Taller 3. - Métodos de Arreglo, Array Destructuring y Promesas

Tiempo 60 Minutos.

Recursos:
https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

1. Tiene un arreglo de objetos de usuario, cada uno tiene user.name. Escriba el código que lo convierte en un arreglo de nombres.

```
let john = { name: "John", age: 25 };
let pete = { name: "Pete", age: 30 };
let mary = { name: "Mary", age: 28 };

let users = [ john, pete, mary ];

let names = /* ... your code */

console.log( names ); // John, Pete, Mary
```

2. Tiene una arreglo de objetos de usuario, cada uno tiene nombre, apellido e identificación. Escriba el código para crear otro arreglo a partir de él, de objetos con id y fullName, donde fullName se genera a partir de nombre y apellido.

```
let john = { name: "John", surname: "Smith", id: 1 };
let pete = { name: "Pete", surname: "Hunt", id: 2 };
let mary = { name: "Mary", surname: "Key", id: 3 };

let users = [ john, pete, mary ];

let usersMapped = /* ... your code ... */

/* el arreglo resultado debe ser el siguiente
usersMapped = [
  { fullName: "John Smith", id: 1 },
  { fullName: "Pete Hunt", id: 2 },
  { fullName: "Mary Key", id: 3 }
]
*/

console.log( usersMapped[0].id ) // 1
console.log( usersMapped[0].fullName ) // John Smith
```

3. Escriba la función getAverageAge (usuarios) que recibe un arreglo de objetos con edad (age) como propiedad y devuelve la edad promedio.
La fórmula para el promedio es (age1 + age2 + ... + ageN) / N. Por ejemplo:

```
let john = { name: "John", age: 25 };
```

```
let pete = { name: "Pete", age: 30 };
let mary = { name: "Mary", age: 29 };

let arr = [ john, pete, mary ];

console.log( getAverageAge(arr) ); // (25 + 30 + 29) / 3 = 28
```

4. Sea arr sea un arreglo. Cree una función unique (arr) que debería devolver un arreglo con elementos únicos de arr. Por ejemplo:

```
function unique(arr) {
  /* your code */
}

let values = ["Hare", "Krishna", "Hare", "Krishna",
  "Krishna", "Krishna", "Hare", "Hare", ":-O"
];

console.log( unique(values) ); // Hare, Krishna, :-O
```

5. Teniendo el siguiente objeto:

```
let user = {
  name: "John",
  years: 30
};
```

Escriba la tarea de desestructuración para obtener:

La propiedad name en una variable name.
La propiedad age en una variable age.
La propiedad isAdmin en la variable isAdmin (falso, si no existe tal propiedad)

Aquí hay un ejemplo de los valores después de la tarea:

```
let user = { name: "John", years: 30 };

// your code to the left side:
// ... = user

console.log( name ); // John
console.log( age ); // 30
console.log( isAdmin ); // false
```

6. La fábrica de bicicletas. Dado el siguiente inventario:

```
let bicicletasEnsamblar=[
            {
                numSerie:1,
                partes: [
                            {
                                tipo:"llanta_trasera",
```

```
                                referencia:"Michelin R29"
                            },
                            {
                                tipo:"marco",
                                referencia:"Specialized Rock Hopper
"
                            },
                            {
                                tipo:"llanta_delantera",
                                referencia:"Michelin R29"
                            }
                        ]
            },
            {
                numSerie:2,
                partes: [
                            {
                                tipo:"llanta_trasera",
                                referencia:"Michelin R27"
                            },
                            {
                                tipo:"marco",
                                referencia:"Trek 125"
                            },
                            {
                                tipo:"llanta_delantera",
                                referencia:"Michelin R27"
                            }
                        ]
            }
        ];
```

Por favor implemente el código para ensamblar las bicicletas del inventario en un arreglo de objetos bicicleta como se observa a continuación:

```
[ { serie: 1,
    bici:
     { llanta_trasera: 'Michelin R29',
       marco: 'Specialized Rock Hopper',
       llanta_delantera: 'Michelin R29' } },
  { serie: 2,
    bici:
     { llanta_trasera: 'Michelin R27',
       marco: 'Trek 125',
       llanta_delantera: 'Michelin R27' } } ]
```

7. Eres desarrollador de una gran empresa, con múltiples bases de datos en todo el mundo. Su trabajo es reunir toda la información para un perfil de personal del usuario. Sin embargo, la información se distribuye en diferentes bases de datos y servicios.

   Su función toma una identificación simple (un número) y debe devolver una promesa con un objeto como datos. El objeto debe contener toda esta información:

   {

   id: un número,

   nombre de usuario: una cadena,

   país: una cadena,

   nombre: una cadena,

   apellido: una cadena,

   correo electrónico: una cadena

   }

   Para lograr su tarea, debe utilizar los diferentes servicios proporcionados:

   central: debido a la cantidad de usuarios, no podemos almacenarlos en una sola base de datos. Entonces tenemos 3 bases de datos. La base de datos central identifica en qué base de datos están almacenados los usuarios. Úselo así: central (id) .then (function (data) {...}). data es una cadena con 3 valores posibles: 'db1', 'db2' y 'db3'. Si la base de datos central tiene un error, su función debe devolver una promesa rechazada con 'Error central' en los datos.

   db1, db2 y db3 contienen la información básica para los usuarios. Úselo así: db1 (id) .then (function (data) {...}). data es un objeto que contiene 2 propiedades: nombre de usuario y país. Si una base de datos tiene un error, su función debe devolver una promesa rechazada con 'Error db1' (o 'Error db2' o 'Error db3') en los datos.

   bóveda: los datos personales están restringidos por ley. Este tipo particular de datos a menudo se almacena en una base de datos específica. La bóveda se puede utilizar para obtener información personal sobre un usuario. Úselo así: vault (id) .then (function (data) {...}). data es un objeto con 3 propiedades: nombre, apellido y correo electrónico. Si la bóveda tiene un error, su función debe devolver una promesa rechazada con 'Bóveda de error' en los datos.

   marca: cada vez que alguien lee un perfil de usuario, debemos marcarlo. El servicio de marca creará esta marca. Úselo así: mark (id) .then (function () {...}). No llame al servicio de marca si hay otro servicio por error. Además, no espere a que el servicio de marca complete el procesamiento. Si el servicio de marca tiene un error, no haga nada específico. Simplemente devuelva su promesa con la información.

   Todos los servicios responden en 100 ms, excepto la bóveda. (La seguridad de los datos personales es más pesada, por lo que es más lenta. Responderá en 150 ms). Su

código debe responder en máximo 300 ms. Si hay varios servicios por error, devuelva el primer error que encontró.