Formación

Ejercicios JavaScript

Ejercicio 1

Dado un array de objetos, obtener el objeto con el id 3

```
const arrNames = [
  {id: 1, name: 'Pepe'},
  {id: 2, name: 'Juan'},
  {id: 3, name: 'Alba'},
  {id: 4, name: 'Toby'},
  {id: 5, name: 'Lala'}
]
```

```
const arrNames = [
  {id: 1, name: 'Pepe'},
  {id: 2, name: 'Juan'},
  {id: 3, name: 'Alba'},
  {id: 4, name: 'Toby'},
  {id: 5, name: 'Lala'}
]

const filterList = arrNames.filter(arrNames =>{
  return arrNames.id ==3;
}).map(arrNames =>({
  id: arrNames["id"],
  name: arrNames["name"]
}))

console.log(JSON.stringify(filterList))
```

```
"[{\"id\":3,\"name\":\"Alba\"}]"
>
```

Ejercicio 2

Dado un array de valores, devolver un array truthy (sin valores nulos, vacíos, no números, indefinidos o falsos)

```
const arrDirty = [NaN, 0, 5, false, -1, '',undefined, 3, null, 'test']
```

```
const arrDirty = [NaN, 0, 5, false, -1, '',undefined, 3, null, 'test'];

function arrClear1(arr){
  var myFilterArray = arr.filter(Boolean);
  return myFilterArray;
}
const solution1 = arrClear1(arrDirty);
console.log(solution1);
```

```
[5, -1, 3, "test"]
```

Ejercicio 3

Dado un array de ciudades, sacar todas las ciudades de España que no sean capitales

```
const arrCities = [
  {city: 'Logroño', country: 'Spain', capital: false},
  {city: 'Paris', country: 'France', capital: true},
  {city: 'Madrid', country: 'Spain', capital: true},
  {city: 'Rome', country: 'Italy', capital: true},
  {city: 'Oslo', country: 'Norway', capital: true},
  {city: 'Jaén', country: 'Spain', capital: false}
]
```

```
const arrCities = [
  {city: 'Logroño', country: 'Spain', capital: false},
  {city: 'Paris', country: 'France', capital: true},
  {city: 'Madrid', country: 'Spain', capital: true},
  {city: 'Rome', country: 'Italy', capital: true},
  {city: 'Oslo', country: 'Norway', capital: true},
  {city: 'Jaén', country: 'Spain', capital: false}
]

const filterList = arrCities.filter(arrCities =>{
  return arrCities.country === 'Spain' && arrCities.capital === false
}).map(arrCities =>({
    city: arrCities["city"]
}));

console.log(JSON.stringify(filterList));
```

```
"[{\"city\":\"Logroño\"},{\"city\":\"Jaén\"}]"
```

Ejercicio 4

Dado tres arrays de números, sacar en un nuevo array la intersección de estos.

```
const arrNumber1 = [1,2,3];
const arrNumber2 = [1,2,3,4,5];
const arrNumber3 = [1,4,7,2];
```

```
const arrNumber1 = [1,2,3];
const arrNumber2 = [1,2,3,4,5];
const arrNumber3 = [1,4,7,2];

var comun = arrNumber2.filter(arrNumber2 => arrNumber3.indexOf(arrNumber2) !==-1
                              && arrNumber1.indexOf(arrNumber2) !==-1);
console.log(comun);
```

```
[1, 2]
>
```

Ejercicio 5

Dado un array de ciudades, sacar en un nuevo array las ciudades no capitales con unos nuevos parámetros que sean **city** y **isSpain**. El valor de **isSpain** será un booleano indicando si es una ciudad de España.

Ejemplo: {city: "Logroño", isSpain: "true"}

```
const arrCities2 = [
  {city: 'Logroño', country: 'Spain', capital: false},
  {city: 'Bordeaux', country: 'France', capital: false},
  {city: 'Madrid', country: 'Spain', capital: true},
  {city: 'Florence', country: 'Italy', capital: true},
  {city: 'Oslo', country: 'Norway', capital: true},
  {city: 'Jaén', country: 'Spain', capital: false}
]
```

```
1  const arrCities2 = [
2    { city: 'Logroño', country: 'Spain', capital: false },
3    { city: 'Bordeaux', country: 'France', capital: false },
4    { city: 'Madrid', country: 'Spain', capital: true },
5    { city: 'Florence', country: 'Italy', capital: true },
6    { city: 'Oslo', country: 'Norway', capital: true },
7    { city: 'Jaén', country: 'Spain', capital: false },
8  ];
9  const filterList = arrCities2
10   .filter(arrCities2 => {
11     return arrCities2.country === 'Spain' && arrCities2.capital === false;
12   })
13   .map(arrCities2 => ({
14     city: arrCities2['city'],
15     isSpain: arrCities2["country"]==='Spain'
16   }));
17 console.log(JSON.stringify(filterList));
```

```
× Clear console
[{"city":"Logroño","isSpain":true},{"city":"Jaén","isSpain":true}]
```

Ejercicio 6

Crea una función que redondee un número float a un número específico de decimales.

La función debe tener dos parámetros:

- Primer parámetro es un número float con x decimales
- Según parámetro es un int que indique el número de decimales al que redondear
- Evitar usar el método toFixed()

Ejemplo de uso de la función:

```
const roundedResult = roundTo(2.123, 2);
console.log(roundedResult); // 2.12

const roundedResult = roundTo(1.123456789, 6);
console.log(roundedResult); // 1.123457
```

```
1  roundTo = function(numero,decimal){
2      const resul =Math.pow(10,decimal);
3      return Math.round((numero +Number.EPSILON)*resul)/resul
4  }
5  const roundedResult = roundTo(2.123, 2);
6  console.log(roundedResult); // 2.12
7
8  const roundedResult1 = roundTo(1.123456789, 6);
9  console.log(roundedResult1); // 1.123457
```

Clear console

```
2.12
1.123457
```

Ejercicio 7

Crea una función que retorne los campos de un objeto que equivalgan a un valor “falsy” después de ser ejecutados por una función específica.

La fundación debe tener dos parámetros:

- Primer parámetro es un objeto con x número de campos y valores
- Segundo parametro es una funcion que retorne un booleano, que se tiene que aplicar al objeto del primer parámetro

Ejemplo de uso de la función:

```
const result = returnFalsyValues({ a: 1, b: '2', c: 3 }, x => typeof x === 'string')

console.log(result); // {a: 1, c: 3}
```

```
const returnFalsyValues =(obj, callback) => {
  return Object.entries(obj)
                            .filter(([_key,value]) => !callback(value))
                            .reduce((prev,[key, _value]) =>{
                            prev[key] = obj[key];
                            return prev;
  })
}
const result = returnFalsyValues({ a: 1, b: '2', c: 3}, x => typeof x === 'string')
console.log(JSON.stringify(result)); // {a: 1, c: 3}
```

```
"[\"a\",1]"
```

Ejercicio 8

Crea una función que convierta un número de bytes en un formato con valores legibles ('B', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB')

La función debe tener 2 parámetros:

- Primer parámetro debe ser el número de bytes
- Segundo parámetro debe ser un número especificando la cantidad de dígitos a los que se debe truncar el resultado (esto se puede hacer con Number.prototype.toPrecision()). Por defecto, este parámetro debe de tener un valor de 3.

Ejemplo de uso de la función:

```
const result = fromBytesToFormattedSizeUnits(1000);
console.log(result); // 1KB

const result = fromBytesToFormattedSizeUnits(123456789);
console.log(result); // 123MB

const result = fromBytesToFormattedSizeUnits(-12145489451.5932, 5);
console.log(result); // -12.145GB
```

```
fromBytesToFormattedSizeUnits = function(number,decimal=3){
  const b = 1000;
  const tn= decimal < 0 ? 0 : decimal;
  const sizes = ['B', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'];

  if(number>0){
     const i= Math.floor(Math.log(number)/Math.log(b));
    return parseFloat((number/Math.pow(b,i)).toPrecision(tn))+sizes[i];
  }else{
    number *=-1;
    const i= Math.floor(Math.log(number)/Math.log(b));
    return "-"+parseFloat((number/Math.pow(b,i)).toPrecision(tn))+sizes[i];
  }

};
const result = fromBytesToFormattedSizeUnits(1000);
console.log(result); // 1KB

const result1 = fromBytesToFormattedSizeUnits(123456789);
console.log(result1); // 123MB

const result2 = fromBytesToFormattedSizeUnits(-12145489451.5932, 5);
console.log(result2);
```

```
"1KB"
"123MB"
"-12.145GB"
>
```

Ejercicio 9

Crea una función que a partir de un objeto de entrada, retorne un objeto asegurándose que las claves del objeto estén en lowercase.

La función debe tener un objeto como único parámetro.

Ejemplo de uso de la función:

```
const myObject = { NamE: 'Charles', ADDress: 'Home Street' };
const myObjLowercase = toLowercaseKeys(myObject);
console.log(myObjLowercase); // { name: 'Charles', address: 'Home Street' }
```

```
const myObject = { NamE: 'Charles', ADDress: 'Home Street' };
const myObjLowercase = toLowercaseKeys(myObject);


function toLowercaseKeys(obj){
  for(var key in obj){
    obj[key.toLowerCase()] = obj[key];
    delete obj[key];
  }
  console.log(JSON.stringify(obj));
}
```

```
"{\"name\":\"Charles\",\"address\":\"Home Street\"}"
>
```

Ejercicio 10

Crea una función que elimine las etiquetas html o xml de un string.

La función debe tener un string como único parámetro.

Ejemplo de uso de la función:

```
const result = removeHTMLTags('<div><span>lorem</span>
```

```
<strong>ipsum</strong></div>');

console.log(result); // lorem ipsum
```

```
const removeHTMLTags= (string) =>{
  return string.replace( /(<([^>]+)>)/ig, '');
}
const result = removeHTMLTags('<div><span>lorem</span> <strong>ipsum</strong></div>');

console.log(result);
```

```
"lorem ipsum"
>
```

Ejercicio 11

Crea una función que tome un array como parametro y lo divida en arrays nuevos con tantos elementos como sean especificados.

La función debe tener dos parámetros:

- El primer parámetro es el array entero que se quiere dividir.
- El segundo parámetro es el número de elementos que deben tener los arrays en los que se divida el array original del primer parámetro.

Ejemplo de uso de la función:

```
const result = splitArrayIntoChunks([1, 2, 3, 4, 5, 6, 7], 3);
console.log(result); // [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7 ] ]
```

```
var arrayNuevo=[];
const splitArrayIntoChunks= (arr,n) =>{
  for(let i=0;i<arr.length;i+=n){
    let trozo =arr.slice(i,i+n);
    arrayNuevo.push(trozo);
  }
  return arrayNuevo;
}
const result = splitArrayIntoChunks([1, 2, 3, 4, 5, 6, 7], 3);
console.log(result);
```

```
[[1, 2, 3], [4, 5, 6], [7]]
>
```