String Methods:

Sl no:	string Method	Arguments	used to	returns
1.	toLowerCase	()	convert the string to lowercase	lowercase string
2.	toUpperCase	()	convert the string to Uppercase	Uppercase string
3.	trimStart	()	remove the spaces at starting of string	string without spaces at starting
4.	trimEnd	()	remove the spaces at ending of string	string without spaces at ending
5.	trim	()	remove the spaces in both starting and ending	removed spaces from both start & end.
6.	concat	(arg1 n-number ....)	join two or more strings	Joined string
7.	replace	(arg1, arg2) old string, new string to be added	replace old string with new string (only replaces first occurrence by default)	replaced string
8.	replaceAll	(arg1, arg2)	replace old string with new string for all occurences	replaced string

9.	slice	(arg1, arg2) ↑ start index (included) ↑ end index (excluded)	Cut Part of the string	Part of the string
10.	substring	(arg1, arg2) (same as slice)	Cut Part of the string but	Part of the string
11.	PadStart	(arg1, arg2) ↑ length of the string to be joined ↑ new string to be joined	Join a new string at start of the string	return the string as per the length joined
12.	PadEnd	(arg1, arg2) Same as PadStart	Join a new string at end of the string	(same as Pad Start)
13.	charAt	(arg1) ↑ index Position	get the character based on index Positions	returns character available in the given index
14.	at	(arg1) Same as charAt	get the character based on ~~fail~~ index Positions for both positives negative	Same as charAt
15.	charCodeAt	(arg1) ↑ index Position	get the ASCII valued based on Index Positions. Index not available then Nan	* return ASCII valued. * no index means Nan
16.	String.fromCharCode	(arg1) ↑ ASCII Valued	get characters of ASCII Valued	returns character of the given ASCII valued
17.	split (convert to string from array)	(arg1, arg2) ↑ string to split ↑ limit of split	Split the string based on given string	returns array of splited strings.

18.	startswith	True (or) false	(arg 1) ↑ string to given	check whether the given string is starting or not	if it is there then return true else return false
19.	Endswith	True (or) False	(arg 1) Same as startswith	Same as but in endswith string	Same as startswith
20.	includes	True (or) False	(arg 1, arg 2) ↑ string to be searched ↑ index pos from where to search	check whether the given string is available or not inside actual string	Same as above
21.	indexof		(arg 1, arg 2) Same as above	Same as above	return index position of string if the string is available else -1 always return first occured string
22.	lastindexof		(arg 1, arg 2)	Same as above	Same as above but in backward direction
23.	search		(arg 1) ↑ regular exp (or) string	Same as above index of	Same as above search the actual string
24.	match		(arg 1)		
25.	matchAll		(arg 1)		

search in forward direction

Search in backward direction

Sl. no.	Array Methods	Arguments	used to	returns	(overwriting) affects the original array or not
1.	Includes T/F	(arg1, arg2) ↑ array to be searched ↑ index Position from where to search	used to check whether the given array is available or not inside actual array	returns True (or) False	NO
2.	IndexOf	(arg1, arg2) same as above	same as above (Forward) direction	returns array index Position if available else returns -1	No
3.	lastIndexof	(arg1, arg2) Same as above	same as above (Backward) direction	Same as above	NO
4.	slice	(arg1, arg2) ↑ start index (included) ↑ end index (excluded)	used to cut part of the array	returns Part of the array	NO
5.	Concat	(arg1.....n) ↑ new array to be joined	used to join two or more arrays	returns a join array	NO
6.	unshift	(arg1.....n) ↑ new elements to be added	used to add elements at the starting of an array	returns updated array length	yes
7.	Push	(arg1.....n) ↑ same as above	used to add elements at the ending of an array	returns updated array length	yes

8.	shift	( )	used to remove the first element from the array	returns the removed element from the array	yes
9.	Pop	( )	used to remove the last element from the array	returns the removed element from the array	yes
10.	splice	(arg1, arg2, arg3....) arg1: index Position from where to add or delete arg2: number of elements to be deleted arg3: from third argument all consider as new elements to be added	used to add or delete the elements from the array based on index Position.	returns an array of removed elements	yes
11.)	Fill	(arg1, arg2, arg3) arg1: value to be filled arg2: start index Position from where to fill (included) arg3: end index Position from where to fill (excluded)	used to fill all the array elements with a new value	returns an array of newly filled elements	yes
12.)	reverse	( )	used to reverse the array elements	returns an array of reversed elements	yes
13.)	Join	(arg1) type string (separator) " if dont Pass any argument it seperate , comma by default)	used to seperate array elements with Particular seperator	returns a string of seperated array elements	no

14.)	at	(arg1) ↑ index Position	used to get the array elements based on index position	return element present in the given index	no
15.)	forEach	(arg1) ↑ callback function (arg1, arg2, arg3) ↑ array elements ↑ index position ↑ whole array	used to iterate array elements → call back function runs the total number of times the elements available inside the array (0 to till length-1)	always return undefined Note: = foreach should be used only to iterate array elements that means we should not use these method when we are returning any value	no
16.)	map	(arg1) same as above	used to perform some changes on array elements	returns an array of updated elements	no
17.)	filter	(arg1) same as above	used to filter out the array elements based on condition Note: it stores the elements inside a new array if the condition is satisfied else it doesn't store the elements	return an array of filtered elements Note: If the condition is not satisfied with any elements then it returns an empty array	no
18.)	some (T, F)	(arg1) Same as above	used to check atleast one element is satisfying the given condition or not	it returns true if any one condition is satisfied else it returns false	no

19.)	Every (T, F) Some: Some: it's going to terminate the loop once condition is satisfied	(arg 1) same as above	used to check whether the given condition is satisfies or not.	it returns true if all the condition satisfies else it returns false (it terminates the not satisfied condition)	no
20.)	Flat	(arg 1) ↑ Depth of flat (type number) (if we don't pass any argument it's going to flatten one depth)	used to flatten the array (it joins array elements) to the existing array	returns the flattened array	no
21.)	flatMap				
22.)	find	(arg 1) ↑ Callback function (arg 1, arg 2, arg 3) ↑ ↑ ↑ array elements index positions whole array	used to get the element which is satisfying the condition	when condition satisfied it returns element else undefined (when condition satisfies it's going to terminate the loop)	no
23.)	findIndex	same as above	used to get the index of the element which is satisfying the condition	when condition satisfied it returns elements index position else -1.	no

24.)	sort	(arg1) (Call back Function) (arg1, arg2) ↓ ↓ array elements	used to sort the array elements based on ASCII values (By Default). Note: ## we are sorting a string then we should convert the array elements to either uppercase or lowercase and sort and it doesn't need any argument	yes
25.)	reduce	(arg1, arg2) ↓ callback function ↓ initial value ↓ (arg1, arg2, arg3, arg4) ↓ current value (or) initial value (or) total value ↓ element of the array available at ith index ↓ index position of element available at ith index ↓ whole array (if the reduce() 2nd agr is not passed)	used to reduce all the array elements to a single value. ↓ if the 2nd argument of reduced method passed then, the call back function 1st argument will be having reduce() and the call back function 2nd argument will be having array element at index position 0).	
26.)	reduce right			

iterative loops:

new identifier

Array we are iterating

1.) for of ⇒ for (let ele of Products) {statement}

2.) for in ⇒ for (let index of Products) {statement}

new identifier which the index stored

1.) for of:

for of loop is used to iterate elements of iterable type of objects

2.) for in:

for in loop is used to iterate index positions.

Array.isArray()	if the given argument is an array then it returns true else it returns false.	It is used to check whether the given argument is array or not
Array.from()	It accepts one arg where the argument is iterable types of objects	It is used to convert iterable type of objects to a pure array.
set object	same as above Array.from()	set is an object in Javascript which is used to remove duplicate elements from the array.
new		

JSON:

* JSON stands for JavaScript object notation which is used in creation of APIs.
* API stands for Application Program interface which interacts between client and the server.
* API will be in the form of JSON data.
* In JSON object key must be a double quoted string.
* In Javascript object must be stored in .js file. and the JSON object mut be stored in .JSON files.

JSON.stringify():

* JSON.stringify() converts Javascript object to a string of JSON objects. (whole object is a single string).

JSON.Parse():

* JSON.Parse method converts JSON objects to a Javascript objects.
* Spread operator removes reference of only one level. If we want to remove reference of every nested level we use JSON.Parse(). like this

eg:
```
let student = JSON.Parse(JSON.stringify(employee))
clg(student)
```

Spread operator: (only can we in start and middle)

* Spread operator is used to spread the array elements as individual elements and object Properties as individual Properties.

Rest Parameter:- (only can use at the end)

* Rest Parameter is used to store the rest of the parameters in the form of array in functions and rest of the array elements in the form of array in array destructuring and rest of the object properties in the form of objects in object destructuring.

Destructuring:

* Destructuring is the process of extracting array elements and object properties and storing them inside individual variable.

Array Destructuring	object Destructuring
It avoids using Array names again and again when we are accessing the array elements.	It avoids using object names again and again when we are accesing the object properties.
when we do destructuring the elements will be stored inside the variable in sequential order.	when destructuring the variable name must be same as the object key name.
when we don't don't want to store a array element (or) skip the array element just Pass , (comma)	when we don't want to store or skip the object Properties just ignore the key name at the time of destructuring.

Object Methods:

Sl.no.	object methods	Process
1.	object.keys(objectname)	Gives the object keys
2.	object.values(objectname)	Gives the object values
3.	object.entries	Gives both key and value in an array → key in double Quotation names
4.	object.keys(objectname).length	Gives length of the object ⇒6
5	object.hasOwnProperty("detail")	checks whether its available or not if available it returns true else false.
6.	object.freeze(objectname)	freezes the object we cannot add, update, delete
7.	object.frooze h(objectname)	checks whether freeze or not (returns true or false)
8.	object.seal(objectname)	same as freeze but we can update
9.	object.sealed	checks whether seal or not (returns true or false.

Dom methods to get the html elements:

~~1.) document.~~

Sl.No.	Dom Methods()	used to d agg	returns	if element is not available
1.	document.getElementsById()	used to get the elements ById. It accepts one argument of type string where the argument is elements id.	it id is available for any element it returns first occured elements.	it id is not available for any element it returns null
2.	document.querySelector()	replace with given method of Dom	same as above	same as above
3.	(Node list) document.querySelectorAll()	selects all the occurences replace with given method of Dom	if the selector is available it returns all the elements in the form of node list	if the selector is not available for any of the elements then it returns empty node list
4	(~~Node~~ Html collection) document.getElementsByClassName()	//	it returns all the elements available with the classname in the form of ~~Html collection~~ Html collection	if the class name is not available for any element then it returns empty html collection
5.	(Html collection) document.getElementsByTagName()	// Same as above	// Same as above	Same as above