

Laboratorio M5-16

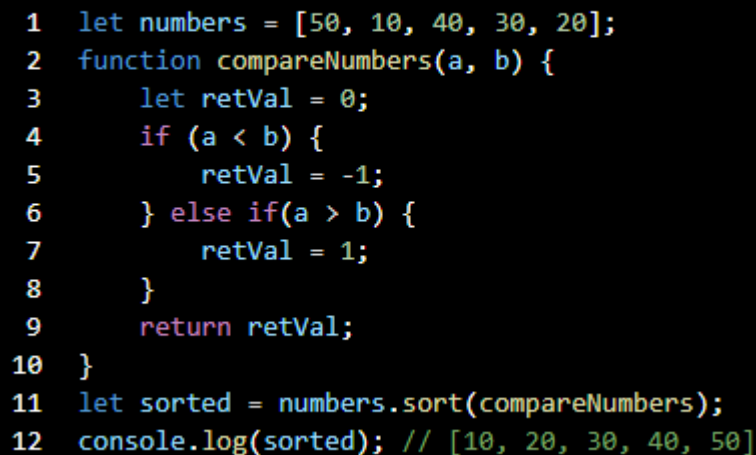
Realizar los siguientes ejercicios en tu Editor de Código preferencia Visual Studio Code, adjunta tus archivos js.

Enviar laboratorio a: tareas@juanpageek.com

Recuerda adjuntar el número de grupo al que perteneces en el asunto del email

Ejercicio No.37

Los arreglos en JavaScript tienen disponible un método **sort** que, como puedes suponer, te permite ordenar sus elementos. Este método toma como argumento una función que comparará dos elementos del arreglo. La función debe devolver cero si consideramos que los argumentos son iguales, un valor menor que cero si consideramos que el primero es menor que el segundo y un valor mayor que cero en caso contrario. Mira el ejemplo:



```
1 let numbers = [50, 10, 40, 30, 20];
2 function compareNumbers(a, b) {
3     let retVal = 0;
4     if (a < b) {
5         retVal = -1;
6     } else if(a > b) {
7         retVal = 1;
8     }
9     return retVal;
10 }
11 let sorted = numbers.sort(compareNumbers);
12 console.log(sorted); // [10, 20, 30, 40, 50]
```

A. Intenta modificar el código anterior para que sea lo más corto posible.
Sugerencias:

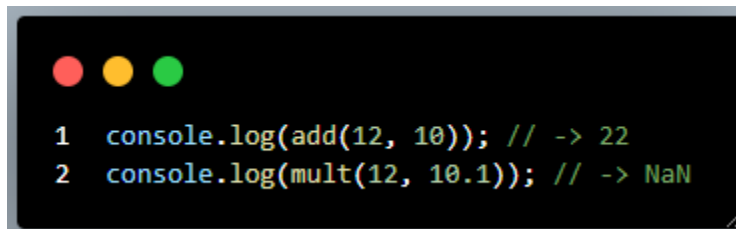
- Usar una función anónima.
- Usar una función arrow o flecha
- Considera omitir la sentencia if.

B. Luego modifica la función para que los elementos se ordenen en orden descendente, no en orden ascendente como en el ejemplo.

Ejercicio No.38

Escribe tres funciones con los nombres **add**, **sub** y **mult**, que tomarán dos argumentos numéricos. Las funciones son para verificar si los argumentos dados son enteros (emplea **Number.isInteger**). Si no, devuelven **NaN**, de lo contrario, devuelven el resultado de la suma, la resta o la multiplicación, respectivamente. Las funciones deben declararse mediante una instrucción de función.

Ejemplo de su uso y resultados esperados:

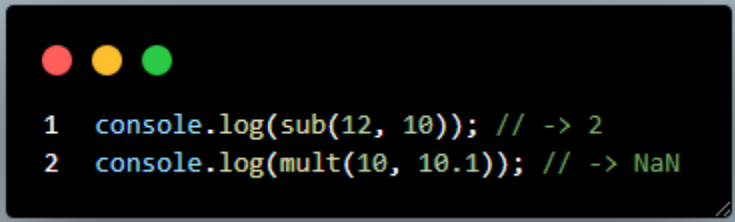


```
1 console.log(add(12, 10)); // -> 22
2 console.log(mult(12, 10.1)); // -> NaN
```

Ejercicio No.39

Reescribe las funciones de la tarea anterior usando una expresión de función arrow o flecha, tratando de escribirlas en la forma más corta posible.

Ejemplo de su uso y resultados esperados:

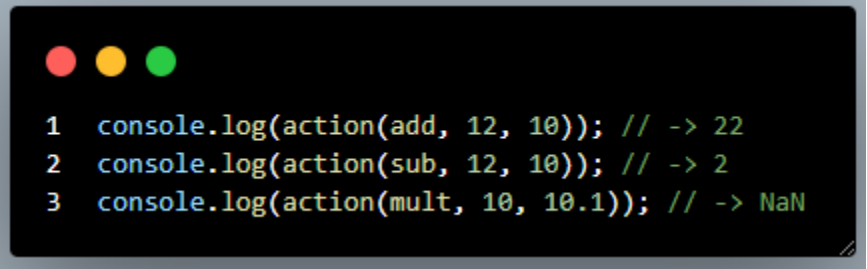


```
1 console.log(sub(12, 10)); // -> 2
2 console.log(mult(10, 10.1)); // -> NaN
```

Ejercicio No.40

Escriba una función **action** que tomará la función callback como su primer argumento y los otros dos argumentos como números. Como función callback, podrá pasar una de las tres funciones de la tarea anterior. La función **action** llamará a la función callback que se le pasó y devolverá el resultado obtenido. La función callback aceptará el segundo y el tercer argumento de la invocación.

Ejemplo de su uso y resultados esperados:



```
1 console.log(action(add, 12, 10)); // -> 22
2 console.log(action(sub, 12, 10)); // -> 2
3 console.log(action(mult, 10, 10.1)); // -> NaN
```

Ejercicio No.41

Escribe un programa que imprima (en la consola) números enteros consecutivos 10 veces, en intervalos de dos segundos (comienza con el número 1). Utiliza las funciones `setInterval`, `clearInterval` y `setTimeout`.

Ejemplo de su uso y resultados esperados:

1
2
3
4
5
6
7
8
9
10

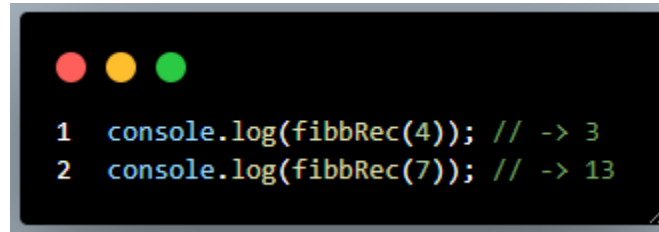
Ejercicio No.42

Escribe una función que calcule el n-ésimo elemento de la sucesión de Fibonacci. Esta secuencia se define mediante una fórmula:

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{if } n > 1 \end{cases}$$

Entonces, cada elemento de la secuencia (excepto los dos primeros) es la suma de los dos anteriores. Por ejemplo: $F_1 = 1$, $F_2 = F_1 + F_0 = 1$, $F_3 = F_2 + F_1 = 2$ y $F_6 = F_5 + F_4 = 8$. La función debe usar recursividad. En la definición, usa una expresión de función (almacena una función anónima en una variable).

Ejemplo de su uso y resultados esperados:



```
1 console.log(fibbRec(4)); // -> 3
2 console.log(fibbRec(7)); // -> 13
```

Ejercicio No.43

Reescribe la función del ejercicio 42 usando una expresión de función arrow o flecha, pero intenta acortar tu código tanto como sea posible (emplea operadores condicionales y trata de no usar variables adicionales que no sean el parámetro **n**).

Ejercicio No.44

Escribe una versión iterativa de la función del ejercicio 42 (usa el bucle **for**). Declara la función usando una instrucción de función.