10/20/2025

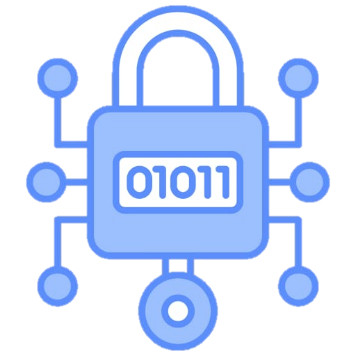
Caesar Cipher

Encryption/Decryption

Written by: **Eman Fatima**

**Caesar Cipher Implementation Assignment**

### ****Introduction:****

The Caesar Cipher is one of the simplest and most well-known encryption techniques. It is a type of **substitution cipher** in which each letter in the plaintext is replaced by another letter that is a fixed number of positions away in the alphabet.  
For example, with a shift of 3:

* A → D
* B → E
* C → F

This web application implements the Caesar Cipher using **HTML**, **CSS**, and **JavaScript**, allowing users to encrypt and decrypt text directly in the browser.

It is named after Julius Caesar, a Roman leader who used it around 58–50 BCE to send private military messages. He would shift the letters of the alphabet by a fixed number so that only his generals, who knew the key, could understand the message.

### ****Objective:****

The main goal of this project is to:

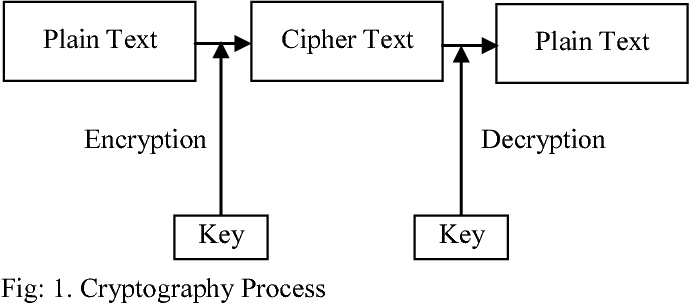
* Build an interactive web page for Caesar Cipher encryption and decryption.
* Understand the logic of character shifting in JavaScript.
* Provide a simple user interface (UI) using HTML and CSS for user input and output.

**System Design:**

The Caesar Cipher system follows a three-layered architecture:

1. **Input Layer**: User provides plaintext and key.
2. **Processing Layer**: Algorithm performs encryption or decryption.
3. **Output Layer**: Displays the result.

**Algorithm**



**How the Caesar Cipher Works?**

The Caesar Cipher works by shifting the letters of the alphabet by a certain number. This number is called the shift value or key.

In **encryption**, each letter in the original message (called plaintext) is replaced by another letter that comes after it in the alphabet.

In **decryption**, we do the opposite — we shift each letter backward by the same number to get the original message.

For example, with a shift of 3:

* A → D
* B → E
* C → F and so on…

**Example:**

Plaintext (original message): HELLO

Shift: 3

Cipher Text (encrypted message): KHOOR

Here’s how:

* H → K
* E → H
* L → O
* L → O
* → R

**Decryption:**

To decrypt (get the original message back), you simply shift the letters backward by the same number.

**For example**:

KHOOR → HELLO (shift back by 3)

**Implementation:**

Tools that are used for implementation of Caesar cipher are as below:

* Frontend: HTML, CSS, JavaScript
* IDE: Visual Studio
* Browser: Chrome/edge

### ****JavaScript Explanation****

The main functionality is handled by the script section at the end of the HTML file.

**The Caesar Cipher Function:**

function caesarCipher (text, shift) {

shift = shift % 26;

let result = '';

for (let char of text) {

if (char.match(/[a-z]/i)) {

const isUpper = char === char.toUpperCase();

const base = isUpper ? 'A'.charCodeAt(0) : 'a'.charCodeAt(0);

const charCode = char.charCodeAt(0);

const shiftedCharCode = ((charCode - base + shift + 26) % 26) + base;

result += String.fromCharCode(shiftedCharCode);

} else {

result += char;

}

}

return result;

**Explanation:**

1. **shift = shift % 26** ensures the shift value stays within the alphabet range (0–25).
2. A loop **(for...of)** goes through each character in the text.
3. **char.match(/[a-z]/i)** checks if the character is a letter (ignoring case).
4. It distinguishes between uppercase and lowercase letters using **isUpper.**
5. It converts each character to its ASCII code using **char. charCodeAt (0).**
6. The shifted character is calculated and wrapped around the alphabet using **modulus % 26**.
7. Non-alphabetic characters (like spaces, punctuation, or numbers) remain unchanged.
8. The shifted characters are combined into the result string.

**Encrypt Button Logic:**

document.getElementById('encryptBtn').addEventListener('click', () => {

const text = document.getElementById('inputText').value;

const shift = parseInt(document.getElementById('shift').value, 10) || 0;

const encrypted = caesarCipher(text, shift);

document.getElementById('outputText').value = encrypted;

});

When the **Encrypt** button is clicked:

* It retrieves the input text and shift value.
* Calls caesarCipher(text, shift) to encrypt the text.
* Displays the encrypted text in the output area.

#### **Decrypt Button Logic:**

document.getElementById('decryptBtn').addEventListener('click', () => {

const text = document.getElementById('inputText').value;

const shift = parseInt(document.getElementById('shift').value, 10) || 0;

const decrypted = caesarCipher(text, -shift);

document.getElementById('outputText').value = decrypted;

});

When the **Decrypt** button is clicked:

* It again gets the input text and shift value.
* Calls the same function caesarCipher but with a **negative shift** (to reverse the encryption).
* Displays the decrypted result in the output area.

### ****Working Example:****

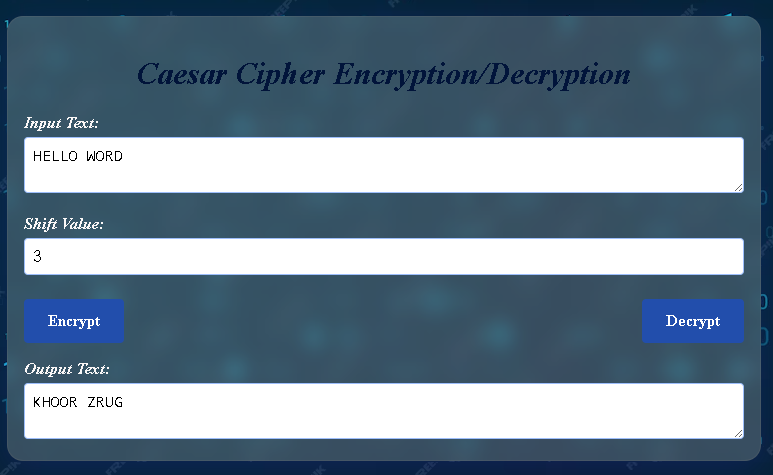
If the input text is:

* HELLO WORLD
* shift value is **3**,

then encryption gives:

* KHOOR ZRUOG

Decryption (with the same shift) converts it back to:

* HELLO WORLD

### ****Conclusion:****

This project demonstrates how the **Caesar Cipher** algorithm can be implemented using simple web technologies. It shows:

* The power of JavaScript for performing text manipulation.
* The role of HTML and CSS in creating an interactive and user-friendly interface.
* How event handling connects user actions (button clicks) to functional logic.

The Caesar Cipher is not secure for modern encryption needs, but it is an excellent educational example for learning about substitution ciphers, loops, string handling, and modular arithmetic in programming.