**Lab 1 Documentation**

**The Caesar Cipher**

**Marcu Alexandru Daniel, Gr. 934/1**

**Description of the cipher:**

Being one of the oldest and simplest ciphers in the history of cryptography, the Caesar Cipher is a type of substitution cipher consisting of shifting each character in the plaintext with a number of letters to the right (or left). The key of the cipher is a natural number representing with how many positions each character is shifted.

**Technologies used**

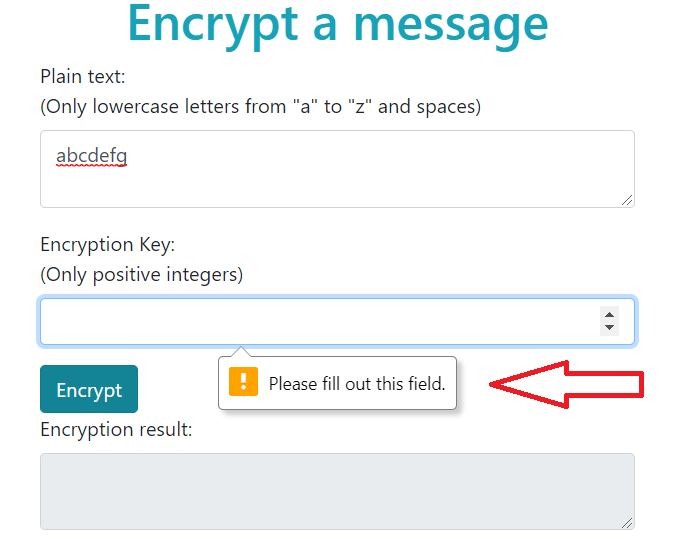
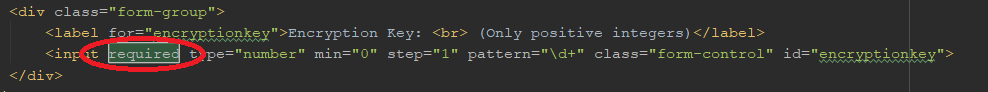
The technologies I chose to implement this algorithm were JavaScript for the back-end, where I used the jQuery library for a cleaner and faster implementation, and plain HTML files with Bootstrap classes to design the elements of the front-end.

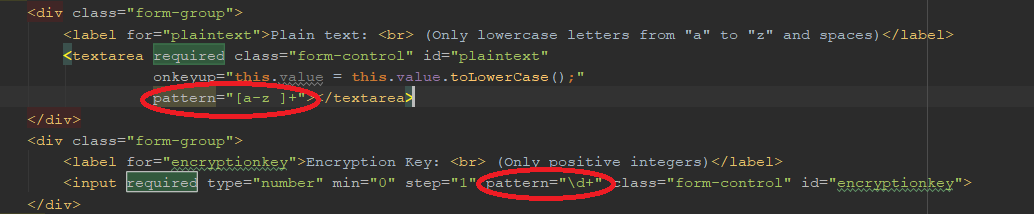
**Validation of the input data**

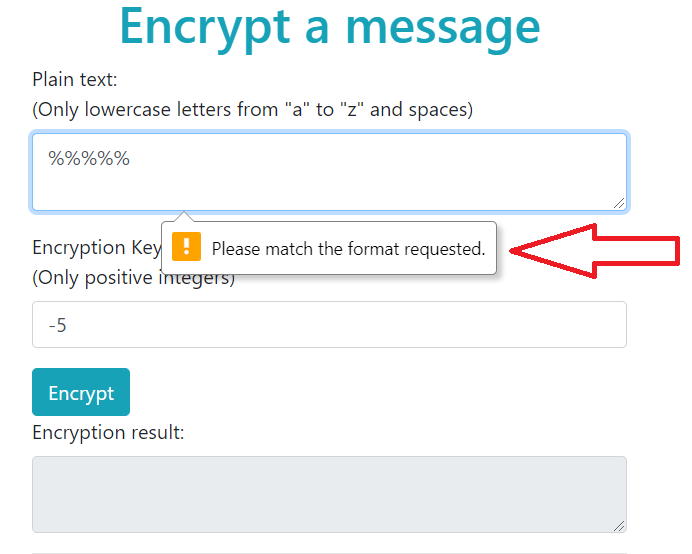
**The alphabet** used in the implementation consists of the small letters of the English alphabet (a to z) and space character. **The keys** could only be a natural number (0, 1, 2, 3…).

In order to ensure all of these constraints, I made use of the “**required**”, “**min**”, “**step**” and “**pattern**” attributes of Bootstrap.

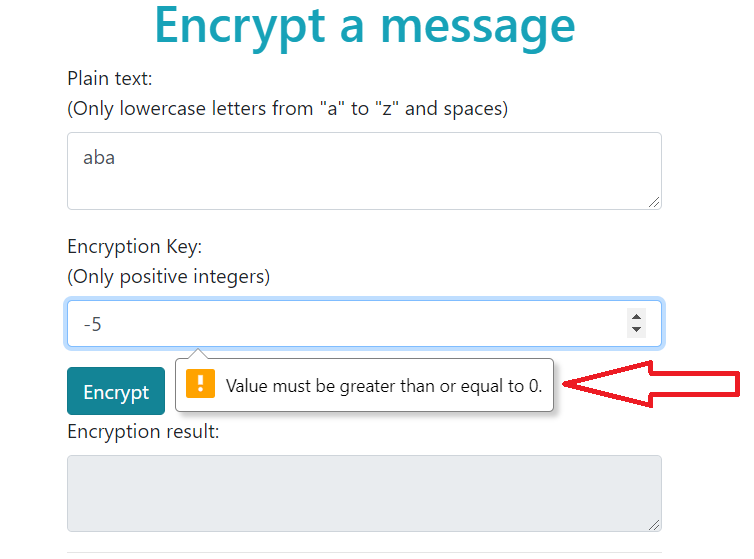
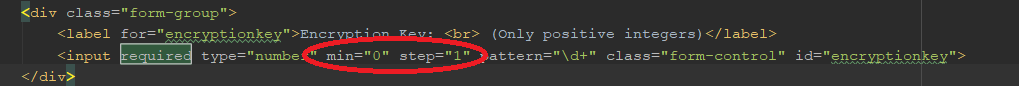
* The “**required**” attribute was used to ensure that the user puts at least a character in the plain text or in the cipher text, and also chooses a key. If he presses the Encrypt/Decrypt button without filling one of the inputs, a tooltip will appear near the empty input prompting the user to fill in that field.



* The “**pattern**” attribute was used to ensure that the user could not encrypt/decrypt a plain text/cipher text containing characters that are not in the accepted alphabet, and that the user cannot give a key which is consisting of letters. 



* The “**min**” and “**step**” attributes were used on the **key** inputs to ensure that the user cannot give akey which is not a natural number.



* A special event listener was added to the plain text/cipher text input so that when a user enters an uppercase letter, it will be automatically converted to a lowercase letter

**The algorithm**

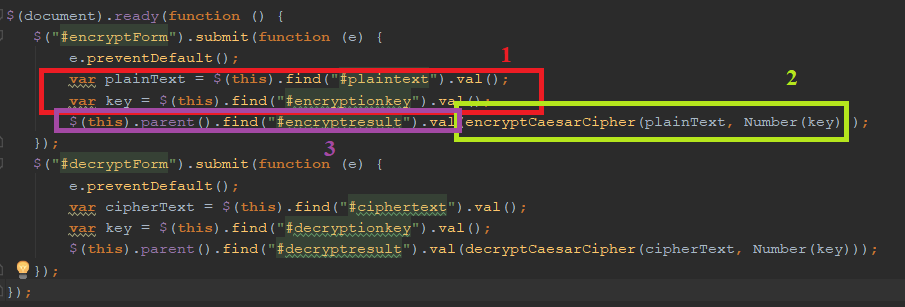
First of all, the mapping of the letters into numbers and from numbers into letters was hardcoded as a Map in the .js file where the algorithm was implemented.

The whole algorithm is computed of 3 functions:

* + a main function that submits the input on button press and displays the output in a reserved read-only area
  + a function that encrypts the plain text given a key
  + a function that decrypts the cipher text given a key

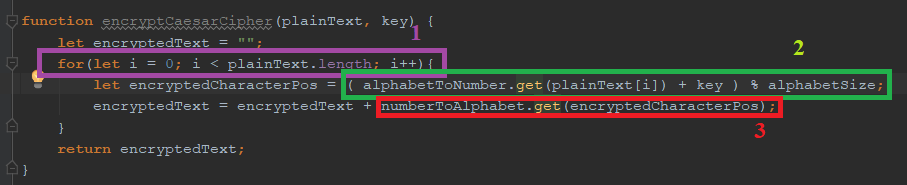
**The main function:**

1. The main function takes the text to be encrypted/decrypted from the corresponding inputs.
2. The main function passes the input to the encryption/decryption function.
3. The main function displays the output in the corresponding reserved area.



**The encryption function:**

1. Start with an empty encrypted text. Consider every character in the plain text.
2. For each character in the plaintext, it first maps it to its corresponding position from the alphabet, then adds the key to that position (shifts) and after that, if the shifted position is out of bounds of the alphabet, we simply to the “mod” operation with the size of the alphabet, so we bring the position to the correct bounds.
3. Transform the position to the corresponding letter and add it to the encrypted text.



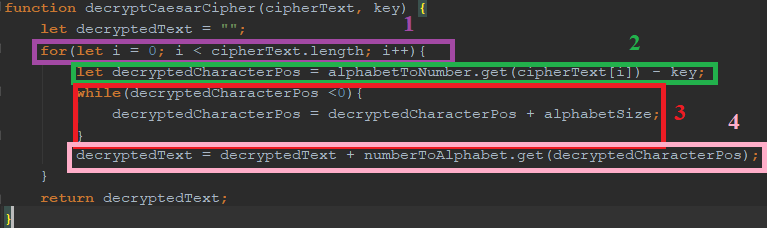
**The decryption function:**

1. Start with an empty decrypted text. Consider every character in the cipher text.

2. For each character in the cipher text, we map it to the corresponding position in the alphabet and then we substract the key.

3. If the key is big enaugh, it can get the position of the decrypted letter out of the bounds of the alphabet. So in order to ensure that the position of the decrypted letter is in the bounds of the alphabet, we keep adding the alphabet size untill the position is bigger or equal to 0.

4. We map the position of the decrypted character into the caracter itself and we add it to the decrypted text.



**How does the application look**

