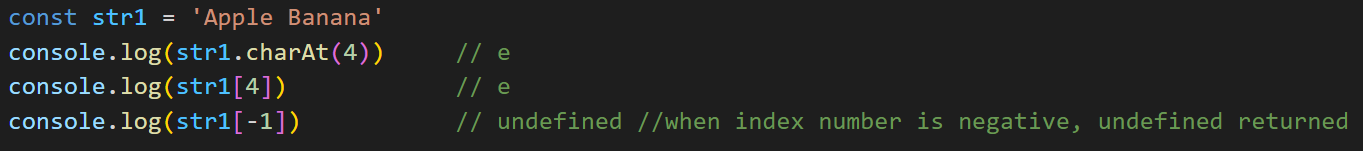
STRING METHODS

***“NONE OF THE BELOW STRING METHODS WILL MODIFY ORIGINAL STRING BECAUSE STRINGS ARE IMMUTABLE IN JAVASCRIPT”*  
  
String property – length**

* one and only property of string (rest are methods)
* returns a number (calculates number of characters in string)  
  A computer screen shot of a computer code

  Description automatically generated

**String Method – charAt()**

* *str.charAt(index)*
* used to access a certain character at a certain index in a string
* returns a character value at specified index
* returns undefined if index is negative
* *Alternative way to access a character – str[index]*  
  

**String Method - indexOf and lastIndexOf**

* *mainString.indexOf(searchstring, startindex:optional)*
* *mainString.lastIndexOf (searchstring, startindex:optional)*
* both methods perform a search operation over the string
* *indexOf*: parses the string from first index to last (index 0 to length-1)
* *lastIndexOf*: parses the string from last index to first (length -1 to 0)
* if searchstring pattern present in mainstring, index of “first” such occurrence is returned
* search operation is case-sensitive
* if search string not present, -1 returned
* if startindex is negative, it is auto-converted to 0
* mainstring.indexOf(searchstring,5): search operation will start from index = 5 to (mainstring.length -1) (L to R)
* mainstring.lastIndexOf(searchstring,5): search operation will start from index = 5 to 0 (R to L)  
  A screen shot of a computer program

  Description automatically generated

**String Method – slice**

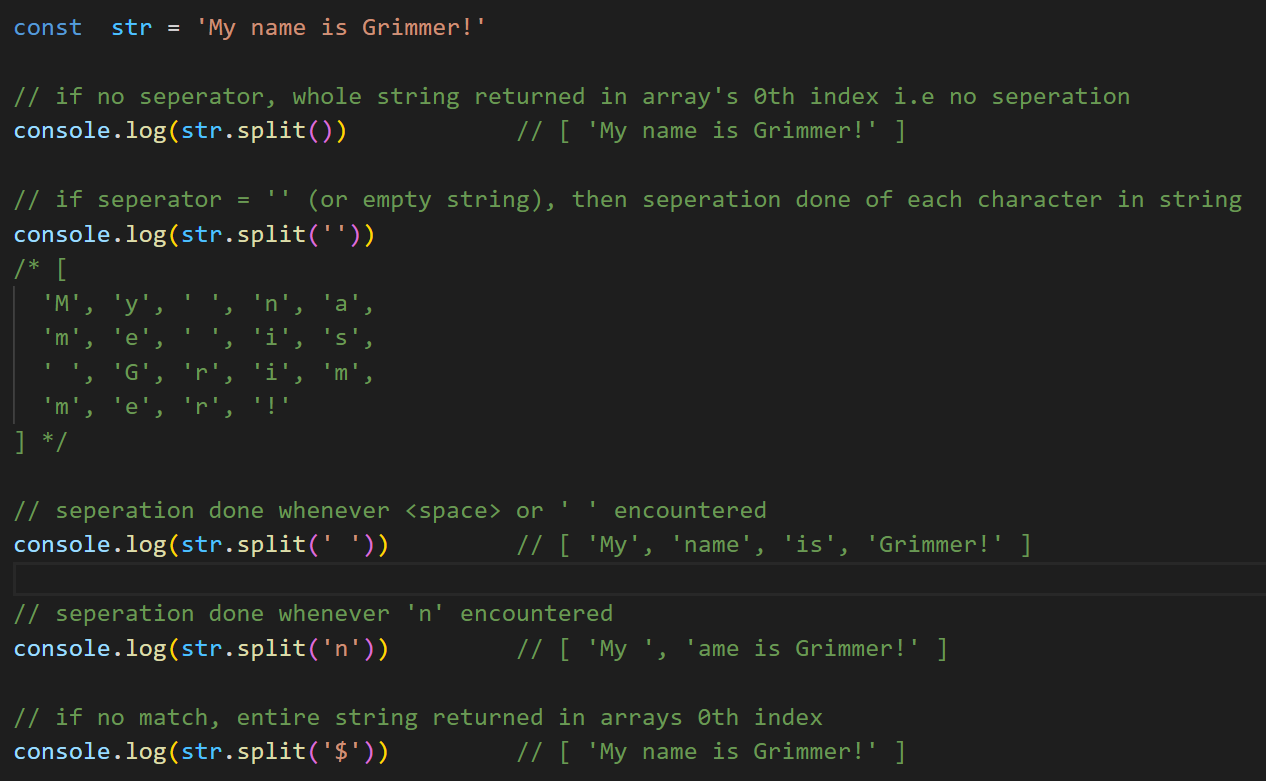
* *str.slice(startindex, stopindex:optional)*
* returns a “part” of a string
* string gets trimmed from index: [startindex,stopindex)
* if stopindex not defined, by default it is = string.length -1
* if startindex = stopindex, empty string returned
* if startindex > stopindex, empty string returned **A screen shot of a computer program

  Description automatically generated**
* startindex and stopindex can be negative (for more details refer code)
* Note: negative values of indexes work differently with slice ( -1 represents last index etc)

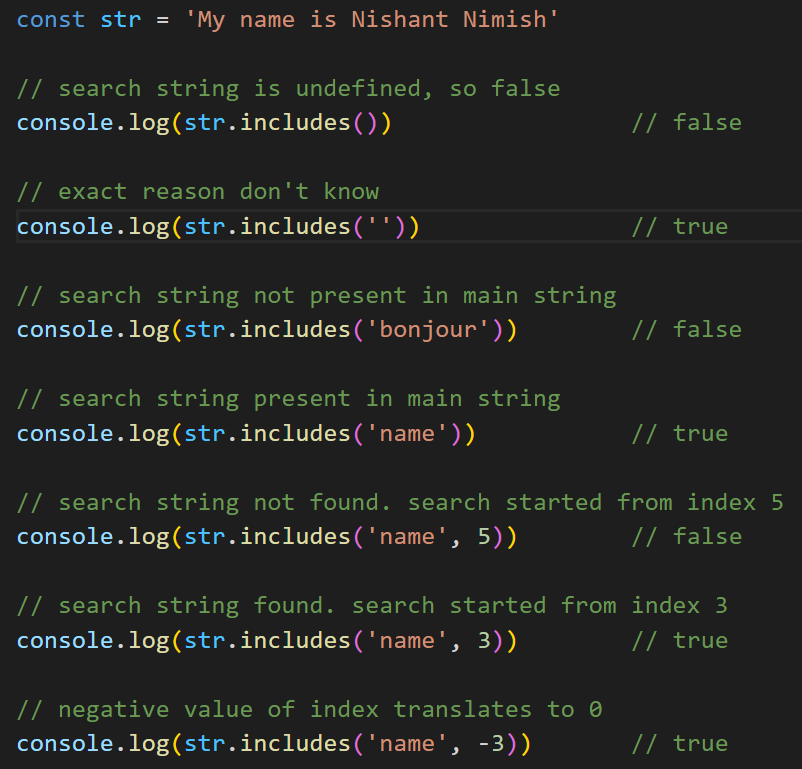
**String Method – substring**

* Same as slice with below major difference:
* if start > stop, indexes are swapped behind the scenes
* any negative value of index translates to 0

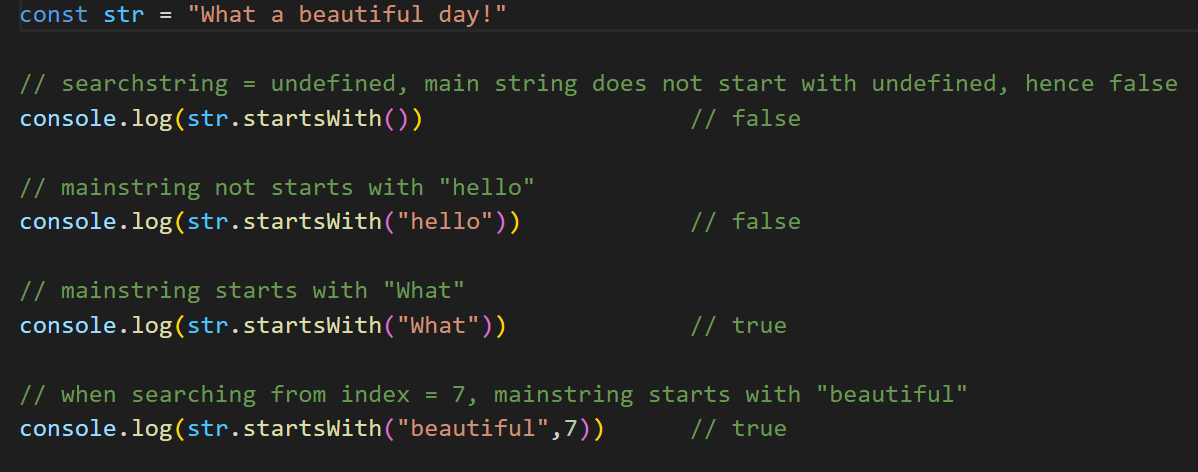
**String Method – split**

* *str.split(separator)*
* returns an array of strings
* the string is parsed and each time the separator appears in the string, left substring of separator is stored as new entry in array
* In simple words, an individual 'string' gets split into an 'array of strings' based on separator defined
* if no separator, whole string returned in array's 0th index i.e. no separation
* if no separator, whole string returned in array's 0th index i.e. no separation
* if separator = '' (or empty string), then separation done of each character in string (See screenshot)  
  ****

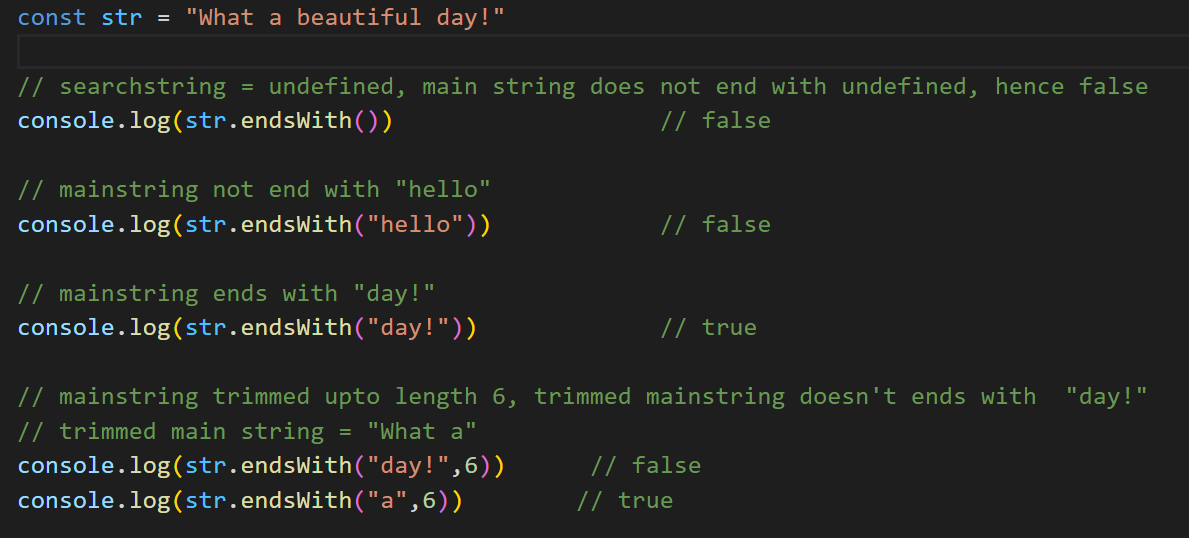
**String Method – includes**

* mainstring.includes(searchstring, startindex:*optional*)
* returns Boolean
* if searchstring present inside mainstring, return true, else false
* at least one occurrence is needed for result to be true
* negative value of startindex translates to 0 ****

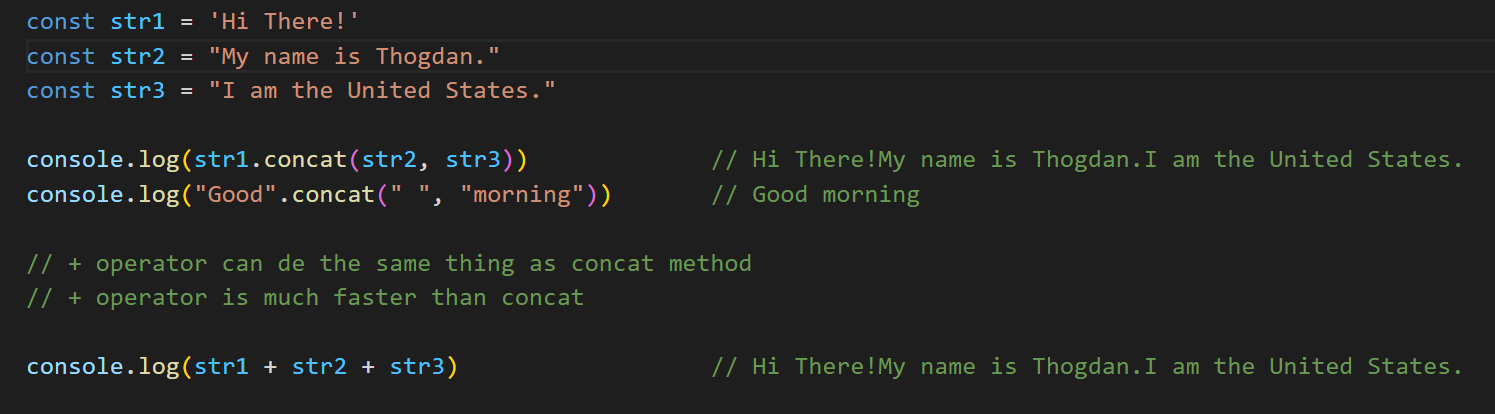
**String Method – startsWith**

* mainstring.startsWith(searchstring, startindex:*optional*)
* returns a Boolean
* verifies if the mainstring “starts with” searchstring
* verification will start from startindex, if not defined, by default = 0  
  ****

**String Method – endsWith**

* mainstring.endsWith(searchstring, length:*optional*)
* returns a Boolean
* verifies if the mainstring “ends with” searchstring
* if length defined, means mainstring trimmed to specified length and search operation performed on trimmed string
* by default, if not defined, length = mainstring.length  
  ****

**String Method – concat**

* mainstring.concat(string1,string2...)
* concatenate multiple strings into one
* returns a new string
* ***“+” operator is also used to concatenate strings. It is much faster than concat method***

**String Method – repeat**

* str.repeat(count)
* used to repeat a string ‘count’ number of times
* returns a string **A screenshot of a computer

  Description automatically generated**

**String Method – trim,trimStart,trimEnd**

(a) *str.trimStart()* - trim whitespace at the start of string  
 (b) *str.trimEnd()* - trim whitespace at the end of string

(c) *str.trim()* - trim whitespace at start and end of string

* whitespaces are trimmed at start and end of string BUT not in between
* returns a string **A screen shot of a computer

  Description automatically generated**

**String Method – padStart,padEnd**

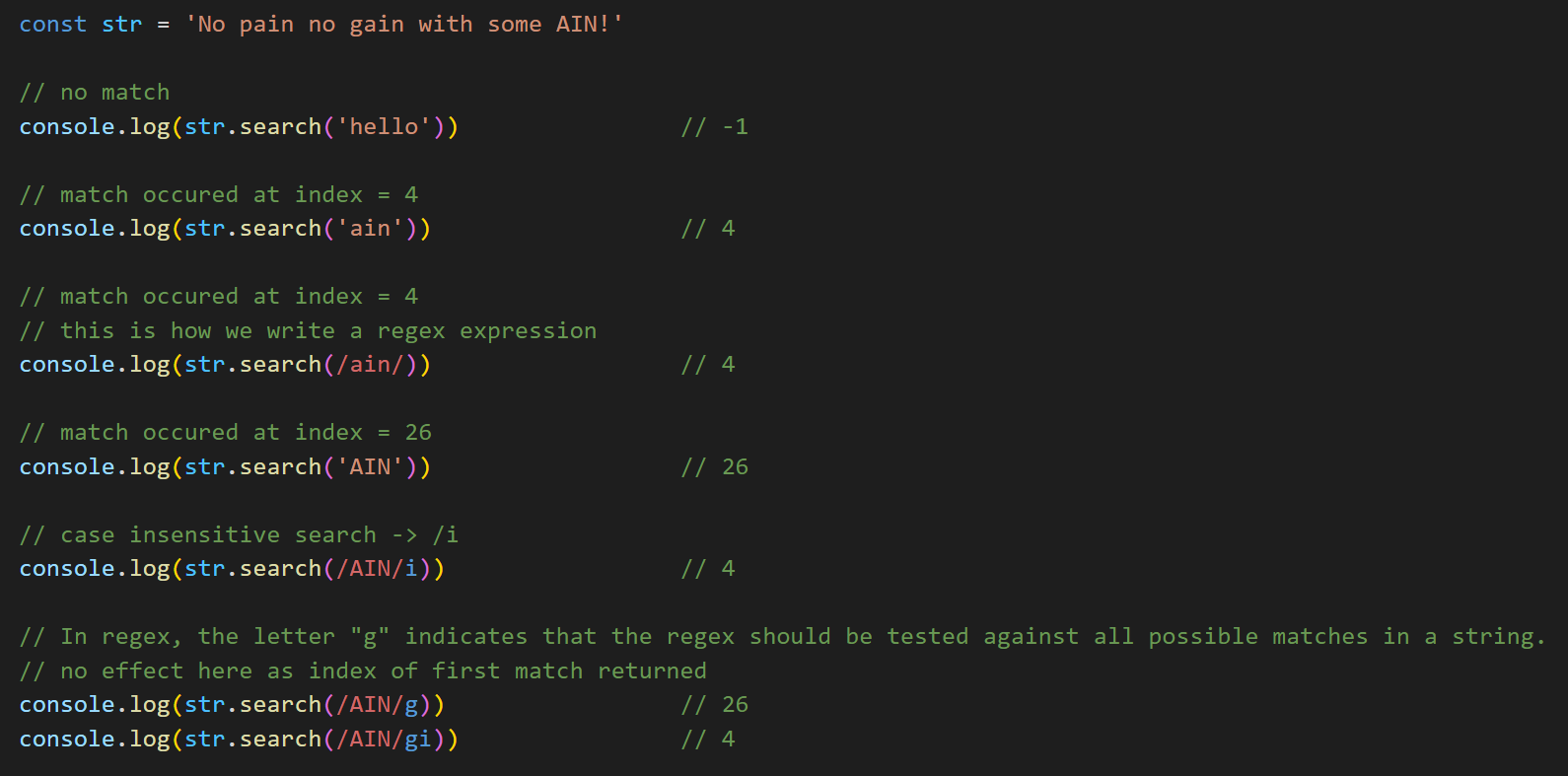
* check code for details

**String Method – localeCompare**

* mainString.localeCompare(compareString,locales:*optional*)
* locales: different languages have different set of alphabets. By default, its 'en'
* localeCompare checks the order in which characters appear in the language
* for example - 'a' letter comes before 'b' letter in English alphabet
* if mainstring comes before comparestring, a negative value is returned (different for different browsers)
* if mainstring comes after comparestring, a positive value is returned (different for different browsers)
* if mainstring and comparestring have same order, 0 is returned  
    
  **A screen shot of a computer program

  Description automatically generated**

**String Method – search**

* *str.search(regex or string)*
* searches for a string or a regex pattern and returns index of “first” match
* returns index of only “first” match
* returns -1 if no such string or regex pattern
* by default search is case-sensitive  
  

**String Method – match**

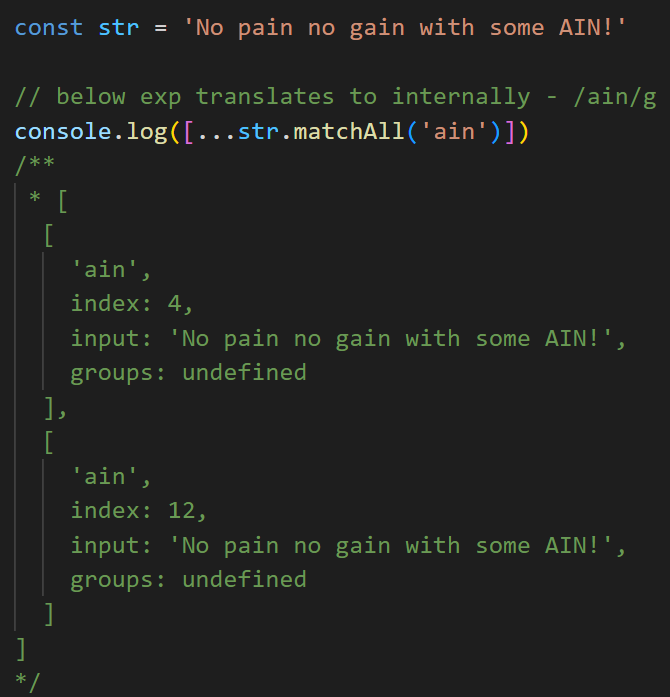
* str.match(regex or string)
* like str.search,provides index and other details of “first” match , however, can also be used to find out all possible matches (with the help of /g regex)
* string match: returns array containing details of first match
* regex match:  
   Case 1: if /g not used: returns array containing details of “first” match   
   Case 2: if /g used: returns array of “all” matches (no details only matched string)
* returns null if not a match  
    
  A screen shot of a computer code

  Description automatically generated  
  A computer screen shot of a black screen

  Description automatically generated  
  A black screen with text

  Description automatically generated

**String Method – matchAll**

* str.matchAll(regex or string)
* used to find all possible matches in a string
* returns: Object [RegExp String Iterator] {}, to extract data we spread into an array
* string match: returns “all” possible matches
* regex match:  
   Case 1: if /g not used: throws error (matchAll expects us to use /g)   
   Case 2: if /g used: returns “all” possible matches
* returns null if not a match  
  ****  
  **A screen shot of a computer program

  Description automatically generated**  
  **A screenshot of a computer program

  Description automatically generated**  
  **A screenshot of a computer error

  Description automatically generated**

**String Method – replace**

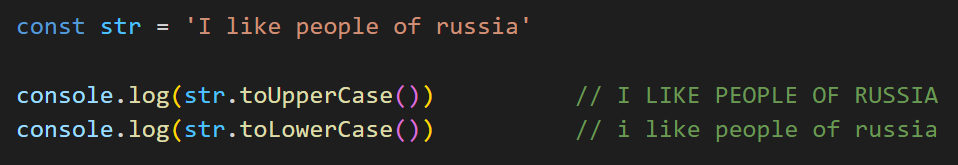
* *str.replace(searchString or regex, replaceString)*
* *str.replace(searchString or regex, callbackFunction)*
* like String.match only difference is that it replaces the matched string with replaceString or callbackFunction return value. (first or all depends)
* the process of search based on string/regex is same as String.match
* returns the new replaced string   
    
  *replaceString vs callbackFunction:*
* if matched searchString needs to be replaced by a constant value, use replaceString
* if some dynamic operation needs to be performed on matched searchString, like upper-casing them, use callbackFunction  
  ******

**String Method – replaceAll**

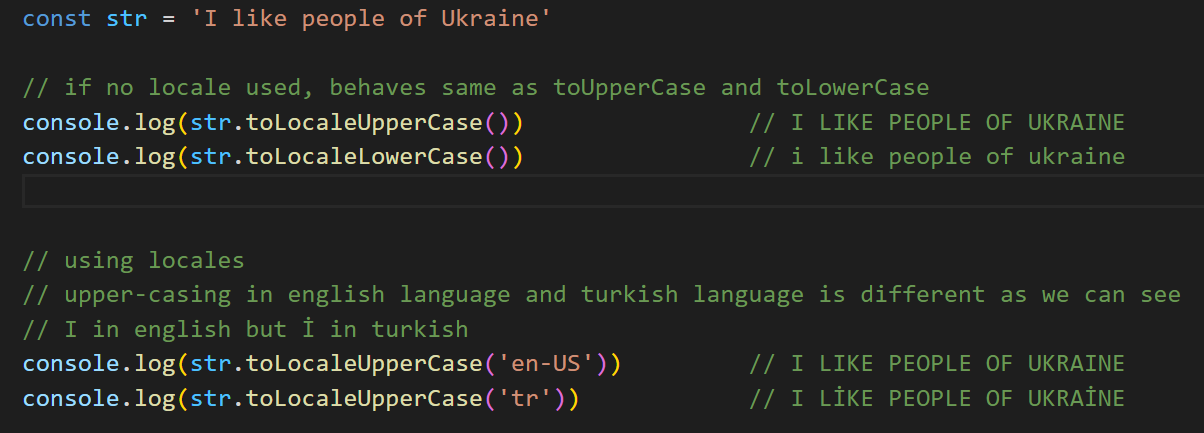
* *String.replaceAll(searchString or regex, replaceString)*
* *String.replaceAll(searchString or regex, callbackFunction)*
* like String.matchAll only difference is that it replaces “all” the matched string with replaceString or callbackFunction return value.
* search process same as String.matchAll
* to choose replaceString or callbackFunction, check String.replace
* returns replaced string  
  **A screen shot of a computer program

  Description automatically generated**

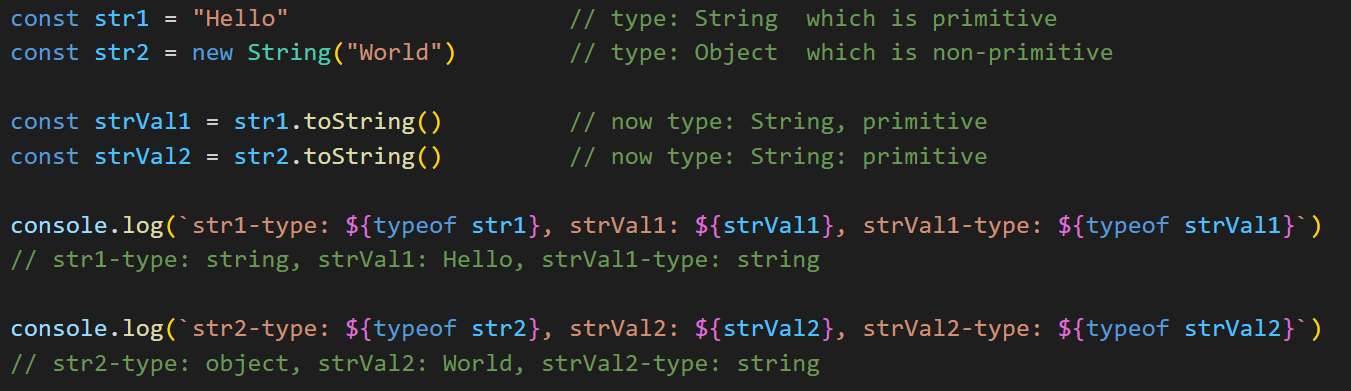
**String casing method(s): toUpperCase | toLowerCase**

* *str.toUpperCase()*
* *str.toLowerCase()*
* converts whole string into lower or upper cases
* returns a cased string  
  ****

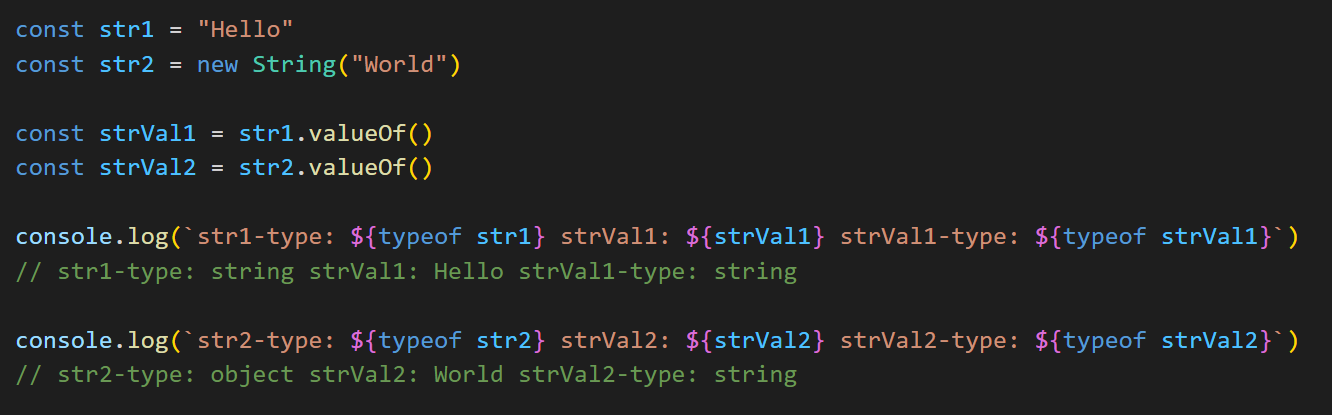
**String locale casing method(s): toLocaleUpperCase | toLocaleLowerCase**

* *str.toLocaleUpperCase()*
* *str.toLocaleUpperCase(locales)*
* *str.toLocaleLowerCase()*
* *str.toLocaleLowerCase(locales)*
* converts whole string into lower or upper cases (w.r.t locales if used)
* *locales*:  
   - optional  
   - locales mean which country's language to be used to perform case operation
* returns a cased string  
  

**String Method – toString**

* str.toString()
* returns a string data type which is primitive  
   - non-primitive String object (See str2) is also converted to primitive string  
   - value remains same only data type converted to primitive data type  
  ****

**String Method – valueOf**

* str.valueOf()
* on paper looks like str.toString but leave for future discussion  
  ****