Certified Tech Developer
The Ultimate Degree

Bucles y repeticiones

Para este ejercicio utilizaremos todos los elementos de las clases anteriores y de la actual. Los objetivos son:

- Reconocer **patrones**, elementos con características generales que interactúan, o no, con otros. Poder crear grupos y relaciones.
- **Abstraer** en conceptos y objetos computables los elementos del problema.
- **Descomponer** en conceptos y objetos de menor complejidad, dividir el problema en partes más pequeñas.
- Modelar el problema de la consigna bajando a papel o planilla de cálculos la representación en números, textos o gráficos de estos elementos.
- Expresar el modelo en formato de código, **algoritmizando** la solución.

¿Te suena familiar? Eso es porque esto no es otra cosa que el pensamiento computacional, algo que venimos trabajando desde que arrancamos (y realmente algo que hacemos durante toda nuestra vida en ¡situaciones cotidianas!).

Por lo tanto, te proponemos el siguiente ejercicio integrador para poder desarrollar un poco más esta práctica a nivel programación y código.

Empecemos...

Para esta oportunidad vamos a retomar la actividad de la clase anterior. Estuvimos trabajando sobre arreglos, accediendo a ellos, modificándolos, agregando y quitando elementos, y algunas cosas más. Pero como vimos en esa clase, muchas tareas se repetían incluso dentro de la misma función, parecía que debíamos realizar el mismo paso una y otra vez, una x cantidad de veces. Vamos a ver si podemos darle algo de dinamismo y eficiencia a nuestro código.

1. Partimos de nuestro array de películas, el cual estaba conformado de la siguiente manera:

```
let peliculas = ["star wars", "totoro",  "rocky", "pulp fiction",  "la vida es bella"]
```

Nos habían solicitado que pasemos todos los elementos a mayúsculas, lo cual en su momento lo habíamos hecho de manera “manual”. Ahorá, hagámoslo de una forma más automática, utilizando bucles.

```
function convertirAMayusculas(array){
  array[0] = array[0].toUpperCase()
  array[1] = array[1].toUpperCase()
  array[2] = array[2].toUpperCase()
  array[3] = array[3].toUpperCase()
  array[4] = array[4].toUpperCase()
  return array
}
```

2. Ahora necesitamos modificar la función *pasajeDeElementos()* la cual nos permite agregar los contenidos de nuestro array de pelis animadas al array de pelis original.

```
function pasajeDeElementos(array1, array2) {
  array1.push(array2.pop().toUpperCase())
  array1.push(array2.pop().toUpperCase())
  array1.push(array2.pop().toUpperCase())
  array1.push(array2.pop().toUpperCase())
  array1.push(array2.pop().toUpperCase())
  return array1
}
```

3. Para este punto, si decidiste trabajar en un archivo distinto en lugar de modificar el anterior, recordá que teníamos un infiltrado dentro de nuestras pelis animadas que debíamos sacar y guardar en otra variable antes de realizar el pasaje de elementos de un array a otro.

4. Por último, debemos modificar nuestra función comparadora de puntajes para las películas como venimos haciendo hasta ahora.

```
const asiaScores = [8, 10, 6, 9, 10, 6, 6, 8, 4];
const euroScores = [8, 10, 6, 8, 10, 6, 7, 9, 5];

function compararCalificaciones(asia, europa) {
  let comparacionesAsiaEuropa = []
  comparacionesAsiaEuropa[0] = asia[0] === europa[0]
  comparacionesAsiaEuropa[1] = asia[1] === europa[1]
  comparacionesAsiaEuropa[2] = asia[2] === europa[2]
  comparacionesAsiaEuropa[3] = asia[3] === europa[3]
```

```
  comparacionesAsiaEuropa[4] = asia[4] === europa[4]
  comparacionesAsiaEuropa[5] = asia[5] === europa[5]
  comparacionesAsiaEuropa[6] = asia[6] === europa[6]
  comparacionesAsiaEuropa[7] = asia[7] === europa[7]
  comparacionesAsiaEuropa[8] = asia[8] === europa[8]
  return comparacionesAsiaEuropa
}
```

Extra bonus

Si llegaste hasta acá, estás más que bien. ¡Felicidades!

Para que no te quedes con las ganas y puedas seguir practicando si así lo deseas, te proponemos este otro ejercicio. Tené en cuenta que a partir de acá los ejercicios pueden escalar en dificultad. Como siempre decimos, paciencia, ignorá la complejidad y tratá de resolverlo con las herramientas que tengas a tu disposición. También podés buscar información extra en Google o documentaciones que conozcas.

Concurso de clavados

Un cliente nos pide que hagamos una aplicación que pueda determinar a los ganadores de un concurso de clavados que se realizó el fin de semana.

Para ello vamos a necesitar seguir las siguientes instrucciones e información para poder desarrollar nuestra app.

Cada participante cuenta con 5 clavados, de los cuales se formarán sus puntajes individuales. Los participantes con sus respectivos puntajes son los siguientes:

- Participante A: 5, 8, 4, 9, 5.
- Participante B: 8, 7, 8, 6, 8.
- Participante C: 7, 5, 10, 8, 3.

La competencia consta de 2 modalidades de selección para un ganador:

- Mejor promedio (el mayor puntaje promedio entre los competidores).
- Mayor puntaje (el mayor puntaje de entre los 5 clavados de cada participante).

Con esta información, nuestro tech leader nos pide lo siguiente:

1. Determinar cuál sería la forma ideal de representar a cada participante con sus puntajes.

2. Crear una función *puntajePromedio* la cual recibe un participante por parámetro y deberá calcular —y **retornar**— el puntaje promedio del mismo.

3. Crear una función *puntajeNayor* la cual recibe un participante por parámetro y deberá buscar —y **retornar**— el mayor puntaje que tenga el participante.

4. Luego, nuestro tech leader nos solicita que —creadas estas dos funciones— generemos una nueva funcionalidad llamada *competencia* la cual recibirá a los 3 participantes por parámetros y ejecutará las dos funciones creadas previamente para calcular los promedios y puntajes mayores de cada uno. Además, esta deberá anunciar (mostrar por la consola) al ganador de cada modalidad de puntaje.