/\*

Day 05 and Day 04

For loop

For of loop

/\*

**for...of**

The **for...of** statement executes a loop that operates on a sequence of values sourced from an [iterable object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Iteration_protocols#the_iterable_protocol). Iterable objects include instances of built-ins such as [Array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array), [String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String), [TypedArray](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/TypedArray), [Map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map), [Set](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set), [NodeList](https://developer.mozilla.org/en-US/docs/Web/API/NodeList) (and other DOM collections), as well as the [arguments](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments) object, [generators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Generator) produced by [generator functions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function*), and user-defined iterables.

\*/

For in

/\*

**for...in**

The **for...in** statement iterates over all [enumerable string properties](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Enumerability_and_ownership_of_properties) of an object (ignoring properties keyed by [symbols](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol)), including inherited enumerable properties.

const object = { a: 1, b: 2, c: 3 };

for (const property in object) {

console.log(`${property}: ${object[property]}`);

}

// Expected output:

// "a: 1"

// "b: 2"

// "c: 3"

### [Parameters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in#parameters)

variable

Receives a string property name on each iteration. May be either a declaration with [const](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const), [let](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let), or [var](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/var), or an [assignment](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Assignment) target (e.g. a previously declared variable or an object property).

object

Object whose non-symbol enumerable properties are iterated over.

statement

A statement to be executed on every iteration. May reference variable. You can use a [block statement](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/block) to execute multiple statements.

\*/

Plain to Cypher text

Cypher to Plain text

// encryptor

let alphabet = "abcdefghijklmnopqrstuvwxyz";

let plain\_text = "mjgfajtauppatipsuaupacfatfsjpvtabmmauifaujnf";

key = 1;

let encoded = "";

for (let letter of plain\_text) {

position = (alphabet.indexOf(letter) + key+26) % 26;

encoded += alphabet[position];

}

console.log(`encrypted text: ${encoded}`);

/\*\*

If(alphabet.indexOf(letter) == -1) {

Encoded+=letter;

Continue;

}

Or

If(alphabet.includes(letter)){

Encoded+=letter;

Continue;

}

// Decryptor

let alphabet = "abcdefghijklmnopqrstuvwxyz";

let plain\_text = "mjgfajtauppatipsuaupacfatfsjpvtabmmauifaujnf";

key = 1;

let encoded = "";

for (let letter of plain\_text) {

position = (alphabet.indexOf(letter) - key+26) % 26;

encoded += alphabet[position];

}

console.log(`encrypted text: ${encoded}`);

//\*\*

Sheraz code for cypher

//let prompt = require("prompt-sync")();

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let user\_name = "sheraz"//prompt("Please Enter Plain Text here to Convert it into Cypher : " );

let user\_key = 2;//prompt("Please Enter Key size : " );

let mycypher = "";

for(let mychar of user\_name){

for(let i =0; i<=string\_range.length;i++){

if(mychar == string\_range[i]){

mycypher = mycypher + string\_range[i + user\_key];

}

}

}

console.log("Cyphered Message is : " + " " + mycypher);

//\*\*

Decoder

//let prompt = require("prompt-sync")();

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let user\_name = "sheraz"//prompt("Please Enter Plain Text here to Convert it into Cypher : " );

let user\_key = 2;//prompt("Please Enter Key size : " );

let mycypher = "";

for(let mychar of user\_name){

for(let i =0; i<=string\_range.length;i++){

if(mychar == string\_range[i]){

mycypher = mycypher + string\_range[i - user\_key];

}

}

}

console.log("Decoded Message is : " + " " + mycypher);

//\*\* this code has erros below this we have corrected version

//let prompt = require("prompt-sync")();

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let user\_name = "zhisziszme"//prompt("Please Enter Plain Text here to Convert it into Cypher : " );

let user\_key = 2;//prompt("Please Enter Key size : " );

let mycypher = "";

for(let mychar of user\_name){

for(let i =0; i<=string\_range.length;i++){

if(mychar == 'z'){ mychar = string\_range[0];}

if(string\_range.indexOf(mychar) == -1){

mycypher = mycypher + mychar;

continue;

}

if(mychar == string\_range[i]){

mycypher = mycypher + string\_range [i - user\_key];

}

}

}

console.log("Cyphered Message is : " + " " + mycypher);

\*/

/\*

//improved and corrected code: for encryption

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let user\_name = "zhisziszme"//prompt("Please Enter Plain Text here to Convert it into Cypher : " );

let user\_key = 2;//prompt("Please Enter Key size : " );

let mycypher = "";

for(let mychar of user\_name){

if (string\_range.indexOf(mychar.toLowerCase()) === -1){

mycypher += mychar;

}else{

let shift = (string\_range.indexOf(mychar.toLowerCase()) + user\_key) % 26;

mycypher += string\_range[shift];

}

}

console.log("Cyphered Message is : " + " " + mycypher);

/\*

* I added **.toLowerCase()** to convert any capital letter to lowercase before checking if it's in the string\_range array and also when finding the index of the letter in the array.
* I added a check to see if the character is not in the string\_range array, if so, it is just appended to the cyphertext without shifting it.
* I used the modulus operator to wrap the index around the length of the array to handle user\_key greater than 25
* I removed the unnecessary if block "if(mychar == 'z'){ mychar = string\_range[0];}", as it's already handled by the modulus operator.
* I removed the continue statement inside the if block "if(string\_range.indexOf(mychar) == -1)" and put it outside the loop.

This code should work correctly, handling the case of capital letters and any other special characters and also handling user\_key greater than 25.

\*/

//improved decoder

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let cypher\_text = "bkkujkujog"//prompt("Please Enter Cypher Text here to Convert it into Plain Text : " );

let user\_key = 2;//prompt("Please Enter Key size : " );

let plain\_text = "";

for(let mychar of cypher\_text){

if (string\_range.indexOf(mychar.toLowerCase()) === -1){

plain\_text += mychar;

}else{

let shift = (string\_range.indexOf(mychar.toLowerCase()) - user\_key + 26) % 26;

plain\_text += string\_range[shift];

}

}

console.log("Plain Text is : " + " " + plain\_text);

\*/

* /\*I changed the variable name from **user\_name** to **cypher\_text** and **mycypher** to **plain\_text** to reflect the change in functionality.
* I changed the operation in the for loop from adding the user\_key to subtracting it.
* I added **+26** in the calculation for shift, this is to handle negative numbers which can occur when subtraction of user\_key is greater than the index of character.

This code should now correctly decode the cyphertext back to the original plaintext when given the appropriate user\_key.

\*/

/\*

//improved special characters and space avoid ancoder

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let user\_name = "vjku ku vguv#."//prompt("Please Enter Plain Text here to Convert it into Cypher : " );

let user\_key = 2;//prompt("Please Enter Key size : " );

let mycypher = "";

for(let mychar of user\_name){

if(mychar.match(/[^a-zA-Z0-9]/)) {

mycypher += mychar;

} else if(mychar === ' ') {

mycypher += mychar;

} else {

let shift = (string\_range.indexOf(mychar.toLowerCase()) - user\_key + 26) % 26;

mycypher += string\_range[shift];

}

}

console.log("Cyphered Message is : " + " " + mycypher);

/\*\*

* I added a new condition to check if the current character is a special character or not by using the match method with a regular expression that only matches characters that are not letters or digits.
* The first condition checks if the current character is a special character or not by using the match() method with a regular expression that only matches characters that are not letters or digits.
* The second condition checks if the current character is a space or not.
* If it is not an alphabet or digit, it adds the character to the cypher without any change.
* Also, when shifting the letters I added 26 to the result of the shift calculation to prevent negative numbers.

\*/

\*/

/\*

//encoder code without regex with special character checks

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let plain\_text = "hello world"//prompt("Please Enter Plain Text here to Convert it into Cypher : " );

let user\_key = 2;//prompt("Please Enter Key size : " );

let cypher\_text = "";

for(let mychar of plain\_text){

if(mychar.charCodeAt() < 65 || (mychar.charCodeAt() > 90 && mychar.charCodeAt() < 97) || mychar.charCodeAt() > 122) {

cypher\_text += mychar;

} else if(mychar === ' ') {

cypher\_text += mychar;

} else {

let shift = (string\_range.indexOf(mychar.toLowerCase()) + user\_key) % 26;

cypher\_text += string\_range[shift];

}

}

console.log("Encrypted Message is : " + " " + cypher\_text);

/\*

* This code is the same as the previous one, but with the shift calculation for encryption.
* Here, instead of subtracting the key value from the index of the character, it adds the key value to the index of the character and takes the modulus with 26 to keep it within the range of the alphabet.

\*/

\*/

/\*

//decoder

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let cypher\_text = "bklukluvklu"//prompt("Please Enter Cypher Text here to Convert it into Plain Text : " );

let user\_key = 2;//prompt("Please Enter Key size : " );

let plain\_text = "";

for(let mychar of cypher\_text){

if(mychar.charCodeAt() < 65 || (mychar.charCodeAt() > 90 && mychar.charCodeAt() < 97) || mychar.charCodeAt() > 122) {

plain\_text += mychar;

} else if(mychar === ' ') {

plain\_text += mychar;

} else {

let shift = (string\_range.indexOf(mychar.toLowerCase()) + 26 - user\_key) % 26;

plain\_text += string\_range[shift];

}

}

console.log("Decrypted Message is : " + " " + plain\_text);

\*/

/\*/Decoder while we have no key

let string\_range = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

let cypher\_text = "jgnnq yqtnf"; // this is the cypher text you want to decode

let possible\_plain\_texts = [];

for(let i = 1; i <= 26; i++) {

let user\_key = i; // this is the key that was used to encode the original message

let plain\_text = "";

for(let mychar of cypher\_text){

if(mychar.charCodeAt() < 65 || (mychar.charCodeAt() > 90 && mychar.charCodeAt() < 97) || mychar.charCodeAt() > 122) {

plain\_text += mychar;

} else if(mychar === ' ') {

plain\_text += mychar;

} else {

let shift = (string\_range.indexOf(mychar.toLowerCase()) - user\_key + 26) % 26;

plain\_text += string\_range[shift];

}

}

possible\_plain\_texts.push(plain\_text);

}

console.log("Possible Decoded Messages : " + " " + possible\_plain\_texts);

/\*

This code is a simple implementation of the Caesar Cipher decryption technique, but it tries all possible keys (from 1 to 26) and generates all possible plain text messages. The cypher text message is passed through a loop where each character of the message is checked. If the character is a non-alphabetic character or a space, it is added to the plain text without any modification. Otherwise, the character is shifted by the negative of the user-specified key value (from 1 to 26) and its corresponding shifted character from the string\_range array is added to the plain text. The final plain text is then logged to the console. This process is repeated for all possible keys and all possible plain text messages are stored in an array **possible\_plain\_texts**. It will give you all possible plain text messages, but you need to check which one is correct.

-\*/

\*/