Certified Tech Developer
The Ultimate Degree

Una carrera de autos

Introducción

Ahora que ya tenemos todos los conceptos principales aprendidos, es hora de combinarlos para construir una aplicación completa de principio a fin.

Para esta práctica vamos a estar modelando una carrera de autos, y decimos modelando porque vamos a armar un modelo usando Javascript que pueda representar una carrera real y nos permita simular sus resultados determinando los puestos en los que quedarían los corredores.

La estructura de los datos

Para representar a los corredores, tendremos un archivo JSON que contendrá un array de objetos literales. Es muy importante que te familiarices con este tipo de estructuras de datos, porque es de las que más se usan en el mundo de la programación web 😎👌.

Antes de ponernos a resolver las consignas es muy importante que sepamos con qué datos vamos a contar, veamos uno de los elementos del array.

```
{
      "piloto": "Monah Lyal",
      "patente": "ABC123",
      "velocidad": 161,
      "aceleracion": 0.31,
      "anguloDeGiro": 272,
      "cilindrada": 1500,
      "peso": 262.7,
      "puntaje": 76,
      "sancionado": false
},
```

- piloto → Es un **string** con el nombre del piloto
- patente → Es un **string** con la patente o placa del auto
- velocidad → Es un **number** con la velocidad del auto
- aceleración → Es un **number** con el coeficiente de aceleración del auto
- anguloDeGiro → Es un **number** con el máximo ángulo de giro del auto
- cilindrada → Es un **number** con el volumen de los cilindros del auto
- peso → Es un **number** con el peso en kilogramos del auto

- puntaje → Es un **number** con el puntaje obtenido durante la carrera
- sancionado → Es un **boolean** que nos indica si el corredor ha sido sancionado o no durante la carrera.

No te preocupes si luego no utilizamos todos los datos aquí contenidos, es habitual que nos llegue más información de lo necesario.

Tampoco te preocupes si los valores no tienen tanto sentido, en muchos casos, son auto-generados 🤖✨.

Ahora que sabemos con qué datos vamos a contar, es hora de empezar a programar, vamos con las consignas.

Consignas

A continuación te planteamos varios desafíos que deberás resolver usando tu ingenio y lo aprendido hasta el momento.

Es probable que no puedas terminar todos estos ejercicios durante el tiempo que tiene la mesa de trabajo, no te preocupes, lo importante es que los termines luego a tu ritmo para asegurarte de haber comprendido cómo llevar todo a la práctica.

En caso de que te queden dudas, no olvides usar el formulario, consultar por Discord y traer las dudas que queden después de eso a la próxima clase.

Sin más preámbulos, vamos con las consignas.

1. Obtener el listado de posibles participantes

Tomando como base el siguiente archivo JSON,

- Leer el archivo utilizando el módulo correspondiente de Node
- Parsearlo utilizando las herramientas que te provee Javascript

Resultado esperado: un array de objetos modificado correctamente para que funcione con las herramientas de JavaScript.

2. Crear un objeto literal que represente la carrera

Este objeto literal, que podemos llamar **carrera**, será nuestra representación de la carrera (valga la redundancia) con su datos (propiedades) y sus funcionalidades (métodos).

A. Agregar una propiedad llamada ***autos*** que contenga los autos obtenidos en el punto anterior.

B. Agregar una propiedad llamada **autosPorTanda** que contenga el valor **6**. Este valor representará la cantidad máxima de autos por tanda.

C. Agregar un método **autosHabilitados**, que retorne todos los autos habilitados, es decir, aquellos que **no estén sancionados**.

   - Este método no recibirá ningún parámetro.
   - Este método devolverá un array con los autos que estén habilitados para correr.

D. Agregar un método **listarAutos** que reciba como parámetro un array de autos e imprima por consola la siguiente información:

- El nombre piloto
- La placa o patente del auto
- El peso del auto
- El estado del piloto
    i. "sancionado" → si **sancionado** es **true**
    ii. "habilitado" → sí **sancionado** es **false**

PD: Este método deberá ser utilizado en la ejecución de los demás métodos que retornan un array de vehículos.

Resultado esperado al ejecutar el método: un mensaje por consola por cada auto con el siguiente formato:

Piloto: _____, patente: _____, peso en kg: _____, estado: _____.

Ejemplos:

Piloto: Leandro Ezequiel, patente: ABC123, peso en kg: 267, estado: habilitado.

Piloto: Esteban Piazza , patente: EFG567, peso en kg: 357, estado: sancionado.

E. Agregar un método **buscarPorPatente** que permita buscar el auto en función de su patente.

- Este método recibirá por parámetro un string que represente la patente a buscar
- En caso de encontrar un auto con la patente buscada, devolverá el objeto literal que lo representa.
- En caso contrario devolverá *undefined*

F. Agregar un método **filtrarPorCilindrada** que permite filtrar **los autos habilitados**, siempre y cuando su propiedad cilindrada sea menor o igual a la cilindrada enviada como argumento.

   - Este método recibirá por parámetro un number que represente la cilindrada máxima a buscar.
   - Este método devolverá un array con todos los autos que cumplan con la condición mencionada.
   - En caso de no encontrar ningún auto, devolverá un array vacío.
   - Este método debe usar **autosHabilitados** para buscar incluir solamente aquellos autos que estén habilitados**.**

G. Agregar un método **ordenarPorVelocidad** que ordene los autos de menor a mayor según su velocidad.

   - Este método devolverá un array con todos los autos ordenados por velocidad.

   Recordemos que Javascript tiene un método para hacer justamente lo que necesitamos 😉*.*

H. Agregar un método **habilitarVehiculo** que permita premiar una película en función de su título y guardar los cambios en el archivo JSON.

   - Este método recibirá por parámetro un **string** que represente la **patente** a buscar
   - En caso de encontrar un auto con la **patente** buscada:

i. Cambiará el valor de la propiedad **sancionado** a **true**.

ii. Escribirá los cambios en el archivo JSON que contiene los vehículos.

iii. Devolverá el auto modificado

- En caso contrario devolverá *undefined*

I. Agregar un método **generarTanda** que permita retornar un array de autos, que cumplan con las siguientes condiciones:

- Su piloto está habilitado.
- Su cilindrada sea menor o igual al valor enviado como argumento.
- Su peso sea menor o igual a un valor enviado como argumento.
- La cantidad devuelta sea como máximo la expresada en la propiedad **autosPorTanda**.

Para este método vamos a dejar que vos determines los parámetros que debería recibir.

Te recomendamos que pienses qué métodos de los que ya programaste podés reutilizar en este paso 😉.

J. Agregar un método llamado **pesoPromedio**, que me permita saber cual es el peso promedio de los vehículos que componen la tanda

- El método deberá reutilizar **autosPorTanda**

- Deberá retornar un mensaje indicando el peso promedio de los autos.

K. Agregar un método **listarPodio**, que calcule el podio en función del puntaje de los pilotos y muestre el resultado por consola.

- El método recibirá como parámetro un array de autos. Los mismos deberán ser generados con **generarTanda.**
- El método ordenará por puntaje los autos recibidos.
- El método imprimirá por consola los tres primeros puestos.

Resultado esperado al ejecutar el método: un mensaje por consola por cada auto con el siguiente formato:

El ganador es: _____, con un puntaje de _____.
El segundo puesto es para _____, con un puntaje de _____.
El tercer puesto es para _____, con un puntaje de _____.

Ejemplo:

El ganador es: Leandro Ezequiel, con un puntaje de: 70.
El segundo puesto es para Martin Cejas, con un puntaje de 55.
El tercer puesto es para Nicolas Lopez, con un puntaje de 52.