Certified Tech Developer
The Ultimate Degree

Programación Imperativa

Ejercicio para las mesas de trabajo

Esta semana vimos cómo manipular tipos de datos string y array de forma repetitiva. Luego, trabajamos con la automatización de código mediante ciclos y bucles para evitar esa repetición manual. A su vez, conocimos algunos métodos de arrays que nos ayudan y facilitan ciertas funcionalidades útiles a la hora de efectivizar nuestro código. Para afianzar estos conceptos, vamos a desarrollar una situación específica en la que todos ellos nos puedan ser de utilidad, para así poder obtener una solución efectiva a nuestros problemas. ¡Éxitos!

Descripción del problema

Nos contactan desde un teatro donde hace falta determinar el ganador de un concurso de stand up que consta de 3 presentaciones por participante. Nos piden que armemos el programa que determine al ganador y —a continuación— nos explican el funcionamiento del concurso para que lo tengamos en cuenta a la hora de realizar nuestro programa:

El público, máximo 100 personas, votó quién considera que estuvo mejor al terminar cada etapa. Por ejemplo, sube Alicia, se presenta y baja; sube Bob, se presenta y baja. Terminada la presentación, el público vota indicando quién cree que ganó esa tanda. Luego, continúa la siguiente tanda, igual que la primera. Y finalmente, una tercera.

¿Cómo representarías en una tabla tipo planilla de cálculos este problema?

Participante	votos etapa 1	ganador v1	vot etapa 2	G v2	v etapa 3	G v3	Resultado
				x		x	
		x					

Como estos concursos se dan online, ocurren miles cada día. Nos llega a nuestro servidor la información por cada concurso en formato de arrays, uno por cada participante. Por lo tanto, los recibimos así:

- El array de Alicia es: alicia = [23, 82, 46]
- El array de Bob es: bob = [45, 8, 32]

Vayan planteando en código mientras piensan cómo realizar la comparación de cada etapa.

```
const alicia = [23, 69, 32];
const bob = [12, 67, 43];

function encontrarGanador(a, b) {
   //su solución aquí
}
```

La tarea consiste en enfrentar estas votaciones comparando a[0] con b[0], a[1] con b[1] y a[2] con b[2].

Si a[i] > b[i], entonces, Alicia recibe 1 punto.
Si a[i] < b[i], entonces, Bob recibe 1 punto.
Si a[i] === b[i], ninguna persona recibe un punto.

Los puntos de comparación son los puntos totales que ganó una persona. ¡Ojo! No los votos, sino los puntos ganados en cada etapa.

Ejemplo:
const alicia = [23, 67, 32];
const bob = [12, 67, 43];

puntosAlicia = 1
puntosBob = 1

Para los elementos en el índice 0, Alicia recibe un punto porque a[0] > b[0].
Para la etapa siguiente a[1] y b[1] son iguales, no se obtienen puntos.
Finalmente, para los elementos del índice 2 (tercera etapa), a[2] < b[2] por lo que Bob recibe un punto.

```
    const alicia = [23, 69, 32];
    const bob = [12, 67, 43];



    function encontrarGanador(a, b) {
        let puntosPrimerParticipante = 0;
        let puntosSegundoParticipante = 0;
       //continua con esto.
    }

    console.log("El ganador es: " + encontrarGanador(a, b) + "
participante");
```

Dadas las consignas y cómo funciona el sistema de puntos, debemos calcular y encontrar al ganador utilizando una estructura **for** que nos evite tener que calcular cada tanda por separado. Recordemos usar las estructuras **if/else** en caso de ser necesario para comparar los votos de cada participante!.

Ejercicios extra bonus

Para que no te quedes con las ganas y puedas seguir practicando si así lo deseas, te proponemos algunos ejercicios más. Tené en cuenta que a partir de acá los ejercicios pueden escalar en dificultad, tanto estructural como lógica. 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.

Aclaración: este ejercicio no es obligatorio hacerlo durante la clase porque probablemente el tiempo no lo permita.

digitalHouse()

Crea la función digitalHouse() la cual recibirá 2 números como parámetros. La función deberá imprimir por pantalla los números del 1 al 100, pero teniendo en cuenta los siguientes criterios:

- Si el número a imprimir es múltiplo del primer parámetro que se ingresó, deberá mostrar el string “Digital” en lugar del número.
- Si el número a imprimir es múltiplo del segundo parámetro ingresado, deberá mostrar el string “House” en su lugar.
- Si el número a imprimir es múltiplo de ambos parámetros, deberá mostrar el string “Digital House” en lugar del número.

sumArray()... Reloaded

Vamos a retomar el ejercicio sumArray() que hicimos en la clase de arrays, pero esta vez lo modificaremos para que pueda recibir un array de números de cualquier cantidad de elementos.

Si no recordás el enunciado original: deberás crear una función sumaArray que acepte un arreglo de números y devuelva la suma de todos ellos.

Ejemplo:

- sumArray([1,2,3]) // 6
- sumArray([10, 3, 10, 4]) // 27
- sumArray([-5,100]) // 95

Ya que estamos retocando funciones, hagamos lo propio con la función del ejercicio simulación join().
Deberás modificar la función join() de manera que pueda recibir un array de strings de cualquier cantidad de elementos.

Importante: no podés usar el método Array.join() original.

Por ejemplo:

join(["h","o","l","a"]) debe retornar el string "hola".

join(["c","h","a,"u"]) debe retornar el string "chau".