Módulo 6 - Algoritmos II - Laboratorio

Enigma

En criptografía, una de las formas más sencillas de codificar mensajes consiste en sustituir o reemplazar letras por otras distintas.

Por ejemplo, dada la siguiente frase:

> hola amigo

Podríamos reemplazar la letra 'a' por una 's', la 'o' por una 'z' y la 'm' por el símbolo '*'. La frase quedaría:

> hzls s*igz

En este ejemplo, la transformación sufrida podría expresarse como:

- a => s
- o => z
- m => *

En código podríamos expresarlo como:

```
var plainAlphabet = "aom";
var encryptedAlphabet = "sz*";
```

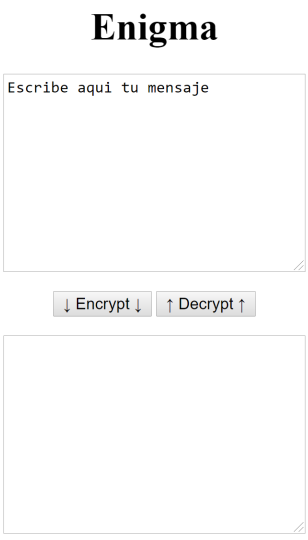
Donde cada letra se relaciona con su letra transformada a través del índice o posición que ocupa en el string.

Vamos a implementar un encriptador / desencriptador de mensajes utilizando esta técnica.

Interfaz

- Utiliza dos elementos `textarea`, uno que contendrá el mensaje en claro y otro con el mensaje encriptado.
- Añade también 2 botones, uno para encriptar y otro para desencriptar.

Un ejemplo sencillo podría ser el siguiente:



Algoritmo

El algoritmo de encriptación es sencillo. La transformación que vamos a aplicar a cada letra va a venir dada por:

```
var plainAlphabet = "abcdefghijklmnopqrstuvwxyz:()!¡,'";
var encryptedAlphabet = "qw,ert(yuio'pa:sdfg!hjklz¡xcv)bnm";
```

De modo que:

- 'a' se encriptará como 'q'
- 'b' se encriptará como 'w'
- 'c' se encriptará como ','
- y así sucesivamente

Implementa también la desencriptación, que es exactamente igual pero aplicando la transformación al revés:

- 'q' se desencriptará como 'a'
- 'w' se desencriptará como 'b'
- etc

De esta forma, cuando el usuario introduzca un texto en claro y haga click en 'Encrypt', debería encriptarse el mensaje y mostrarse en el `textarea` inferior. Si por contra introduce un mensaje encriptado y presiona 'Decrypt', deberá desencriptarse el mensaje y mostrarse en el `textarea` superior.

A prueba

¿Podrías descifrar el siguiente mensaje?

> b': yqg ,:agr(hue:) shrerg jq,u'qf q !hg ,:psqñrf:g rg,fuwurae: s:f r' ,yq!x mbg:z 'q 'r,yrn !hfua(t'usqfuq ,:apu(:)m z q ,:a!uahq,u:a rajuq ha pragqir ,utfqe:n br' dhr !h dhurfqg)

Opcional

Simplifica el algoritmo anterior gracias a la siguiente información:

Los strings son iterables y se comportan de forma similar a un array. Disponen de métodos que nos ayudan a buscar elementos. Uno de estos métodos que puede resultaros de gran utilidad es `indexOf()`. Este método devuelve el primer índice (en base 0) encontrado para un carácter dado, o lo que es lo mismo, la posición de la primera ocurrencia de dicho carácter. Es decir, este método hace una iteración por el string buscando el carácter que habéis proporcionado, y en cuanto lo encuentra devuelve la posición de dicho carácter dentro del array. Ejemplo:

```
var myString = "casa";
myString.indexOf("a"); // 1
```

Generador Aleatorio

Vamos a construir un generador aleatorio que permita escoger, de forma aleatoria, una cantidad determinada de números de entre un mínimo y un máximo. Para entenderlo mejor, os daremos la firma que debe tener la función:

```
var randomPick = (n, min, max) => {
  ...
}
```

De esta forma, para escoger 10 números aleatorios entre el 1 y el 100, usaremos la función anterior del siguiente modo:

```
randomPick(10, 1, 100);
```

y nos devolverá un array con dichos números, por ejemplo:

```
[24, 83, 2, 94, 84, 38, 23, 69, 16, 89];
```

IMPORTANTE: No deben repetirse los numeros escogidos aleatoriamente.

Este tipo de generador flexible es muy útil para una gran diversidad de problemas, por ejemplo:

```
randomPick(6, 1, 49); // Apuesta automática de la primitiva
randomPick(15, 1, 15); // Escoge combinación de bolas de billar
randomPick(1, 1, 6); // Tirada aleatoria de un dado
```

Para este ejercicio no se necesita construir una UI, simplemente muestra el resultado por consola.