Certified Tech Developer
The Ultimate Degree

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. Maria 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. Descarga el archivo JSON y guardalo en una carpeta de trabajo donde también crearás un archivo JS en el que crearemos el módulo de lectura y escritura.
   En tu archivo JS requerí el módulo nativo File System para poder trabajar con sus funcionalidades.

2. Ahora vamos a realizar nuestra primera función, la de lectura de archivos, ¿que función del módulo **fs** podemos utilizar para realizar una lectura sincrónica del archivo json? crea 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 codigo aca!
}
```

3. Una vez creada nuestra función de lectura, haremos lo propio con la función de escritura. Crea 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 (utilizaremos la función de lectura para poder manipular el listado de autos, modificarlos y agregar autos, y la función de escritura para sobreescribir nuestro json con la nueva lista actualizada cuando sea necesario)

```
const escribirJson = function (nombreArchivo, datos) {
  return; // Escribí tu codigo aca!
};
```

4. Una vez creadas nuestras funciones necesitamos poder exportarlas para utilizar en otro u otros archivos que necesitemos. Maria 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 codigo aca!
  },
```

```
  escribirJson: function (nombreArchivo, datos) {
    return; // Escribí tu codigo aca!
  },
};
```

5. Una vez exportadas las funciones, o el objeto si utilizaste el ejemplo del punto anterior, requeri estas funcionalidades en un nuevo archivo js, y revisa 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. Empezemos:

6. En el archivo JS en el que tenemos requeridas nuestras funciones de lectura / escritura de archivos, crea el objeto literal **concesionaria**, luego agrega la propiedad **autos**, la cual deberá tener la lista de vehículos del archivo json (previamente parseada). Verifica que puedas visualizar esta propiedad correctamente.
7. Al objeto **concesionaria** crea 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 debemos guardar en nuestra base de datos (reescribir el JSON) la lista actualizada

   PD: todos los vehículos nuevos tienen su propiedad vendido como false por defecto
8. Por último crea el método **venderAuto** el cual deberá recibir una patente por parámetro, y tendrá que recorrer la lista de autos de **concesionaria** y cuando encuentre al auto indicado, deberá modificar su propiedad vendido a true.

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

Si llegaste hasta este punto, ¡Felicidades! Maria está muy contenta por nuestro trabajo y la concesionaria quedó muy satisfecha con la aplicación.

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

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. Crea un método llamado **eliminarAuto** el cual deberá recibir una patente por parámetro y eliminar el vehículo indicado. Investiga cómo podés hacer para poder eliminar un elemento en específico de un array. Luego deberás

actualizar la base de datos siguiendo los pasos para actualizar la lista como en puntos anteriores.