Certified Tech Developer
The Ultimate Degree

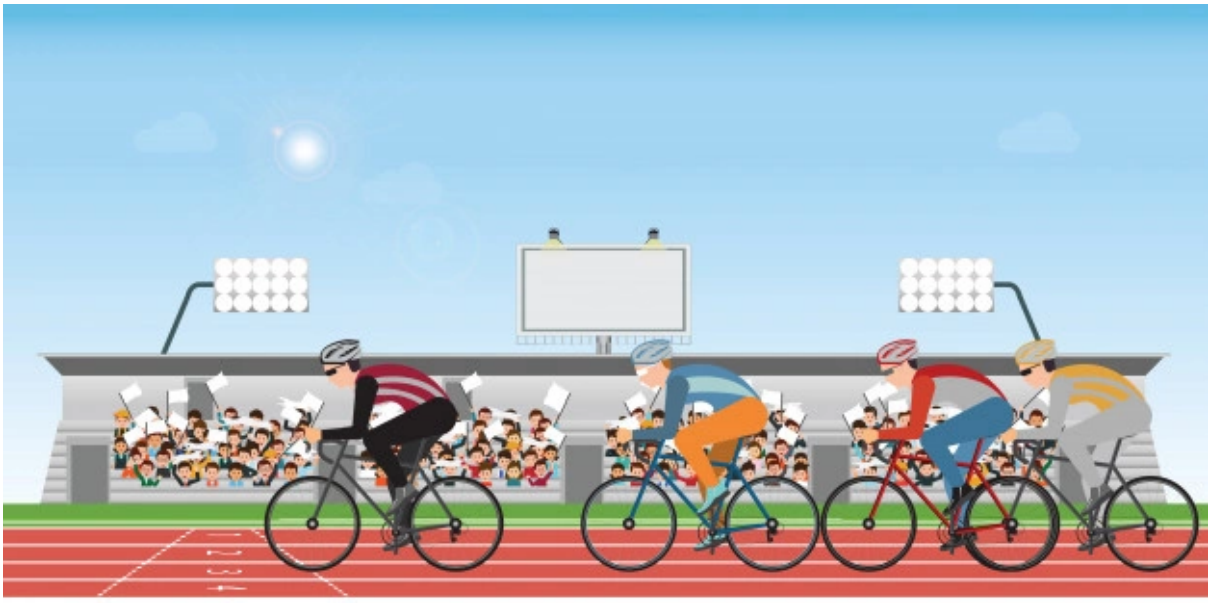
Una carrera de bicicletas

Introducción

Como dijimos ya varias veces, la práctica es muy importante a la hora de mejorar nuestras habilidades como programadores 💻🥷✨.

Les traemos entonces otra aplicación para desarrollar, en este caso vamos a estar modelando una carrera de bicicletas.

La idea es muy similar a lo que ya hicimos con la carrera de autos, solo que esta vez cambiaremos un poco los métodos.



La estructura de los datos

Para representar a los ciclistas tendremos un archivo JSON que contendrá un array de objetos literales. Ya lo dijimos, pero vale repetir, es 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 😎👌.

Veamos el detalle de la estructura de datos antes de ir a resolver las consignas.

```
{
      "id": 1,
      "ciclista": "Maure Benko",
      "puntaje": 6.29,
      "marca": "Specialized",
      "rodado": 64,
      "peso": 9.248,
      "largo": 116.17,
      "dopaje": true
},
```

- id → Es un **number** con un identificador único del registro
- ciclista → Es un **string** con el nombre del ciclista
- puntaje → Es un **number** con el puntaje obtenido durante la carrera
- marca → Es un **string** con la marca de la bicicleta
- rodado → Es un **number** con el tamaño del rodado de la bicicleta
- peso → Es un **number** con el peso en kilogramos de la bicicleta
- largo → Es un **number** con el largo en centímetros de la bicicleta
- dopaje → Es un **boolean** que nos indica si el corredor ha dado positivo en las pruebas de sustancias prohibidas para el deporte.

No te preocupes si luego no utilizamos todos los datos aquí contenidos, es habitual que nos llegue más información de la que necesitamos para nuestra aplicación.

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

Consignas

A continuación te planteamos varios desafíos que deberás resolver usando tu ingenio y lo aprendido hasta el momento. Sabemos que muchos de los métodos serán similares a los de la vez anterior, es importante que no copies las soluciones anteriores y que en lugar de eso intentes resolverlos desde cero.

Ya con la práctica anterior debería estar en condiciones de terminar los ejercicios en la mesa de trabajo, de todas maneras no te preocupes si no los terminas, lo importante es que lo hagas 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
- Guardar el listado en una variable

Resultado esperado: variable conteniendo un array con todos los ciclistas disponibles.

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 ***bicicletas*** que contenga las bicicletas obtenidas en el punto anterior.
B. Agregar una propiedad llamada **bicicletasPorTanda** que contenga el valor **4**. Este valor representará la cantidad máxima de bicicletas por tanda.
C. Agregar un método **ciclistasHabilitados** que devuelva una lista donde los ciclistas tengan un dopaje negativo.
    - Este método no recibirá ningún parámetro.
    - Este método devolverá un array con los autos que estén habilitados para correr.
D. Agregar un método **listarBicicletas** que reciba como parámetro un array de ciclistas e imprima por consola la siguiente información:
    - El nombre ciclista
    - El peso de la bicicleta

- El largo de la bicicleta
- El estado del ciclista
    i. “inhabilitado” → si **dopaje** es **true**
    ii. “habilitado” → si **dopaje** es **false**

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

Ciclista: _____, marca: _____, rodado: _____, peso: ____ kg, largo: ____ cm, estado: _____.

Ejemplos:

Ciclista: Leandro Ezequiel, marca: Venzo, rodado: 26, peso: 8.4 kg, largo: 168 cm, estado: habilitado.

Ciclista: Esteban Piazza , marca: Aurorita, rodado: 26, peso: 7.3 kg, largo: 177 cm, estado: inhabilitado.

E. Agregar un método **buscarPorId** que permita buscar un ciclista en función de su id.
    - Este método recibirá por parámetro un number que represente el id a buscar
    - En caso de encontrar un ciclista con el id buscado, devolverá el objeto literal que lo representa.
    - En caso contrario devolverá *undefinded*

F. Agregar un método **buscarPorRodado** que permita filtrar los ciclistas habilitados, siempre y cuando su rodado sea igual al enviado como argumento.
    - Este método recibirá por parámetro un number que represente el rodado a buscar.

- Este método devolverá un array con todos los ciclistas que cumplan con la condición mencionada.
- En caso de no encontrar ningún ciclista, devolverá un array vacío.
- Este método debe usar **ciclistasHabilitados** para buscar incluir solamente aquellos autos que estén habilitados**.**

G. Agregar un método **ordenarPorRodado** que ordene las bicicletas de menor a mayor según su rodado.

- Este método no recibirá ningún parámetro.
- Este método devolverá un array con todos las bicicletas ordenadas por rodado.

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

H. Agregar un método **generarTanda** que retorne un array de ciclistas, que cumplan con las siguientes condiciones:
   - El ciclista esté habilitado
   - El rodado sea igual al valor enviado como argumento
   - El peso sea menor o igual al valor enviado como argumento
   - La cantidad devuelta sea como máximo la expresada en la propiedad **bicicletasPorTanda**.

   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 😉.

I. Agregar un método que permita **calcularPodio**, el mismo deberá calcular al ganador y los siguientes dos puestos en función de su puntaje.

- El método recibirá como parámetro un array de ciclistas. Los mismos deberán ser generados con **generarTanda.**
- El método ordenará por puntaje los ciclistas 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.