el utlimo elemento de un arreglo

.shift( ) - Eliminar del inicio del arreglo

carrito.splice(1,1) - Elminar de en medio (Toma 2 parametros) :

//El primero es de donde quieres empezar a cortar

//El segundo cuantos quieres elimiar a partir de ahí

**Metodos para iterar un array**

.forEach( (pendiente, index) =>{ }); - pasa automaticamende el índice

Como el forEach esta disenado para recorrer arreglos, le puedes poner un índice

.map( (producto) =>{ return producto.nombre; }) - La unica diferencia es que map crea un arreglo nuevo

**Metodos para saber si un valor existe en un objeto**

.includes( ) - Te dice si un valor existe, el resultado se guarda en la variable, se guarda como si 'true' o 'false' dependiendo de lo que sea.

// En un arreglo de objetos se utiliza .some :

const existe = carrito.some((producto) => {

return producto.nombre === 'Celular'

});

// En un arreglo tradicional:

const existe2 = meses.some((mes => {

return mes === 'Enero'

}));

**Metodos para encontrar la posición en un array**

// Encontrar el indice con .findIndex :

const indice = meses.findIndex((mes) =>{

return mes === 'Abril'

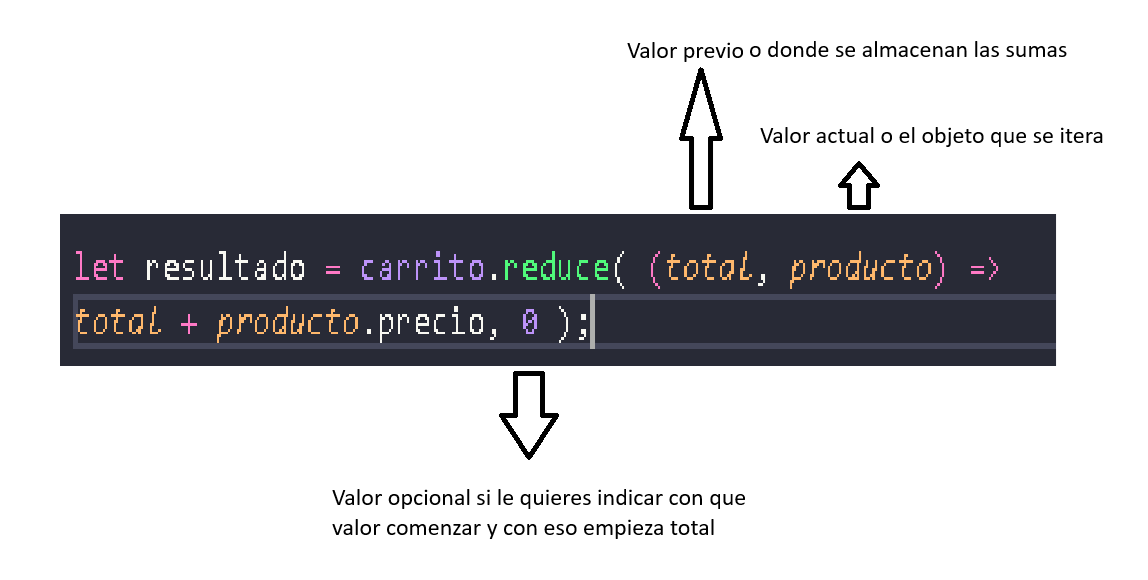
}); - Si lo encuentra te regresa la posicion en donde lo encontro, si no lo encuentra te regresa un -1, Si hay 2 posibles resultados, solamente regresa el indice del primero que encuentra

**Metodo para tomar una gran cantidad de datos, unirlos, y entregarte un resultado**

.reduce:

//Se le pasan 3 valores: 1. el valor anterior. 2. Se pasa el valor actual. 3. Le puedes poner el valor inicial

let resultado = carrito.reduce( (total, producto) => total + producto.precio, 0 ); //Solamente se utiliza el signo "+", porque como se va acumulando automaticamente en la variable "total" que definimos en los parametros, no es necesario colocar "+=". (Ya sabe el valor previo)



**Metodos para crear un nuevo arreglo basado en el parametro que es evaluado**

.filter( ) – Crea un arreglo con todos las las variables que cumplan esa condicion

resultado = carrito.filter((*producto*) =>{

    return *producto*.precio > 400

})

Quitar un producto del arreglo

resultado3 = carrito.filter((*producto*) => {

    return *producto*.nombre !== 'Audifonos' //Treme todos los que no sean los audifonos

})

.find( ) – Crea un arreglo únicamente con la primera que cumple esa condicion

//Con .find, solamente retorna el primer elemento que cumpla con la condicion

const resultado2 = carrito.find((*producto*)=>{

    return *producto*.nombre === 'Tablet'

});

**Metodos para saber si los esa valores cumplen esa condicion para que nos regrese un true**

.every( ) - Necesita que todos cumplan para regresa ‘true’

// Todos los elementos de un arreglo deben de cumplir esa condicion para que nos regrese un true

// Todos los elementos deben de cumplir las condiciones, para que devuelva true

const resultado = carrito.every( (*producto*) =>{

    return *producto*.precio < 1000;

})

.some( ) – solo necesita que cumpla 1 para regresar un ‘true’

//Solo se ocupa cumplir 1 para que de true

const resultado2 = carrito.some( (*producto*) =>{

    return *producto*.precio < 1000;

})

**Metodo para juntar 2 arreglos en 1**

.concat( ) – Juntar 2 arreglos

// Se coloca el que va a ir primero y despues se concatena el siguiente

const resultado = meses.concat(meses2,meses3, 'otro mes');