

## EJERCICIOS - OBJETO STRING

1. Crea las siguientes funciones y llámalas desde una página XHTML para mostrar su funcionamiento:

- a) `invierteCadena(cad_arg)`: devuelve invertida una cadena dada por el usuario
- b) `inviertePalabras(cad_arg)`: devuelve invertidas las palabras contenidas en la cadena.
- c) `encuentraPalabraMasLarga(cad_arg)`: para una cadena de caracteres dada devuelve la longitud de la palabra más larga en ella contenida
- d) `filtraPalabrasMasLargas(cad_arg, i)`: para una cadena de caracteres y un valor numérico (i), devuelva el número de palabras cuya longitud es mayor a i.
- e) `cadenaBienFormada(cad_arg)`: formatea correctamente la cadena pasada, de tal modo que sólo aparece en mayúscula la primera letra y el resto en minúscula.

2. Definir una función que muestre información sobre una cadena de texto que se le pasa como argumento. A partir de la cadena que se le pasa, la función determina si esa cadena está formada sólo por mayúsculas, sólo por minúsculas o por una mezcla de ambas.

3. Realizar un función que permita localizar todas las apariciones de una subcadena dentro de otra.

4. Crea una función que tomando una cadena de texto como entrada coloque todas sus consonantes al principio y todas sus vocales al final de la misma, eliminando los blancos.

5. Desarrolla una función que elimine los caracteres repetidos de una cadena de texto, incluidos los blancos.

6. Implementa una función que tomando dos cadenas como entrada nos diga si la segunda es una subcadena de la primera y cuál es la primera posición a partir de la que esto ocurre.

7. Desarrolla una función que tomando como entrada una cadena de texto nos devuelva si es o no un palíndromo.

8. Implementa una función que tomando como entrada una cadena de texto sea capaz de contabilizar el número de palabras. Ten en cuenta que entre dos palabras puede haber más de 1 blanco, e incluso pueden aparecer al principio o final de ésta.

## 9.-Validación de Tarjeta de Crédito

You're starting your own credit card business. You need to come up with a new way to validate credit cards with a simple function called `validateCreditCard` that returns `true` or `false`.

Here are the rules for a valid number:

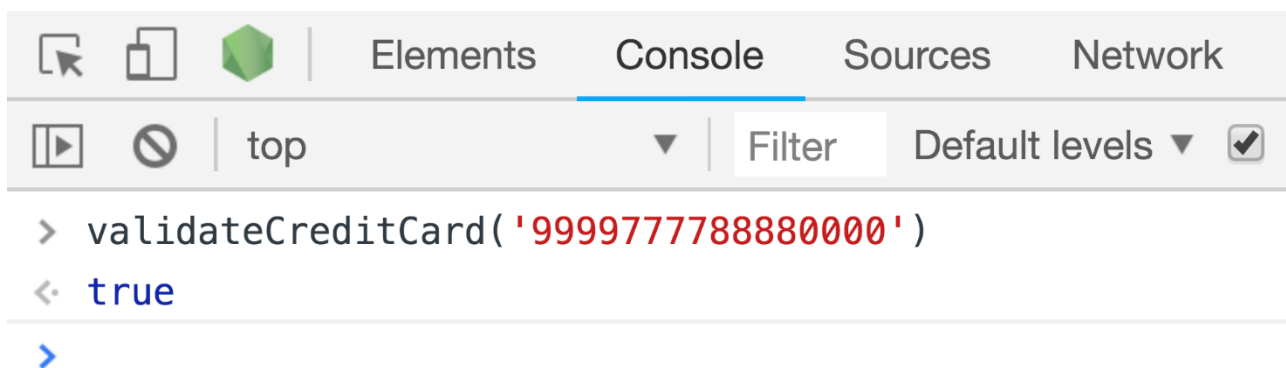
- Number must be 16 digits, all of them must be numbers
- You must have at least two different digits represented (all of the digits cannot be the same)
- The final digit must be even
- The sum of all the digits must be greater than 16

The following credit card numbers are valid:

- 9999777788880000
- 6666666666661666

The following credit card numbers are invalid:

- a92332119c011112 *invalid characters*
- 4444444444444444 *only one type of number*
- 1111111111111110 *sum less than 16*
- 6666666666666661 *odd final number*



```
> validateCreditCard('9999777788880000')
< true
```

10.- Mejora la función anterior creando validateCreditCard2 y añade los siguientes elementos:

A valid credit card number may also contain dashes, to make a card number easier to read. For example, the following credit card numbers are now also valid:

- 9999-7777-8888-0000
- 6666-6666-6666-1666

Update your program to allow such numbers. (Hint: **Remove the dashes** from the input string before checking if the input credit card number is valid.)

### Opcional (mejora propia de validación de tarjeta de crédito)

11.- Make your credit card scheme even more advanced! What are the rules, and what are some numbers that pass or fail? Ideas: check expiration date! Check out the [Luhn Algorithm](#) for inspiration.