DIGITALHOUSE

Programación Web Full Stack

# JSON y más métodos de arrays

## Práctica Integradora

### Objetivo

En esta clase **vamos a trabajar en grupo**. Seguiremos explorando el potencial de JavaScript y experimentando la diversidad de aplicaciones que se pueden realizar gracias al entorno de desarrollo que nos ofrece Node.js. Volveremos a retomar varios de los recursos que venimos utilizando en las clases anteriores.

Cuando tengas alguna duda que te impida avanzar, preguntale a tus profesores.

### ¡ Éxitos !

# Micro desafío 1:

## Instrucciones

### DH-Bici

El Tech Leader presenta al equipo un nuevo proyecto de **compra y venta de bicicletas.** Hay llevar un control del stock de bicicletas y generar código que permita analizar los datos de las mismas. Para lograr estos objetivos, nos encarga las siguientes tareas:

1. Crear una carpeta de trabajo y dentro de ella un archivo (**bicicletas.json**). Este último debe contener un array de objetos literales con todos las bicicletas que se tienen en stock. Por cada bicicleta se necesita conocer la siguiente información:
    a. **Marca** (Fierce, Shifter, Olmo, Battle, TopMega,SLP, Halley, Fixie, etc ).
    b. **Modelo** (FM18SI29AM211, FM18F29AM210, Regal, Sunshine, Techno, Ruta).
    c. **Rodado** (16, 26, 28, 29 ).
    d. **Año de fabricación** (2019, 2020, 2021, 2022, etc);
    e. **Color** (Celeste, negro, gris, amarillo, rosa, rojo, verde, Naranja, etc).
    f. **Peso en Kilogramos**(13, 15, 16, 17 );
    g. **Tipo** (Montañera, Paseo, Retro, Triatlón ).
    h. **Precio** (Coloque el precio que usted desee).
    i. **Vendida** (si ó no).

    Con esta información, puedes realizar todos los registros que quieras.

Para que tengan una idea de lo requerido, aquí le dejamos un ejemplo:

```
[
    {
        "id" : 1,
        "marca" : "Olmo",
        "modelo" :  "Regal",
        "rodado" : 29,
        "anio" : 2022,
        "color": "roja",
        "peso" : 16,
        "tipo" : "Montañera",
        "precio" : 120000,
        "vendida" : "si"
    }
]
```

2. Crear un archivo (**datosBici.js**). Este archivo será un **módulo propio**, en el que debes construir una función que tendrá la responsabilidad de importar el archivo de (**bicicletas.json**). Para esto, seguramente nos convenga usar el módulo nativo de NodeJs. File System - FS.
3. Guardar el contenido del archivo (**bicicletas.json**) en una variable y **convertirla** a un tipo de dato **array**. **¿Se te ocurre cómo?** Te dejamos un enlace para profundizar sobre el recurso a utilizar.

Este archivo es un **módulo propio** y por tal motivo una vez que se crea.. **¿Recuerdas cuál debe ser la última línea del archivo?.**

# Micro desafío 2:

## Instrucciones

1. Crear un nuevo archivo (**app.js**). En este, deberás importar el **módulo** creado (**datosBici.js**) y crear un **objeto literal** con el nombre de (**dhBici**). Este último tendrá como primer atributo (**bicicletas**), y debe contener un listado de todas las bicicletas.
2. Dentro del **objeto literal** crea las funcionalidades que el Tech Leader requiere que desarrollemos en base a lo exigido por el cliente:
    a. Crear una funcionalidad (**buscarBici)** que reciba por parámetro el (**id**) de la bicicleta y **devuelva la bici que corresponde**. En caso de **no encontrarla, deberá retornar null**. Estamos optimizando nuestro código, por lo que deberíamos utilizar el método **filter.**
    b. Crear una funcionalidad de (**venderBici**) que reciba el (**id)**. En caso de encontrar la bicicleta, debe asignarle el estado de vendida (**si**) y retornar todos los datos de la bicicleta. En el caso de no encontrar la bicicleta, debe retornar al usuario: "***Bicicleta no encontrada***". Puedes aprovechar la función (**buscarBici**).
    c. Crear la funcionalidad (**biciParaLaVenta**). Tendrá la responsabilidad de devolver todas aquellas bicicletas que aún no estén vendidas. Recuerda que estamos optimizando nuestro código, por lo que deberíamos utilizar el método **filter.**
    d. Finalmente el Tech Leader nos **felicita** por todo el trabajo y nos pide un último esfuerzo. Tenemos que desarrollar una funcionalidad (**totalDeVentas**) que retorne la suma del valor de todas las ventas realizadas. Se recomienda utilizar la función **reduce.**
3. Una vez hechas todas estas tareas, debemos comprobar la funcionalidad de las mismas, usando el console.log() e invocando cada una de ellas pasando los parámetros.

# Desafío extra (opcional):

## Instrucciones

Si llegaste hasta acá, significa que vienes trabajando bastante bien poniendo en práctica los diferentes contenidos vistos hasta el momento. **¡Felicitaciones!**

Para que no te quedes con las ganas y puedas seguir practicando, te proponemos que desarrolles las siguientes funciones dentro del mismo archivo (**app.js**) :

1. Crear una función (**aumentoBici)** que reciba por parámetro el porcentaje de aumento. Tendrá la responsabilidad de devolver el listado de todas las bicicletas con el aumento que reciba por parámetro. Estamos optimizando nuestro código, por lo cual, deberíamos utilizar el método **map.**
2. Crear la funcionalidad (**biciPorRodado**) que reciba por parámetro el "**número del rodado**". Tendrá la responsabilidad de retornar todas aquellas bicicletas que correspondan al rodado recibido por parámetro. Estamos optimizando nuestro código, por lo que deberíamos utilizar el método **filter.**
3. Crear una función (**listarTodasLasBici**) que tendrá la responsabilidad de mostrar al usuario el listado de todas las bicicletas registradas en el sistema. Para ello puedes valerte del uso de la función **forEach()** y también puedes investigar sobre el uso de la sentencia **for... of()**