Certified Tech Developer
The Ultimate Degree

Programación Imperativa

¡Más prácticas!

Como ya observaron esta semana cerramos con los contenidos que veremos en esta materia: *variables*, *tipos de datos, condicionales y bucles*, que son herramientas esenciales para cualquier tipo de lenguaje de programación, pero para poder realizar un correcto uso de éstas necesitamos ejercitar nuestro pensamiento computacional, entonces vamos a hacerlo resolviendo los siguientes enunciados de la forma que creamos más conveniente.

Ejercicios

1. A partir de el siguiente array de edades nos solicitan realizar lo siguiente:

```
const edades = [11, 12, 15, 18, 25, 22, 10, 33, 18, 5];
```

   a. Obtener en un nuevo array las edades menores a 18.
   b. Obtener en un nuevo array las edades mayor o igual a 18.
   c. Obtener en un nuevo array las edades igual a 18.
   d. Obtener el menor.
   e. Obtener el mayor.
   f. Obtener el promedio de edades.
   g. Incrementar en 1 todas las edades.

Recordemos que hay muchas formas de resolver, nuestra tarea será pensar cuál es la mejor para poder reutilizar nuestro código con diferentes array de edades.

```
const edadesGrupo1 = [21, 12, 15, 18, 25];
const edadesGrupo2 = [2, 11, 54, 63, 24];
```

Excelente, ahora, como ya trabajamos un array, sabemos que este puede tener como elemento cualquier tipo de dato —Number, String, Objetos Literales, Array, etc—. Para continuar con la Mesa de Trabajo trabajaremos con un array de objetos literales, para ello invitamos a repasar la clase de **Objetos Literales** para tener en claro su estructura y sintaxis. Podemos decir que un objeto literal es una colección de propiedades y cada propiedad es una asociación entre una clave y un valor.

```
/*
 Observamos que es un array por su sintaxis que comienza y termina con corchetes []
   y sus elementos tienen la sintaxis de objetos literales {}
   Entonces, podemos decir que tenemos una array de objetos literales,
   o una colección de objetos literales
*/
const arrayDeObjetosLiterales =
[
    {},
    {},
    {},
    {}
];
```

2. Tenemos como base un array de cuentas bancarias, donde a cada una la representamos con un objeto literal. A partir de este array trabajaremos sobre los siguientes enunciados, resolviendo de la forma que nos parezca más adecuada

```
const arrayCuentas =
[
    {
      titular: "Arlene Barr",
      estaHabilitada: false,
      saldo: 3253.40,
      edadTitular: 33,
      tipoCuenta: "sueldo"
    },
    {
      titular: "Roslyn Torres",
      estaHabilitada: false,
      saldo: 3229.45,
      edadTitular: 27,
      tipoCuenta: "sueldo"
    },
    {
      titular: "Cleo Lopez",
      estaHabilitada: true,
      saldo: 1360.19,
      edadTitular: 34,
      tipoCuenta: "corriente"
    },
    {
      titular: "Daniel Malone",
      estaHabilitada: true,
      saldo: 3627.12,
      edadTitular: 30,
      tipoCuenta: "sueldo"
    },
    {
```

```
    titular: "Ethel Leon",
    estaHabilitada: true,
    saldo: 1616.52,
    edadTitular: 34,
    tipoCuenta: "sueldo"
  },
  {
    titular: "Harding Mitchell",
    estaHabilitada: true,
    saldo: 1408.68,
    edadTitular: 25,
    tipoCuenta: "corriente"
  }
]
```

a. Obtener un nuevo array de cuentas cuyas edades sean menores a 30.
b. Obtener un nuevo array de cuentas cuyas edades sean mayor o igual a 30.
c. Obtener un nuevo array de cuentas cuyas edades sean menores o igual a 30.
d. Obtener la cuenta con el titular de la misma más joven.
e. Obtener un array con las cuentas habilitadas.
f. Obtener un array con las cuentas deshabilitadas.
g. Obtener la cuenta con el menor saldo.
h. Obtener la cuenta con el mayor saldo.

Extras:

1. Desarrollar una función llamada generarID que reciba como parámetro el array de cuentas y agregue a cada elemento (objeto literal) una propiedad llamada id con un valor Numérico

2. Desarrollar una función llamada buscarPorId que reciba como parámetro el array de cuentas y un id, en caso de encontrar retornar la cuenta (él objeto literal completo), caso contrario retornar null

3. Desarrollar una función llamada filtrarPorSaldos que reciba como parámetro el array de cuentas y un saldo (Number), deberá retornar un array que se cuyas cuentas se encuentren por debajo del saldo enviado por parámetro

4. Desarrollar una función llamada incrementarSaldo que reciba como parámetro el array de cuentas, un id y un saldo, deberá incrementar él saldo de la cuenta, en caso de encontrar, caso contrario retornar undefined
    a. Reutilizando la función **buscarPorId**