Henry challenge

function sumaTodosImpares(array) {

  // La funcion llamada 'sumaTodosImpares' recibe como argumento un array de enteros.

  // y debe devolver la suma total entre todos los numeros impares.

  // ej:

  // sumaTodosImpares([1, 5, 2, 9, 6, 4]) devuelve 1 + 5 + 9 = 15

  // Tu código aca:

var sumanumerosimpares= 0;

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

if (array[i] % 2 !== 0) {

sumanumerosimpares += array[i];

}

}

return sumanumerosimpares;

}

Creamos la variable sumanúmerosimpares. Iteramos el array y le damos la condición de que si el resto del elemento[i] es distinto de 0(porque es impar), lo sumamos a la variable suma.

function stringMasLarga(str) {

  // La función llamada 'stringMasLarga', recibe como argumento un frase (string) 'str'

  // y debe devolver la palabra (string) más larga que haya en esa frase (Es decir el de mayor cantidad de caracteres)

  // Ej:

  // stringMasLarga('Ayer fui a pasear a una plaza') debe retornar 'pasear'

  // stringMasLarga('Me gusta mucho javascript') debe retornar 'javascript'

  // Tip: podes usar el metodo de String 'split'

  // Tu código aca:

var palabramaslarga= '';

str.split(' ').forEach(function(palabra){

  if (palabra.length > palabramaslarga.length){

       palabramaslarga = palabra

   };

  });

  return palabramaslarga

}

Creamos la variable palabra más larga como un string vacio. Spliteamos con un espacio para dividir la frase en un array de palabras y co un for each a cada una de esas palabras la comparamos con el .length de nuestra variable creada al ppio. Si es mayor… la palabra pasa a convertirse en nuestra variable.

function estaOffline(usuarios, nombre) {

  // La funcion llamada "estaOffline" recibe como argumento un array de objetos llamado 'usuarios' y un string llamada 'nombre'.

  // cada objeto tiene una property 'nombre' que es un string y otra llamada 'online' que es un booleano.

  // La función debe retornar true si el usuario se encuentra offline, de lo contrario false.

  // ej:

  // var usuarios = [

  //     {

  //       nombre: 'toni',

  //       online: true

  //     },

  //     {

  //       nombre: 'emi',

  //       online: true

  //     },

  //     {

  //       nombre: 'agus',

  //       online: false

  //     }

  // ];

  // estaOffline(usuarios, 'agus') retorna true

  // estaOffline(usuarios, 'emi') retorna false

  // Tu código aca:

var offline = undefined;

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

   if (usuarios[i].nombre === nombre) {

     offline = !usuarios[i].online;

   }

}

return offline;

}

Creamos la variable offline que va a retornar nuestra funcion. Iteramos el array usuarios que es donde se encuentran nuestros objetos. Condicionamos que si el nombre del objeto llamado como argumento es el nombre del objeto iterado, le damos a la variable offline lo opuesto (por eso ‘!’) a lo que contiene esa propiedad online del objeto como valor

function actividadesEnComun(persona1, persona2) {

  // La funcion llamada 'actividadesEnComun' recibe como argumento dos arrays de actividades (strings) llamados 'persona1' y 'persona2'

  // y debe devolver un array de strings con las actividades en comun ( aquellas que se repiten ) entre cada array.

  // ej: persona1 = ['leer', 'comer', 'pasear', 'dormir', 'jugar']

  //     persona2 = ['comer', 'dormir', 'futbol']

  // actividadesEnComun(persona1, persona2) => ['comer', 'dormir']

  // Tip: podes usar ciclos for anidados.

  // Tu código aca:

  var newarray = [];

  for (var i = 0; i < persona1.length; i++) {

  for (var j = 0; j < persona2.length; j++) {

  if(persona1[i] === persona2[j]){

    newarray.push(persona2[j]);

}

}

}

return newarray;

}

Creamos la variable para el array que vamos a retornar. Iteramos el primer string del argumento y lo anidamos al otro string. Cada vez que iteremos numero1[i] lo va a comparar con el primer elemento del otro arreglo y asi pusheamos a la variable inicial lo que se repita

function buscaDestruye(arreglo, num) {

  // La función 'buscaDestruye' recibe como argumento un array de enteros 'arreglo' y un entero 'num'.

  // Esta función tiene que eliminar los numeros del array que coincidan el numero recibido como argumento.

  // La función debe retornar el array sin los números sacados.

  // Ej: buscaDestruye([1, 2, 3, 4], 2) devuelve => [1, 3, 4]

  // Nota: Si el número se repite más de una vez, también hay que eliminarlo.

  // Ej: buscaDestruye([1, 2, 3, 4, 1], 1) devuelve => [2, 3, 4]

  //

  // Tu código aca:

  var newarray = [];

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

  if (arreglo[i] !== num) {

  newarray.push(arreglo[i])

  }

  }

return newarray;

}

Creamos una variable array inicial que luego retornaremos. Iteramos el arreglo pasado como argumento y si alguno de sus elementos es distinto del numero pasado como segundo argumento lo pusheamos en la variable a retornar.

function sumarElTipo(arreglo) {

  // La funcion llamada 'sumarElTipo' recibe un array de strings como argumento

  // que contiene tipos de vehiculos y debe devolver un objeto con la cantidad

  // de veces que se repita cada tipo.

  // El objeto que devuelve tiene como propiedades el nombre de cada vehiculo y su valor es la cantidad de veces que se repite.

  // Ej:

  // sumarElTipo(['auto', 'moto', 'auto']); debe retornar {auto: 2, moto: 1}

  // Tip: podes usar el ciclo for o el metodo de Array 'reduce'

  // Tu código aca:

  var objeto = {};

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

    if (arreglo[i] in objeto) {

      objeto[arreglo[i]] =  objeto[arreglo[i]] + 1;}

else objeto[arreglo[i]] = 1;

  }

  return objeto;

}

Creamos la variable que retornaremos que es un objeto. Iteramos el arreglo y para el mismo planteamos la condición de que si el elemento [i] del array esta en la variable objeto que creamos, le daremos el valor que tiene +1. Si no tiene ningún valor, le asignaremos el valor 1

// =======================================================================

function crearClaseEmprendedor() {

  class Emprendedor {

      constructor(nombre, apellido, libros, mascotas) {

          // El constructor de la clase Emprendedor recibe nombre (string), apellido (string), libros (array de objetos), mascotas (array de strings)

          // Inicializar las propiedades del emprendedor con los valores recibidos como argumento

          // Tu código aca:

          this.nombre = nombre;

          this.apellido = apellido;

          this.libros = libros;

          this.mascotas= mascotas;

      }

      addMascota(mascota) {

        // este método debe agregar una mascota (mascota) al arreglo de mascotas del emprendedor.

        // no debe retornar nada.

        // Tu código aca:

this.mascotas.push(mascota);

      }

      getMascotas() {

          // El método 'getMascotas' debe retornar la cantidad de mascotas que tiene el emprendedor.

          // Ej:

          // Suponiendo que el emprendedor tiene estas mascotas: ['perro', 'gato']

          // emprendedor.getMascotas() debería devolver 2

          // Tu código aca:

return this.mascotas.length;

      }

      addBook(book, autor) {

          // El método 'addBook' recibe un string 'book' y un string 'autor' y debe agregar un objeto:

          // { nombre: book, autor: autor} al arreglo de libros del emprendedor.

          // No debe retornar nada.

          // Tu código aca:

this.libros.push({nombre: book, autor: autor});

      }

      getBooks() {

          // El método 'getBooks' debe retornar un arreglo con sólo los nombres del arreglo de libros del emprendedor.

          // Ej:

          // Suponiendo que el emprendedor tiene estos libros: [{nombre: 'El señor de las moscas',autor: 'William Golding'}, {nombre: 'Fundacion', autor: 'Isaac Asimov'}]

          // emprendedor.getBooks() debería devolver ['El señor de las moscas', 'Fundacion']

          // Tu código aca:

          var newarray = [];

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

    newarray.push(this.libros[i]['nombre'])

  }

  return newarray;

      }

      getFullName() {

          // El metodo getFullName debe retornar un string con el nombre y apellido del emprendedor.

          // ej:

          // Suponiendo que el emprendedor tiene: nombre: 'Elon' y apellido: 'Musk'

          // emprendedor.getFullName() deberia devolver 'Elon Musk'

          // Tu código aca:

return (this.nombre + ' ' +  this.apellido);

      }

  }

  return Emprendedor;

}

/\* ====================== EXTRA CREDIT ===================== \*/

// Este ejercicio no cuenta en el puntaje del checkpoint

// pero si llegas hasta aca y lo haces quiere decir que venis super bien!

function mapear() {

  // Escribi una funcion mapear en el prototipo del objeto global 'Array'

  // que recibe una funcion callback , que se ejecuta por cada elemento del array

  // mapear los elementos de ese array segun la funcion callback

  // Esta funcion tiene que devolver un array nuevo con los elementos mapeados.

  // NO USAR LA FUNCION MAP DE ARRAYS.

  // ej:

  // var numeros = [1, 2, 3, 4];

  // numeros.mapear(function(numero) {

  //   return numero + 1;

  // }) devuelve [2, 3, 4, 5]

  // Tu código aca:

  Array.prototype.mapear = function(cb) {

    var newarray = [];

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

     newarray.push(cb(this[i]));

}

  return newarray;

}

}

Creamos la funcion mapear dentro del prototype de Array y le asignamos a esa funcion el argumento call back (cb)… dentro de esa funcion mapear (no afuera!) creamos la variable newarray que luego retornaremos. Iteramos this.length (porque aun no sabemos el nombre del objeto al que afectara lla funcion) y pusheamos en la variable newarray cada elemento, pasado por la funcion cb. Retornamos el newarray dentro de .mapear