https://vimeo.com/581183469/b9be8f71d2

// No cambies los nombres de las funciones.

function deObjetoAmatriz(objeto){

  // Escribe una función que convierta un objeto en una matriz, donde cada elemento representa

  // un par clave-valor en forma de matriz.

  //Ejemplo:

  /\*objeto({

      D: 1,

      B: 2,

      C: 3

    }) ➞ [["D", 1], ["B", 2], ["C", 3]]\*/

  //Escribe tu código aquí objects.entries pódria ser? <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Object/entries>

  return Object.entries(objeto); // devuelve una matriz con cada key y value del objeto encerrado en un subarray

Con objetct.key hacemos un array con las keys del objeto. Y a ese array le hacemos .map, a través de la cual mapeamos un nuevo array para el cual pasamos una funcion con la cual devolvemos en cada key y el valor de la key correspondiente del objeto.

 return Object.keys(objeto).map(function(key) {

   return [key, objeto[key]]});

 var matriz = [];

 for (key in objeto) {

   matriz.push([key, objeto[key]]); //va un [] adentro porque es un subarray para cada par.

 }

return matriz;

}

}

function numberOfCharacters(string) {

  //La función recibe un string. Recorre el string y devuelve el caracter con el número de veces que aparece

  //en formato par clave-valor.

  //Ej: Recibe ---> "adsjfdsfsfjsdjfhacabcsbajda" || Devuelve ---> { a: 5, b: 2, c: 2, d: 4, f: 4, h:1, j: 4, s: 5 }

  //Escribe tu código aquí

 let final = {}

  for (let i in string) {

 if (string[i] in final) {

   final[string[i]] = final[string[i]] + 1 // (de este modo se asigna valor a una key

 }

 else  final[string[i]] = 1;

 }

return final;

}

}

Primero creamos una variable que será el objeto final. Luego iteramos la variable i en el string. Si la letra con ese índice está en el objeto final como propiedad(cosa que no ocurrirá en el primer caso) le sumaremos al valor que ya tiene +1 (es para sumar valor a las letras que ya aparecieron).

Para las letras del índice iterado que no están aun en el objeto final creamos un Else para que se le de valor de 1 a esa propiedad (se toma a cada letra del string como una propiedad del objeto).

Al final se retorna el objeto final.

let object = {};

for (let i = 0; i < string.length; i++) {

  if (Object.keys(object).includes(string[i]))

  object[string[i]] = object[string[i]] + 1; // le damos valor a la propiedad de + 1 si ya aparece como propiedad en la variable que creamos para acumular los resultados

else object[string[i]] = 1; // le damos valor a la propiedad de 1 cuando aun no ha aparecido en el objeto final.

}

return object;

}

function capToFront(s) {

  //Realiza una función que reciba como parámetro un string y mueva todas las letras mayúsculas

  //al principio de la palabra.

  //Ejemplo: soyHENRY -> HENRYsoy // += va sumando(operador de suma)

  //Escribe tu código aquí

let stringmayalppio = '';

let stringminalfinal = '';

for (let i = 0; i < s.length; i++) {

  if (s[i].toUpperCase() === s[i]) {

    stringmayalppio += s[i]; // += es concatenar o sumar

  }

  else if (s[i].toLowerCase()===s[i]) {

  stringminalfinal += s[i]; (porque no anda el.push?)

}

}

return stringmayalppio + stringminalfinal;

// return stringmayalppio.concat('', stringminalfinal) //tmb se puede retornar asi

O asi  //return ''.concat(stringmayalppio, stringminalfinal)

Concat debe aplicarse a un string

}

}

function asAmirror(str) {

  //La función recibe una frase.

  //Escribe una función que tome la frase recibida y la devuelva de modo tal que se pueda leer de izquierda a derecha

  //pero con cada una de sus palabras invertidas, como si fuera un espejo.

  //Ej: Recibe ---> "The Henry Challenge is close!" || Devuelve ---> "ehT yrneH egnellahC si !esolc"

  //Escribe tu código aquí // split(' ') corta cuando encuentra el argumento

var frasealreves = str.split(' ').map(function (element) {

return  element.split('').reverse().join('')}).join(' ');

return frasealreves;

}

Con Split cortamos cada palabra de la frase str… eso lo mapeamos y a cada elemento, que van a ser palabras, le aplicamos una funcion para darlos vuelta.

Para poder usar el .reverse en esa función con cada elemento separado primero hay que splitear cada elemento sin espacios(‘’), luego hacerle un reverse para dar vuelta cada letra, luego un join(‘’) sin espacio para unir cada letra en la palabra correspondiente (sin espacios) y finalmente un join general con (‘ ‘) para unir cada palabra con espacios en el medio

function capicua(numero){

  //Escribe una función, la cual recibe un número y determina si es o no capicúa.

  //La misma debe retornar: "Es capicua" si el número se número que se lee igual de

  //izquierda a derecha que de derecha a izquierda. Caso contrario retorna "No es capicua"

  //Escribe tu código aquí

if (numero.toString().split('').reverse().join('') == numero)

  return 'Es capicua';

  else return 'No es capicua';

}

}

// de este modo la comparación funciona entre un string y el numero porque comparo con == osea no igualdad estricta (da igual si es numero o string asi no evalua el tipo de dato)

de otro modo:

  var numero = numero.toString();

  var numeroinverso;

  numeroinverso = numero.split('').reverse().join('');

  if (numero === numeroinverso) {

  return ('Es capicua'); }

  else {

    return ('No es capicua');

  }

  }

Para hacer una funcion mas ”limpia” convertimos el numero pasado como argumento a string… asi podemos comparar estrictamente (con ===) el numero del argumento pasado a string y el string que obtuvimos al dar vuelta el argumento (que lo hicimos con métodos de string)

}

function deleteAbc(cadena){

  //Define una función que elimine las letras "a", "b" y "c" de la cadena dada

  //y devuelva la versión modificada o la misma cadena, en caso de contener dichas letras.

  //Escribe tu código aquí

  newWord = cadena.replace('a','').replace('b','').replace('c','');

  return newWord;

  }

Creamos una nueva variable y reemplazamos en la cadena cada letra que no deseamos por un espacio vacio. Lo hice 3 veces para cada letra porque no supe como economizar eso

}

.replace crea un string nuevo con el reemplazo descripto

function sortArray(arr) {

  //La función recibe una matriz de strings. Ordena la matriz en orden creciente de longitudes de cadena

  //Ej: Recibe ---> ["You", "are", "beautiful", "looking"] || Devuelve ---> [“You", "are", "looking", "beautiful"]

  //Escribe tu código aquí

  return arr.sort(function (a, b) {

    return a.length - b.length});

}

O si no

  return arr.sort((a, b) => a.length - b.length);

}

}

El .sort normalmente ordena por UTF-16 code units values. Por eso debemos pasarle un argumento para que ordene cada elemento según una función en la cual retornamos la resta del .length del elemento a y el b. De este modo quedara primero en el array el elemento que menos cantidad de letras tenga.

Si esa resta da un negativo es porque a tiene mas que b, entonces mueve a b primero; si por el contrario da positivo es porque a es mas grande y lo deja como esta, si da 0 deja a a, porque son iguales.

function buscoInterseccion(arreglo1, arreglo2){

  //Existen dos arrays, cada uno con 5 números. A partir de ello, escribir una función que permita

  //retornar un nuevo array con la intersección de ambos elementos. (Ej: [4,2,3] unión [1,3,4] = [3,4].

  //Si no tienen elementos en común, retornar un arreglo vacío.

  //Aclaración: los arreglos no necesariamente tienen la misma longitud

  //Escribe tu código aquí

  var interseccion = [];

  for (let i = 0; i < arreglo1.length; i++) {

    if (arreglo2.includes(arreglo1[i]))

   interseccion.push(arreglo1[i]);

}

return interseccion;

}

Iteras el arreglo1 y después das la condición de que si en alguna de las iteraciones un elemento del arreglo1 es incluido también por el arreglo 2, lo pusheamos al nuevo array intersección.

Otra forma de hacerlo es con dos “bucles anidados” … un for dentro de un for

 var interseccion = [];

  for (let i = 0; i < arreglo1.length; i++) {

    for (let j = 0; j < arreglo2.length; j++) {

      if (arreglo1[i] === arreglo2[j])

      interseccion.push(arreglo1[i]);

}

}

return interseccion;

}

Esta forma es interesante porque se usa un bucle anidado, es decir abrimos el for para arreglo 1 y dentro de la funcion que vamos a declarar luego esta el for para el arreglo2. De este mod, cada elemento que tomamos del arreglo1 lo vamos a comparar con todos los elementos del arreglo2 y pusheamos los que sean iguales a la variable final. Luego retornamos la interseccion