

The ones in red are my solutions

Strings

In JavaScript, the textual data is stored as strings. There is no separate type for a single character.

Quotes

Let's remember the kinds of quotes.

Strings can be enclosed either with the single, double quotes or in backticks:

```
let single = 'single-quoted';  
let double = "double-quoted";  
let backticks = `backticks`;
```

Single and double quotes are essentially the same. Backticks allow to embed any expression into the string, including function calls:

```
function sum(a, b) {  
  return a + b;  
}
```

```
alert(`1 + 2 = ${sum(1, 2)}.`); // 1 + 2 = 3.
```

Another advantage of using backticks is that they allow a string to span multiple lines:

```
let guestList = `Guests:  
* John  
* Pete  
* Mary  
`;  
;
```

```
alert(guestList); // a list of guests, multiple lines
```

If we try to use single or double quotes the same way, there will be an error:

```
let guestList = "Guests: // Error: Unexpected token ILLEGAL  
* John";
```

Single and double quotes come from ancient times of language creation, and the need for multiline strings was not taken into account. Backticks appeared much later and thus are more versatile.

Special characters

It is still possible to create multiline strings with single quotes, using a so-called “newline character” written as `\n`, that denotes a line break:

```
let guestList = "Guests:\n * John\n * Pete\n * Mary";
```

```
alert(guestList); // a multiline list of guests
```

So to speak, these two lines describe the same:

```
alert( "Hello\nWorld" ); // two lines using a "newline symbol"
```

```
// two lines using a normal newline and backticks
```

```
alert( `Hello  
World` );
```

There are other, less common “special” characters as well, here’s the list:

| Character | Description |
|----------------------------|--|
| <code>\b</code> | Backspace |
| <code>\f</code> | Form feed |
| <code>\n</code> | New line |
| <code>\r</code> | Carriage return |
| <code>\t</code> | Tab |
| <code>\uNNNN</code> | A unicode symbol with the hex code NNNN, for instance <code>\u00A9</code> – is a unicode for the copyright symbol ©. Must be exactly 4 hex digits. |
| <code>\u{NNNNN NNN}</code> | Some rare characters are encoded with two unicode symbols, taking up to 4 bytes. The long unicode requires braces around. |

Examples with unicode:

```
alert( "\u00A9" ); // ©
```

```
alert( "\u{20331}" ); // 儅, a rare chinese hieroglyph (long unicode)
```

```
alert( "\u{1F60D}"); // a smiling face symbol (another long unicode)
```

All special characters start with a backslash character \. It is also called an “escaping character”.

We should also use it if we want to insert the quote into the string.

For instance:

```
alert( '\m the Walrus!'); // I'm the Walrus!
```

See, we have to prepend the inner quote by the backslash \, because otherwise it would mean the string end.

Of course, that refers only for the quotes that are same as the enclosing ones. So, as a more elegant solution, we could switch to double quotes or backticks instead:

```
alert( `I'm the Walrus!` ); // I'm the Walrus!
```

Note that the backslash \ serves for the correct reading of the string by JavaScript, then disappears. The in-memory string has no \. You can clearly see that in alert from the examples above.

But what if we need exactly a backslash \ in the string?

That's possible, but we need to double it like \\:

```
alert( `The backslash: \\` ); // The backslash: \
```

[String length](#)

The length property has the string length:

```
alert( `My\n`.length ); // 3
```

Note that \n is a single “special” character, so the length is indeed 3.

length is a property

People with background in some other languages sometimes mistype by calling `str.length()` instead of just `str.length`. That doesn't work.

Please note that `str.length` is a numeric property, not a function. There is no need to add brackets after it.

[Accessing characters](#)

To get a character at position `pos`, use square brackets `[pos]` or call the method [str.charAt\(pos\)](#). The first character starts from the zero position:

```
let str = `Hello`;
```

```
// the first character
```

```
alert( str[0] ); // H
```

```
alert( str.charAt(0) ); // H
```

```
// the last character
```

```
alert( str[str.length - 1] ); // o
```

The square brackets is a modern way of getting a character, while `charAt` exists mostly for historical reasons.

The only difference between them is that if no character found, `[]` returns `undefined`, and `charAt` returns an empty string:

```
let str = `Hello`;
```

```
alert( str[1000] ); // undefined
```

```
alert( str.charAt(1000) ); // " (an empty string)
```

Strings are immutable

Strings can't be changed in JavaScript. It is impossible to change a character.

Let's try to see that it doesn't work:

```
let str = 'Hi';
```

```
str[0] = 'h'; // error
```

```
alert( str[0] ); // doesn't work
```

The usual workaround is to create a whole new string and assign it to `str` instead of the old one.

For instance:

```
let str = 'Hi';
```

```
str = 'h' + str[1]; // replace the string
```

```
alert( str ); // hi
```

In the following sections we'll see more examples of that.

Changing the case

Methods `toLowerCase()` and `toUpperCase()` change the case:

```
alert( 'Interface'.toUpperCase() ); // INTERFACE
```

```
alert( 'Interface'.toLowerCase() ); // interface
```

Or, if we want a single character lowercased:

```
alert( 'Interface'[0].toLowerCase() ); // 'i'
```

Searching for a substring

There are multiple ways to look for a substring in a string.

str.indexOf

The first method is `str.indexOf(substr, pos)`.

It looks for the substr in str, starting from the given position pos, and returns the position where the match was found or -1 if nothing can be found.

For instance:

```
let str = 'Widget with id';

alert( str.indexOf('Widget') ); // 0, because 'Widget' is found at the beginning
alert( str.indexOf('widget') ); // -1, not found, the search is case-sensitive

alert( str.indexOf("id") ); // 1, "id" is found at the position 1 (..idget with id)
```

The optional second parameter allows to search starting from the given position.

For instance, the first occurrence of "id" is at the position 1. To look for the next occurrence, let's start the search from the position 2:

```
let str = 'Widget with id';

alert( str.indexOf('id', 2) ) // 12
```

If we're interested in all occurrences, we can run indexOf in a loop. Every new call is made with the position after the previous match:

```
let str = 'As sly as a fox, as strong as an ox';

let target = 'as'; // let's look for it

let pos = 0;
while (true) {
  let foundPos = str.indexOf(target, pos);
  if (foundPos == -1) break;

  alert( `Found at ${foundPos}` );
  pos = foundPos + 1; // continue the search from the next position
}
```

The same algorithm can be layed out shorter:

```
let str = "As sly as a fox, as strong as an ox";
let target = "as";
```

```
let pos = -1;
while ((pos = str.indexOf(target, pos + 1)) != -1) {
  alert( pos );
}
```

str.lastIndexOf(pos)

There is also a similar method [str.lastIndexOf\(pos\)](#) that searches from the end of the string to its beginning.

It would list the occurrences in the reverse way.

There is a slight inconvenience with indexOf in the if test. We can't put it in the if like this:

```
let str = "Widget with id";

if (str.indexOf("Widget")) {
  alert("We found it"); // doesn't work!
}
```

The alert in the example above doesn't show, because str.indexOf("Widget") returns 0 (meaning that it found the match at the starting position). Right, but it considers that to be false.

So, we should actually check for -1, like that:

```
let str = "Widget with id";

if (str.indexOf("Widget") != -1) {
  alert("We found it"); // works now!
}
```

[includes, startsWith, endsWith](#)

The more modern method [str.includes\(substr, pos\)](#) returns true/false depending on whether str has substr as its part.

It's the right choice if we need to test for the match, but don't need its position:

```
alert( "Widget with id".includes("Widget") ); // true
```

```
alert( "Hello".includes("Bye") ); // false
```

The optional second argument of str.includes is the position to start searching from:

```
alert( "Midget".includes("id") ); // true
```

```
alert( "Midget".includes("id", 3) ); // false, from position 3 there is no "id"
```

The methods [str.startsWith](#) and [str.endsWith](#) do exactly what they say:

```
alert( "Widget".startsWith("Wid") ); // true, "Widget" starts with "Wid"
alert( "Widget".endsWith("get") ); // true, "Widget" ends with "get"
```

Getting a substring

There are 3 methods in JavaScript to get a substring: substring, substr and slice.

str.slice(start [, end])

Returns the part of the string from start to (but not including) end.

For instance:

```
let str = "stringify";
alert( str.slice(0,5) ); // 'string', the substring from 0 to 5 (not including 5)
alert( str.slice(0,1) ); // 's', from 0 to 1, but not including 1, so only character at 0
```

If there is no second argument, then slice goes till the end of the string:

```
let str = "stringify";
alert( str.slice(2) ); // ringify, from the 2nd position till the end
```

Negative values for start/end are also possible. They mean the position is counted from the string end:

```
let str = "stringify";

// start at the 4th position from the right, end at the 1st from the right
alert( str.slice(-4, -1) ); // gif
```

str.substring(start [, end])

Returns the part of the string *between* start and end.

Almost the same as slice, but allows start to be greater than end.

For instance:

```
let str = "stringify";

// these are same for substring
alert( str.substring(2, 6) ); // "ring"
alert( str.substring(6, 2) ); // "ring"

// ...but not for slice:
alert( str.slice(2, 6) ); // "ring" (the same)
alert( str.slice(6, 2) ); // "" (an empty string)
```

Negative arguments are (unlike slice) not supported, they are treated as 0.

`str.substr(start [, length])`

Returns the part of the string from start, with the given length.

In contrast with the previous methods, this one allows to specify the length instead of the ending position:

```
let str = "stringify";  
alert( str.substr(2, 4) ); // ring, from the 2nd position get 4 characters
```

The first argument may be negative, to count from the end:

```
let str = "stringify";  
alert( str.substr(-4, 2) ); // gi, from the 4th position get 2 characters
```

Let's recap the methods to avoid any confusion:

| Method | selects... | negatives |
|------------------------------------|----------------------------------|------------------------|
| <code>slice(start, end)</code> | from start to end | allows negatives |
| <code>substring(start, end)</code> | between start and end | negative values mean 0 |
| <code>substr(start, length)</code> | from start get length characters | allows negative start |

Which one to choose?

All of them can do the job. Formally, `substr` has a minor drawback: it is described not in the core JavaScript specification, but in Annex B, which covers browser-only features that exist mainly for historical reasons. So, non-browser environments may fail to support it. But in practice it works everywhere.

The author finds himself using `slice` almost all the time.

[Comparing strings](#)

Strings are compared character-by-character, in the alphabet order.

Although, there are some oddities.

1. A lowercase letter is always greater than the uppercase:

```
alert( 'a' > 'Z' ); // true
```

2. Letters with diacritical marks are "out of order":

```
alert( 'Österreich' > 'Zealand' ); // true
```

That may lead to strange results if we sort country names. Usually people would await for Zealand to be after Österreich in the list.

To understand what happens, let's review the internal representation of strings in JavaScript.

All strings are encoded using [UTF-16](#). That is: each character has a corresponding numeric code. There are special methods that allow to get the character for the code and back.

`str.charCodeAt(pos)`; Returns the code for the character at position pos:

```
// different case letters have different codes
alert( "z".charCodeAt(0) ); // 122
alert( "Z".charCodeAt(0) ); // 90
```

`String.fromCharCode(code)`

Creates a character by its numeric code

```
alert( String.fromCharCode(90) ); // Z
```

We can also add unicode characters by their codes using `\u` followed by the hex code:

```
// 90 is 5a in hexadecimal system
alert( '\u005a' ); // Z
```

Now let's see the characters with codes 65..220 (the latin alphabet and a little bit extra) by making a string of them:

```
let str = "";

for (let i = 65; i <= 220; i++) {
  str += String.fromCharCode(i);
}

alert( str );
// ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~.,f,,
// ¡¢£¥¦§¨©ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜ
```

See? Capital character go first, then few special ones, then lowercase characters.

Now it becomes obvious why `a > Z`.

The characters are compared by their numeric code. The greater code means that the character is greater. The code for `a` (97) is greater than the code for `Z` (90).

- All lowercase letters go after uppercase letters, their codes are greater.
- Some letters like `Ö` stand apart from the main alphabet. Here, its code is greater than anything from `a` to `z`.

[Summary](#)

- There are 3 types of quotes. Backticks allow a string to span multiple lines and embed expressions.
- Strings in JavaScript are encoded using UTF-16.
- We can use special characters like `\n` and insert letters by their unicode using `\u...`
- To get a character: use `[]`.
- To get a substring: use `slice` or `substring`.
- To lowercase/uppercase a string: use `toLowerCase/toUpperCase`.
- To look for a substring: use `indexOf`, or `includes/startsWith/endsWith` for simple checks.
- To compare strings according to the language, use `localeCompare`, otherwise they are compared by character codes.

There are several other helpful methods in strings:


- `str.trim()` – removes (“trims”) spaces from the beginning and end of the string.
- `str.repeat(n)` – repeats the string `n` times.
- ...and others, see the [internet](#) for details.

Also strings have methods for doing search/replace with regular expressions. But that topic deserves a separate chapter, so we’ll return to that later.

Questions and Exercises

1. Write a function `ucFirst(str)` that returns the string `str` with the uppercased first character, for instance:

```
ucFirst("john") == "John";
```



```
function ucFirst(str)
{
    return str[0].toUpperCase()+str.substr(1);
}
```


```
console.log(ucFirst("john"));
```

```
function ucFirst(personsName)
{
  let firstLetter=personsName.charAt(0);
  let remainingString=personsName.substr(1);
  firstLetter=firstLetter.toUpperCase();
  let finalResult=firstLetter+remainingString
  return finalResult
}
```

2. Write a function checkSpam(str) that returns true if str contains 'buy or 'sell', otherwise false.

The function must be case-insensitive:

```
checkSpam('buy a car now') == true
checkSpam('free sell') == true
checkSpam("innocent rabbit") == false
```



```
function checkSpam(str)
{
  if(str.indexOf("buy")>=0 || str.indexOf("sell")>=0)
    {return true;}
  else
    {return false;}
}
```

```
console.log(checkSpam("buy a car"));
console.log(checkSpam("buys"));
function checkSpam(message)
{
  if (message.includes("buy") || message.includes("sell"))
```

```
{return true;}  
else  
{return false;}  
}
```

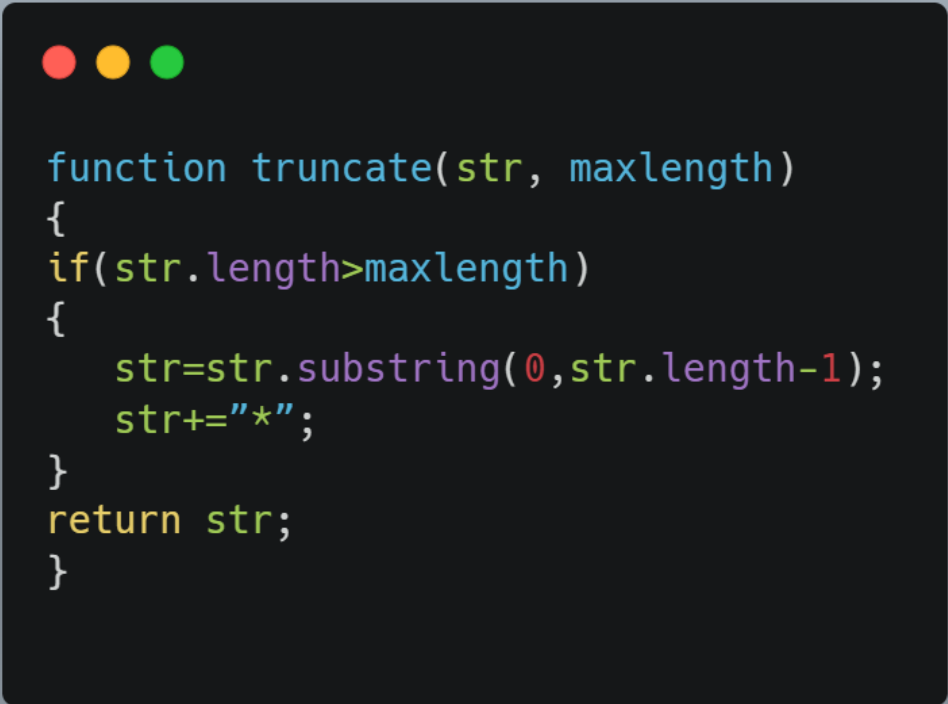
3. Create a function `truncate(str, maxLength)` that checks the length of the `str` and, if it exceeds `maxLength` – replaces the end of `str` with the character "*", to make its length equal to `maxLength`.

The result of the function should be the truncated (if needed) string.

For instance:

```
truncate("What I'd like to tell on this topic is:", 20) = "What I'd like to te*"
```

```
truncate("Hi everyone!", 20) = "Hi everyone!"
```



```
function truncate(str, maxLength)  
{  
  if(str.length>maxLength)  
  {  
    str=str.substring(0,str.length-1);  
    str+="*";  
  }  
  return str;  
}
```

```
console.log(truncate("He", 3));
```

```
function truncate(string, length)
{
  let stringLength=string.length
  if (string.length > length)
  {
    string=string.slice(0, length);
    string=string+"*";
  }
  return string;
}
```

4. We have a cost in the form "\$120". That is: the dollar sign goes first, and then the number.
Create a function `extractCurrencyValue(str)` that would extract the numeric value from such string and return it.

The example:

```
alert( extractCurrencyValue('$120') === 120 ); // true
```



```
function extractCurrencyValue(str)
{
    let newString=str.substr(1);
    let value=+newString;
    return value;
}
```

```
console.log(extractCurrencyValue("$120,000"));

function extractCurrencyValue(money)
{
    let noDollars=money.substr(1);
    return +noDollars;
}
```

More Problems...

Write a program to solve each of the following problems:

a) input a word. Isolate the first letter and output it as a capital.

```
//input word
let capital=word.charAt(0).toUpperCase();
//output capital;
```

b) input a sentence. Count the number of spaces and determine the number of words in the sentence.

```

//input sentence
let wordCount=1;
for(let x=0;x<sentence.length;x++)
{
    if(sentence.charAt(x)==" ")
    {
        wordCount++;
    }
}

//output wordCount

```

- c) input a sentence. Convert the sentence to Uppercase letters and output.

```

//input sentence
let answer=sentence.toUpperCase();
//output answer

```

- d) input a word. Create a new word containing the letters of the original word in the reverse order. For example, "bread" will become "daerb".

```

//input word
let result="";
for(let x=word.length-1;x>=0;x--)
{
    result+=word.charAt(x);
}
//output result

```

2. Write a program that prompts the user to enter a word using only uppercase letters. The program is to display the location of the first vowel in the word and the vowel itself. If the word does not contain a vowel, then an appropriate message should be output to the screen. For example,

Sample Input:

VISUAL

Sample Output:

The first vowel is "I" and it is in position 2.

```
//input sentence

let position=-1;

let letter="";

for(let x=0;x<sentence.length;x++)

{

    letter=sentence.charAt(x).toLowerCase();

    if((letter=="a" || letter=="e" || letter=="I" || letter=="o" || letter=="u" || letter=="y"))

    { position=x; break; }

}

//output position
```

3. Write a program which prompts the user to enter a word and a search pattern. The program is to display the word along with the beginning position of the search pattern in the word. Give an appropriate response if the pattern does not exist in the word.

Sample Input:

Enter a word:

KALAMAZOO

Enter a search pattern:


AZ

Sample Output:

The pattern "AZ" starts at position 6.



```
let sentence=prompt("Enter a sentence");  
let pattern=prompt("Enter a pattern");  
let position=sentence.indexOf(pattern);  
alert("Pattern is found at index "+position);
```



```

let userInputWord = prompt('Enter a word');
let userInputPattern = prompt('Enter a search pattern (make sure it
only contains letters)');

patternChecking(userInputWord, userInputPattern);

function patternChecking(word, pattern){

    if(word >= 'A' && word <= 'z'){
        if(word.includes(pattern)){
            let startingIndexOfPattern = word.indexOf(pattern);
            console.log('The pattern "' + pattern + '" starts at
position ' + (startingIndexOfPattern + 1) + '.');
        } else {
            console.log('It appears that the search pattern you
entered could not be found in the word.');

```

```
let words=prompt("Enter a word");
```

```
vowelFinder(words);
```

```
function vowelFinder(wordss)
```

```
{
```

```
    let position=" ";
```

```
    if (wordss.includes("A") || wordss.includes("E") || wordss.includes("I") || wordss.includes("O") ||
wordss.includes("U"))
```

```
{
```

```
    let A=0;
```

```
    let E=0;
```

```
    let I=0;
```

```
let O=0;

let U=0;

for (let x=0;x<wordss.length;x++)

{

    position=wordss.charAt(x);

    if (position=="A")

    {

        x++

        console.log("The first vowel is "+position+" and it is in position "+x);

        break;

    }

    else if (position=="E")

    {

        x++

        console.log("The first vowel is "+position+" and it is in position "+x);

        break;

    }

    else if (position=="I")

    {

        x++

        console.log("The first vowel is "+position+" and it is in position "+x);

        break;

    }

    else if (position=="O")

    {

        x++

        console.log("The first vowel is "+position+" and it is in position "+x);

        break;
```

```

    }

    else if (position=="U")

    {

        x++

        console.log("The first vowel is "+position+" and it is in position "+x);

        break;

    }

}

}

else

{

    console.log("Theres no vowels cuh")

}

}

```

4. Write a program that accepts a sentence from the user. The output is to be the frequency of each of the vowels used in the sentence. This should not be case sensitive.

Sample Input:

Enter a sentence: I love Visual Basic

Sample Output:

| VOWEL | FREQUENCY |
|-------|-----------|
|-------|-----------|

| | |
|---|---|
| A | 2 |
|---|---|

| | |
|---|---|
| E | 1 |
|---|---|

| | |
|---|---|
| I | 3 |
|---|---|

| | |
|---|---|
| O | 1 |
|---|---|

| | |
|---|---|
| U | 1 |
|---|---|



```
//input sentence
let a=0;
let e=0;
let i=0;
let o=0;
let u=0;
let y=0;
let letter="";
let sentence=prompt("Enter a sentence");
for(let x=0;x<sentence.length;x++)
{
    letter=sentence.charAt(x).toLowerCase();
    if(letter=="a"){a++;}
    if(letter=="e"){e++;}
    if(letter=="i"){i++;}
    if(letter=="o"){o++;}
    if(letter=="u"){u++;}
    if(letter=="y"){y++;}
}
alert("A's="+a+" E's="+e+" I's="+i+" O's="+o+" U's="+u+" Y's="+y);
```

```

let userInputSentence = prompt('Enter a sentence. ');

freqOfVowels(userInputSentence);

function freqOfVowels(sentence){
    let stringForA = 'aA';
    let stringForE = 'eE';
    let stringForI = 'iI';
    let stringForO = 'oO';
    let stringForU = 'uU';

    let counterForA = 0;
    let counterForE = 0;
    let counterForI = 0;
    let counterForO = 0;
    let counterForU = 0;

    for(let i = 0; i < sentence.length; i++){
        if(stringForA.includes(sentence.charAt(i)))
            {counterForA++;}
        else if(stringForE.includes(sentence.charAt(i)))
            {counterForE++;}
        else if(stringForI.includes(sentence.charAt(i)))
            {counterForI++;}
        else if(stringForO.includes(sentence.charAt(i)))
            {counterForO++;}
        else if(stringForU.includes(sentence.charAt(i)))
            {counterForU++;}
    }

    console.log("VOWEL\tFREQUENCY\n\n A\t\t\t" + counterForA + "\n\n
E\t\t\t" + counterForE + "\n\n I\t\t\t" + counterForI + "\n\n
O\t\t\t" + counterForO + "\n\n U\t\t\t" + counterForU);
}

```

5. Write a program to check the spelling rule “i before e except after c” for a word entered by the user. If “cie” occurs, rebuild the word with the proper spelling of “cei”. You may want to look at other special occurrences: for example “height” is not i before e even though there is no c.



```
let word=prompt("Enter a word");  
word=word.replaceAll("cie","cei");  
console.log(word);
```

```
let cie=prompt("Enter a word (CIE)");  
if (cie.includes(cie))  
{  
  cie=cie.replace("cie","cei")  
}  
console.log(cie);
```

6. Enhance #3 so that your program counts the number of occurrences of the pattern.

Sample Input:

Enter a word:

abracadabra

Enter a pattern: ab

Sample Output The pattern "ab" occurs 2 times.

```
let word=prompt("Enter the word.");  
  
let pattern=prompt("Enter a pattern");  
  
let occurrences=0  
  
for (let x=0;x<word.length;x+=pattern.length)  
{
```

```
        if (word.substr(x,pattern.length)==pattern) {occurrences++}

    }

    for (let x=1;x<word.length;x+=pattern.length)

    {

        if (word.substr(x,pattern.length)==pattern) {occurrences++}

    }

    console.log("The pattern "+pattern+" occurred "+occurrences+" times");
```



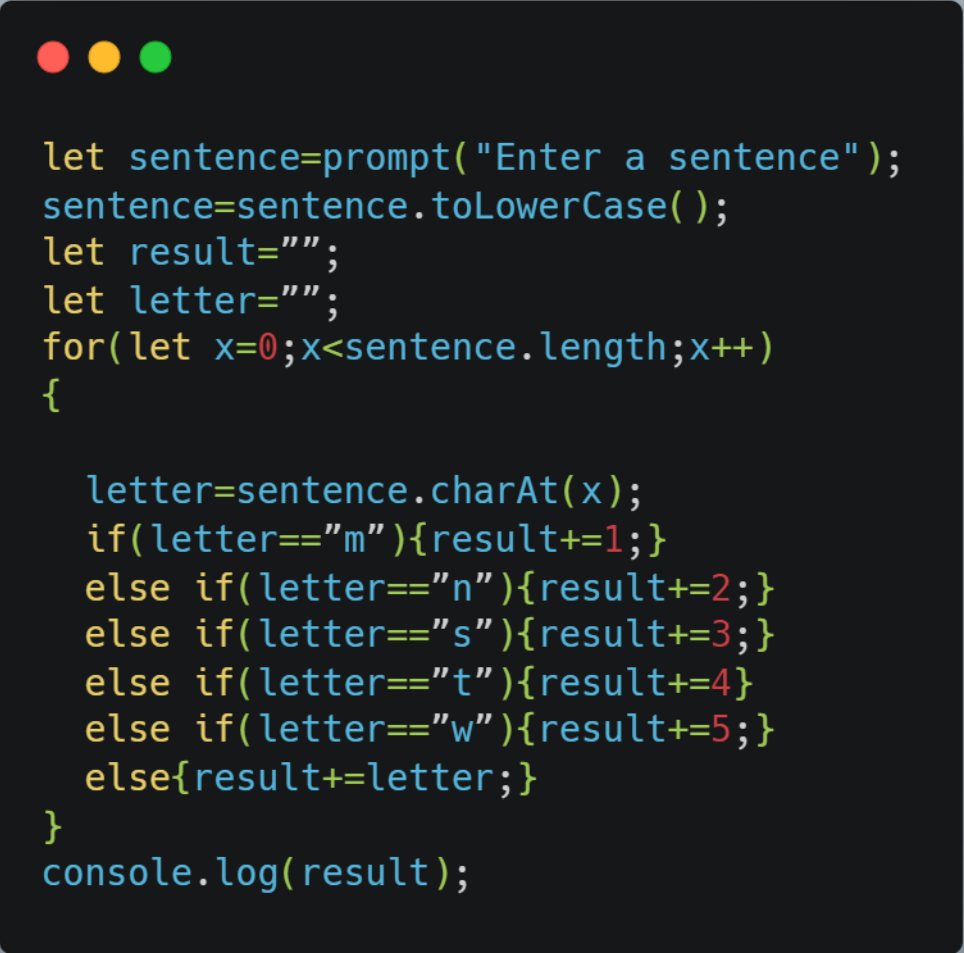
```
let result=0;
let word=prompt("Enter a word");
let pattern=prompt("Enter a pattern");
for(let x=0;x<(word.length-pattern.length);x++)
{
    if(word.substr(x,pattern.length)==pattern){result++;}
}
console.log("The pattern appears "+result+" times");
```



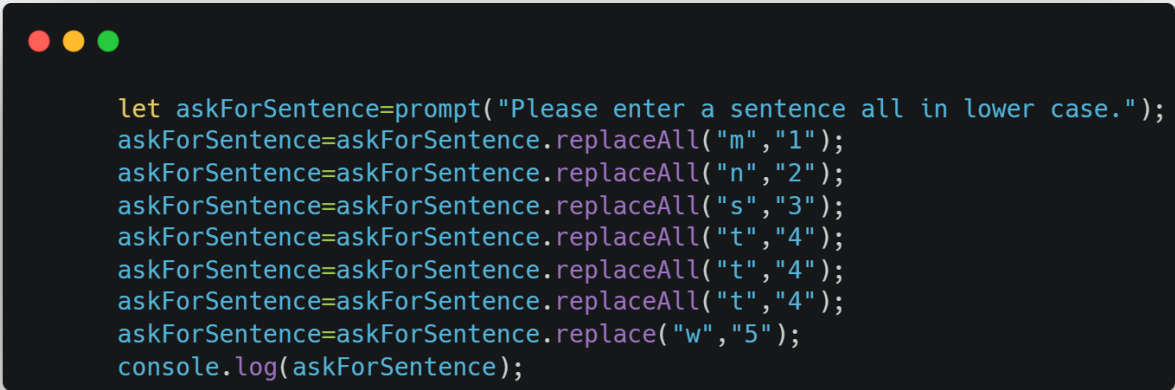

```
let string=prompt("Enter a string");
let pattern=prompt("Enter a search pattern");
let tracker=0;
let position=string.indexOf(pattern);
while(position!=-1)
{
    position=position+1;
    position=string.indexOf(pattern,position);
    tracker=tracker+1;
}
console.log(tracker);
```

7. Write a program that accepts a sentence from the user in lowercase letters only. The program will search for any of these letters m n s t w and replace these with the number corresponding to the position of the letter in the string "mnstw". For example, the sentence "visual basic is the best" would be coded "vi3ual ba3ic i3 4he be34" because the s is the 3rd

letter in the string "mnstw" and t is the 4th letter. The program will display the original sentence as well as the coded sentence.



```
let sentence=prompt("Enter a sentence");
sentence=sentence.toLowerCase();
let result="";
let letter="";
for(let x=0;x<sentence.length;x++)
{
    letter=sentence.charAt(x);
    if(letter=="m"){result+=1;}
    else if(letter=="n"){result+=2;}
    else if(letter=="s"){result+=3;}
    else if(letter=="t"){result+=4}
    else if(letter=="w"){result+=5;}
    else{result+=letter;}
}
console.log(result);
```



```
let askForSentence=prompt("Please enter a sentence all in lower case.");
askForSentence=askForSentence.replaceAll("m","1");
askForSentence=askForSentence.replaceAll("n","2");
askForSentence=askForSentence.replaceAll("s","3");
askForSentence=askForSentence.replaceAll("t","4");
askForSentence=askForSentence.replaceAll("t","4");
askForSentence=askForSentence.replaceAll("t","4");
askForSentence=askForSentence.replace("w","5");
console.log(askForSentence);
```

8. Write a function that returns the first character in a string.

```
function wordass(string)
{
    let word1=string.charAt(0);
    console.log(word1)
}
```



```
function sm(m)
{
  return m.charAt(0);
}
```

9. The caps lock key is broken! Write a function that will turn all capital letters to lowercase and all lowercase letters to uppercase in a given string.

For example, if the given string was `hI mOM!`, the output should be `Hi Mom!`

```
let caps=prompt("The Caps is broken. enter a word cuh");
```

```
let finalWord=" ";
```

```
for (let x=0;x<caps.length;x++)
```

```
{
```

```
if (caps.charCodeAt(x)<=90)
```

```
{
```

```
let letter=caps.charAt(x).toLowerCase();
```

```
finalWord+=letter;
```

```
}
```

```
else if(caps.charCodeAt(x)>=97)
```

```
{
```

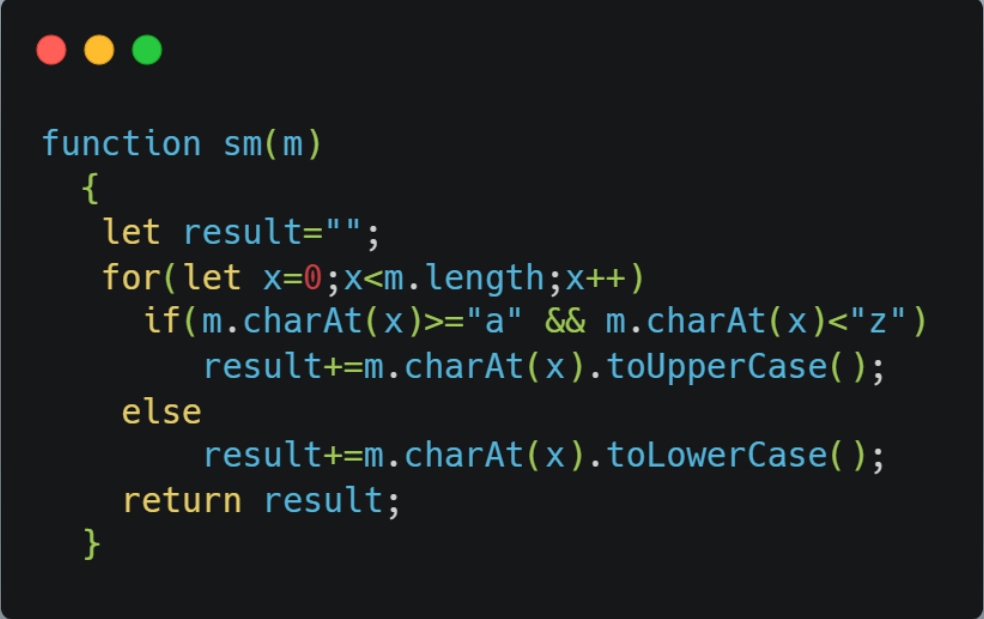
```
let letter=caps.charAt(x).toUpperCase();
```

```
finalWord+=letter;
```

```
}
```

```
}
```


```
console.log(finalWord);
```



```
function sm(m)
{
  let result="";
  for(let x=0;x<m.length;x++)
    if(m.charAt(x)>="a" && m.charAt(x)<"z")
      result+=m.charAt(x).toUpperCase();
    else
      result+=m.charAt(x).toLowerCase();
  return result;
}
```

10. Write a function that returns the length of any given string.

```
let strings=prompt("Enter a string");  
console.log(strings.length);
```

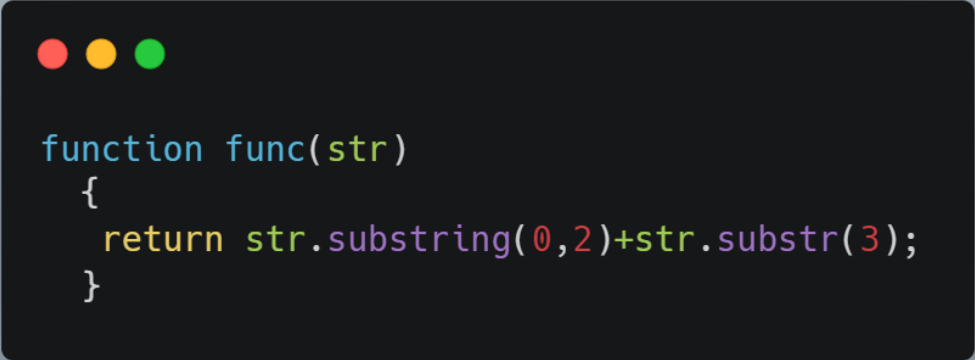


```
function func(str)  
{  
    return str.length;  
}
```

11. Write a function that returns a given string with the value found at the 2nd index removed. (Note: The string will always have at least 3 characters.)

```
let secondString=prompt("Enter a sting an we will remove the second value");  
valueRemoved(secondString);  
function valueRemoved (deString)  
{
```

```
console.log(deString.charAt(0)+deString.substr(2));  
}
```



```
function func(str)  
{  
  return str.substring(0,2)+str.substr(3);  
}
```

12. Write a function that returns the first and last letters of any given string. (Note: All strings will have at least 2 characters.)

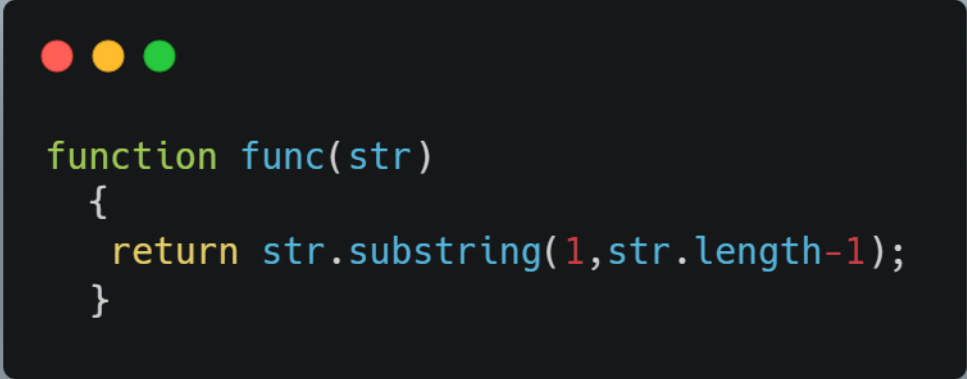


```
function func(str)  
{  
  return str.charAt(0)+str.charAt(str.length-1);  
}
```

13. Write a function that returns a string with the first and last letters removed. (Note: All strings will have at least 2 characters.)


```
firstLast();

function firstLast()
{
    let word=prompt("Enter your word");
    console.log(word.charAt(0)+word.charAt(word.length-1));
}
```



```
function func(str)
{
    return str.substring(1,str.length-1);
}
```

14. You are given 3-letter words that include a vowel in the center. Create a function that returns the word with the vowel replaced with an 'a'.

```
vowelReplacer();

function vowelReplacer()
{
    let wordss=prompt("Enter a 3 letter word")
    wordss=wordss.toUpperCase();
```

```

    if (wordss.includes("A") || wordss.includes("E") || wordss.includes("I") ||
wordss.includes("O") || wordss.includes("U"))
    {
        console.log(wordss.charAt(0)+"A"+wordss.charAt(2));
    }
}

```



```

function func(str)
{
    return str.charAt(0)+"a"+str.charAt(2);
}

```

15. Write a function that returns the character found at index 3. If no character exists at index 3, return 0.

```

index3());

function index3()
{
    let word=prompt("Enter a word and I will return what is found at index 3!");
    if (word.length>=2)
    {

```

```
    console.log(word.charAt(2));  
  }  
  else  
  {  
    console.log("0");  
  }  
}
```



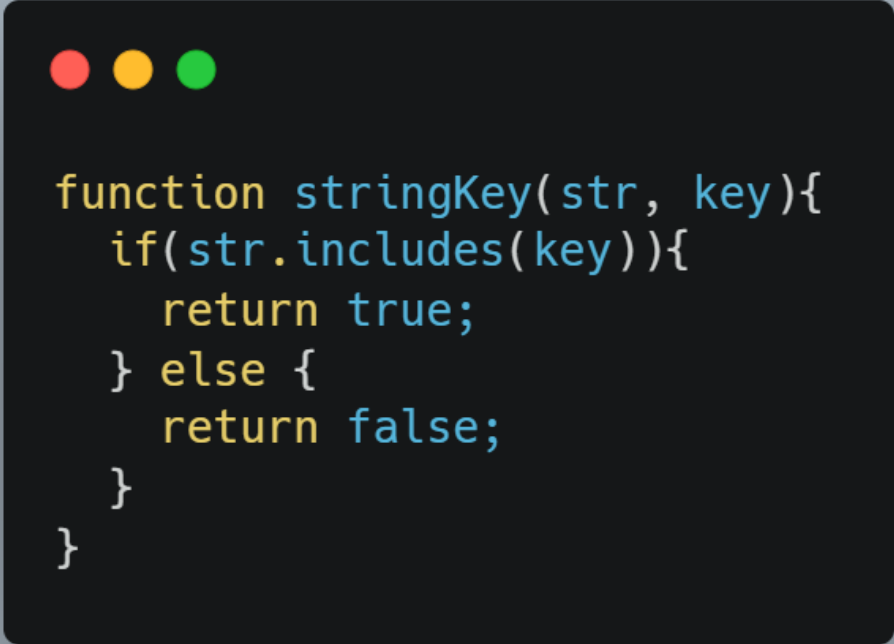
```
function index3(str){  
  if(str.length>=2){  
    return str.charAt(2);  
  } else {  
    return 0;  
  }  
}
```

16. Write a function that takes a string and a key as input. You should return true if the key exists in the index, false otherwise. Assume both the string and key have a length > 0.

Example:

`findKey("Hello Word", "l") --> true`

`findKey("Ursa Major", "bear") --> false`



```
function stringKey(str, key){  
  if(str.includes(key)){  
    return true;  
  } else {  
    return false;  
  }  
}
```

17. In this exercise, you are given a word or phrase. If the length of that word or phrase is even, add a space to the end and return it. If it is odd, just return it as is.

Example:

`makeOdd("Hi") --> "Hi "`

`makeOdd("odd") --> "odd"`



```
let phrase= prompt("Enter sentence/word");
console.log(wordLength(phrase));
function wordLength(word){
  if((word.length)%2==0){
    return word+" ";
  } else {
    return word;
  }
}
```

18. Write a function that will remove the final letter in a word if it is 'e', but only if the second to last letter is also an 'e'. If the final letter or second to last letter are not 'e's, return the string as is.

Example:

pleasee returns please

excite returns excite

computer returns computer



```
let userWord = prompt('Enter a word.');
```

```
removeE(userWord);
```

```
function removeE(word){  
  if(word.charAt(word.length-1) == 'e'){  
    if(word.charAt(word.length-2) == 'e'){  
      let newWord = word.substr(0, word.length-1);  
      console.log(newWord);  
    } else {  
      console.log(word);  
    }  
  }  
}
```



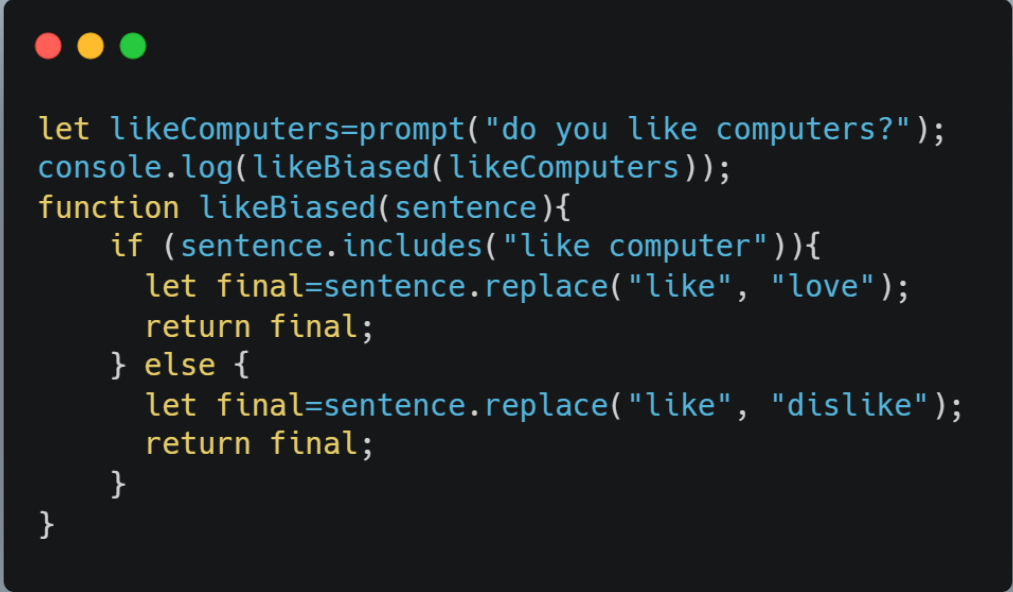
```
let wordE= prompt("enter your e word");  
console.log(doubleE(wordE));  
function doubleE(word){  
  if(word.charAt(word.length-1)=="e"){  
    if(word.charAt(word.length-2)=="e"){  
      return word.slice(0,word.length-1);  
    } else {  
      return word;  
    }  
  } else {  
    return word;  
  }  
}
```

19. In this exercise, you are going to replace the word 'like' with 'dislike', unless the word computer is found after the word like, in which case you will replace 'like' with 'love'.

Example:

```
replace_like("I like recess") --> "I dislike recess"
```

```
replace_like("I like computer science") --> "I love computer science"
```



```
let likeComputers=prompt("do you like computers?");
console.log(likeBiased(likeComputers));
function likeBiased(sentence){
  if (sentence.includes("like computer")){
    let final=sentence.replace("like", "love");
    return final;
  } else {
    let final=sentence.replace("like", "dislike");
    return final;
  }
}
```

20. In this exercise, you are given a word. You should return the string with two dashes added to the middle of the string. If the string is an even length, return both dashes in the middle together. If the string has an odd length, the two dashes should surround the middle letter.

Example:

```
split("even") --> "ev--en"
```

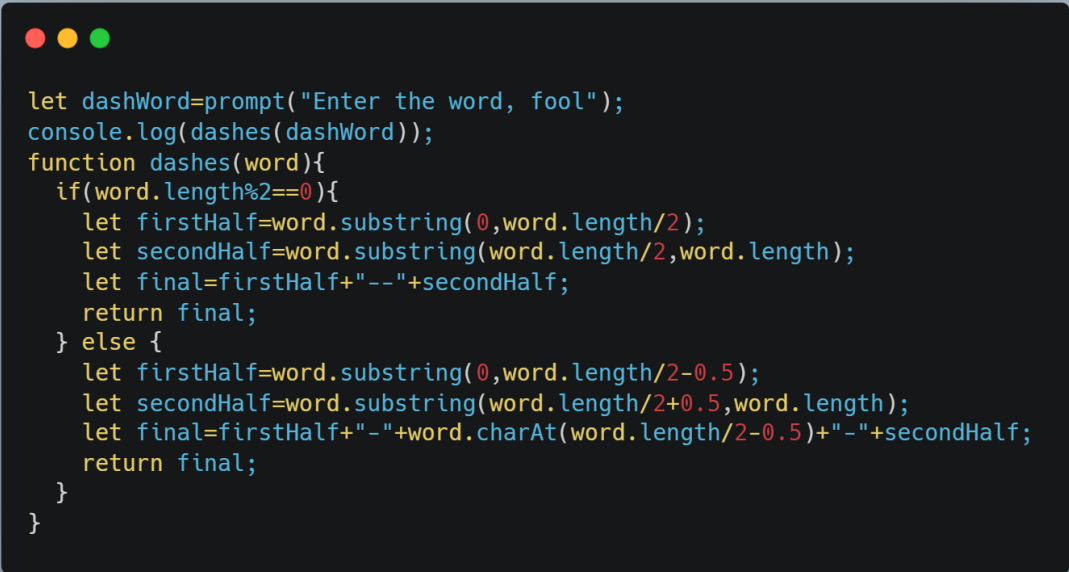
split("friends") --> "fri-e-nds"

```
let userWord = prompt('Enter a word');

console.log(addDashes(userWord));

function addDashes(word){
    let wordInAnArray = word.split('');
    let middle = 0;
    let addingDash = '';

    if(word.length%2 == 0){
        middle = Math.floor((wordInAnArray.length - 1) / 2);
        addingDash = wordInAnArray.splice(middle + 1, 0, '--');
        return wordInAnArray.join('');
    } else {
        middle = Math.floor(wordInAnArray.length / 2);
        addingDash = wordInAnArray.splice(middle, 0, '-');
        addingDash = wordInAnArray.splice(middle + 2, 0, '-');
        return wordInAnArray.join('');
    }
}
```

```

let dashWord=prompt("Enter the word, fool");
console.log(dashes(dashWord));
function dashes(word){
  if(word.length%2==0){
    let firstHalf=word.substring(0,word.length/2);
    let secondHalf=word.substring(word.length/2,word.length);
    let final=firstHalf+"--"+secondHalf;
    return final;
  } else {
    let firstHalf=word.substring(0,word.length/2-0.5);
    let secondHalf=word.substring(word.length/2+0.5,word.length);
    let final=firstHalf+"-"+word.charAt(word.length/2-0.5)+"-"+secondHalf;
    return final;
  }
}

```

21. Write a program that will accept as input a phrase, a code offset and a selection of encode or decode. The program will then encode or decode the text using the offset as a means of determining the letter substitution.

Example: If the offset is 3 then letter "A" becomes "D", 3 letters down the alphabet. Letter "Y" would become "B" as the alphabet would wrap around. The word "THE" would become the word "WKH". Your program must be able to handle a phrase, not just a word and should ignore any non letter characters.

THIS IS INCOMPLETE



```
let string="";
let offsetPhrase=prompt("Enter a phrase to offset");
for(let x=0; x<offsetPhrase.length;x++){
  let letter=offsetPhrase.charAt(x);
  if(letter=="x"||letter=="y"||letter=="z"){
    let num=letter.charCodeAt(0)-23;
    let newLetter= String.fromCharCode(num);
    string+=newLetter;
  } else {
    let num=letter.charCodeAt(0)+3;
    let newLetter= String.fromCharCode(num);
    string+=newLetter;
  }
}
console.log(string);
```