Certified Tech Developer
The Ultimate Degree

Programación Imperativa

Modularizando nuestras aplicaciones

Vamos a realizar una serie de prácticas que nos van a ayudar a entender la funcionalidad y practicidad que traen los módulos a nuestros programas.

Descripción del problema

Vamos a simular una situación de trabajo para situarnos en un contexto. Estás trabajando como desarrollador de CodeAR S.A., una reconocida software factory. En el equipo de trabajo contás con María y con Juan.
María, líder técnica del área, presenta al equipo un nuevo proyecto para una concesionaria de automóviles, cuya principal línea de negocios es la compra y venta de automóviles. La concesionaria necesita construir una lista con todos los vehículos que tiene registrados.

La lista de autos ya registrados la obtendremos del siguiente archivo JSON. María, como prioridad, nos encarga el trabajo de realizar un módulo que se encargue de la lectura, parseo y escritura de estos datos para poder utilizarlos de manera eficiente en el resto de la aplicación. Para ello deberemos realizar las siguientes tareas:

1. Descargá el archivo JSON y guardalo en una carpeta de trabajo donde también crearás un archivo de JavaScript en el que establecerás el módulo de lectura y escritura. En tu archivo de JavaScript requerí el módulo nativo File System para poder trabajar con sus funcionalidades.

2. Ahora vas a realizar tu primera función, la de lectura de archivos. ¿Qué función del módulo fs podés utilizar para realizar una lectura sincrónica del archivo JSON? Creá una función que reciba como parámetro un string con el nombre del archivo

JSON, realice la lectura y haga un parseo de los datos para poder utilizarlos como un array de objetos literales.

```
const leerJson = function (nombreArchivo) {
  return // ¡Escribí tu código acá!
}
```

3. Una vez creada la función de lectura, harás lo propio con la función de escritura. Creá una función que reciba como parámetros el nombre del archivo y los datos a convertir en JSON. Para poder crear tu base de datos en JSON utilizarás la función de lectura para poder manipular el listado de autos —modificar y agregar autos—, y la función de escritura para sobrescribir nuestro JSON con la nueva lista actualizada cuando sea necesario.

```
const escribirJson = function (nombreArchivo, datos) {
  return; // ¡Escribí tu código acá!
};
```

4. Una vez creadas nuestras funciones, necesitamos poder exportarlas para utilizar en otro u otros archivos que necesitemos. María te dejo un ejemplo de cómo podrías modificar tu código para que sea más práctico a la hora de modularizar:

```
const jsonHelper = {
  leerJson: function (nombreArchivo) {
    return; // ¡Escribí tu código acá!
  },
  escribirJson: function (nombreArchivo, datos) {
    return; // ¡Escribí tu código acá!
  },
```

```
};
```

5. Una vez exportadas las funciones —o el objeto si utilizaste el ejemplo del punto anterior—, requerí estas funcionalidades en un nuevo archivo de JavaScript, y revisá que todo funcione correctamente.

Hasta acá María está más que contenta con nuestro desempeño y trabajo. Ahora nos pide avanzar con las características de la concesionaria de autos que se centrará en tener la lista de autos y dos funciones: una para agregar autos a la lista y otra para editar la lista y vender los vehículos. Empecemos:

6. En el archivo de JavaScript en el que tenemos requeridas nuestras funciones de lectura/escritura de archivos, creá el objeto literal **concesionaria**. Luego agregá la propiedad **autos**, la cual deberá tener la lista de vehículos del archivo JSON (previamente parseada). Verificá que puedas visualizar esta propiedad correctamente.

7. En el objeto **concesionaria**, creá un método llamado **agregarAuto**, el cual recibe como parámetros: una marca, un modelo, el año del vehículo, el precio, y la patente. El método deberá agregar el nuevo auto a la propiedad autos, y se debe guardar en la base de datos (rescribir el JSON) la lista actualizada.
PD: todos los vehículos nuevos tienen su propiedad “vendido” como “false” por defecto.

8. Por último, creá el método **venderAuto** el cual deberá recibir una patente por parámetro, luego recorrerá la lista de autos de **concesionaria** y, cuando encuentre al auto indicado, deberá modificar su propiedad “vendido” a “true”. Luego, se debe actualizar la base de datos con la lista actualizada como en el punto anterior.

María te dio un *boiler plate* de cómo podés comenzar con la preparación de estos puntos.

```
const concesionaria = {
  autos: ,
  agregarAuto: function() {
```

```
    //Escribí tu código acá
    return "Vehículo agregado correctamente"
  },
  venderAuto:  function() {
    let seleccionado;
    //Escribí tu código acá
    return "El vehículo: "+ seleccionado.marca + " " +
seleccionado.modelo + " ha sido vendido"
  },
}
```

Si llegaste hasta este punto, ¡felicidades! María está muy contenta por tu trabajo y la concesionaria quedó muy satisfecha con la aplicación.

En caso de que te haya sobrado tiempo o quieras seguir avanzando más tarde, te proponemos algunos métodos extra que podrías crear para abarcar más funcionalidades.

1. Crear un método llamado **totalDeVentas** que recorra la lista de autos y vaya sumando todos los precios de vehículos que hayan sido vendidos, y que retorne el precio total.
2. Crear un método llamado **eliminarAuto** el cual deberá recibir una patente por parámetro y eliminar el vehículo indicado. Investigá cómo podés hacer para eliminar un elemento específico de un array. Luego deberás actualizar la base de datos como en puntos anteriores.