# Array methods

**From, isArray, of, concat, fill, filter, find, findIndex, forEach, includes, indexOf, join, keys, lastIndexOf, map, pop, push, reduce, reduceRight, reverse, shift, slice, some, sort, splice, toString, unshift, values**

1. **Array.from(): New. C**reates new instance from an array-like or iterable object.

Array.from('foo')

// [ "f", "o", "o"]

1. **Array.isArray()**
2. **Array.of(): New. C**reates new instance from variable number of arguments, regardless of number or type of the arguments.

Array.of(1, 2, 3); // [1, 2, 3]

1. **concat(): New. M**erge two or more arrays.

const letters = ['a', 'b', 'c'];

const numbers = [1, 2, 3];

letters.concat(numbers);

// result in ['a', 'b', 'c', 1, 2, 3]

4.5. every(callback)

1. **fill(value, start, end):** Modified. Changes all elements in an array to a static value, from a start index (default 0) to an end index (default array.length). Returns modified array.
2. **filter(callback):** New. arr.filter(callback(element[, index, [array]])[, thisArg])
3. **find(callback)**: Returns the **value** of the **first element** that satisfies the provided function. arr.find(callback(element[, index[, array]])[, thisArg])
4. **findIndex(callback)**:Returns **index** of first element in the array **that satisfies the provided testing function**. Otherwise, it returns -1
5. **forEach(callback): arr.forEach(callback(currentValue [, index [, array]])[, thisArg])**
6. **includes()**: Determines whether an array includes a certain value. Returns true or false. arr.includes(valueToFind[, fromIndex])
7. **indexOf():** First index at which a given element can be found in the array. -1 if not found. arr.indexOf(searchElement[, fromIndex])
8. **join(): Joins array. Returns string. arr.join([separator])**

**join(‘’) converts array to string**

1. **keys(): arr.keys()**
2. **lastIndexOf()**
3. **map(): new**
4. **pop(): returns last. Modifies array**
5. **push(): Modifies array. Returns new length of array**
6. **reduce(): Returns single value.** arr.reduce((prevValue, current, index, arr)=>{}, initialValue)
7. **reduceRight(): Works same but starts from right to left**
8. **reverse(): Modifies. Reverses in place. Returns reversed array.**
9. **shift(): Modifies. Removes first element from array and returns first element. Opposite of pop()**
10. **slice()**: New. Returns a shallow copy of a portion of an array into a new array object selected from start to end (end not included). The original array not modified. **arr.slice([start[, end]]).** A negative index can be used, indicating an offset from the end of the sequence. slice(-2) extracts the last two elements in the sequence.If start is undefined, slice starts from the index 0.
11. **some(callback)**: Tests whether at least one element in the array passes the test by provided function. Returns Boolean.
12. **sort([comparefunction])**: Modifies. Sorts the elements of an array in place and returns the sorted array. **arr.sort([compareFunction])**

let numbers = [4, 2, 5, 1, 3];

numbers.sort((a, b) => a - b);

console.log(numbers);

// [1, 2, 3, 4, 5]

1. **splice()**: Modifies. Changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.

**let arrDeletedItems = array.splice(start[, deleteCount[, item1[, item2[, ...]]]])**

**If you want to add, then deleteCount must be given zero**

const arr = [5,1,2,3,4,5]

arr.**splice**(2,2,'a','b','c') *// [5,1,'a','b','c', 4, 5]*

1. **toString():** For Array objects, the toString method joins the array and returns one string containing each array element separated by commas.
2. **unshift(): Modifies. A**dds one or more elements to the beginning of an array and returns the new length of the array. **arr.unshift(element1[, ...[, elementN]])**
3. **values():** returns a new **Array Iterator** object that contains the values for each index in the array. arr.values()

# String methods

**charAt, concat, endswith, includes, indexOf, lastIndexOf, match, padEnd, padStart, repeat, replace, replaceAll, slice, split, startsWith, substring, toLowerCase, toUpperCase**

1. **length**
2. **charAt(n):** Character at location
3. **concat():** tr.concat(str2 [, ...strN])
4. **endsWith(string): R**eturns true / false
5. **includes(string):** true / false
6. **indexOf(string):** str.indexOf(searchValue [, fromIndex])
7. **lastIndexOf():**str.lastIndexOf(searchValue[, fromIndex])
8. **match():** matches against regex. str.match(regexp)
9. **padEnd():** str.padEnd(targetLength [, padString])
10. **padStart():** str.padStart(targetLength [, padString])
11. **repeat()**: Returns a new string which contains the specified number of copies of the string, concatenated together. str.repeat(count). 'abc'.repeat(2) // 'abcabc'
12. **replace(): New.** const newStr = str.replace(regexp|substr, newSubstr|function). Replaces only first occurrence.
13. **replaceAll(): New.**

const newStr = str.replaceAll(regexp|substr, newSubstr|function)

1. **slice(): New. E**xtracts a section of a string and returns it as a new string, without modifying the original string

str.slice(beginIndex[, endIndex])

1. **split():** str.split([separator[, limit]])
2. **startsWith():** str.startsWith(searchString[, position])
3. **substring(): new.** str.substring(indexStart[, indexEnd]). substring() extracts characters from indexStart up to but not including indexEnd
4. sub
5. **toLowerCase(): str.toLowerCase()**
6. **toUpperCase(), trim(), trimStart(), trimEnd()**

# Objects

Object.keys(), Object.values(), Object.assign()

# Math

1. Math.ceil(0.5); //=> 1 (up)

Math.floor(0.5); //=> 0 (down)

Math.round(0.5); //=> 1 (nearest)

Math.round(0.49); //=> 0

Number.parseFloat() only accepts a single argument, the string that should be parsed into a floating-point number:

Number.parseFloat('3.14159');

//=> 3.14159

1. console.log(12 % 5); // expected output: 2
2. Number.isInteger(42) //=> true

Number.isInteger(0.42) //=> false

1. Number.isNaN(10) *//=> false*
2. Number.isNaN(**undefined**) *//=> false*
3. Number.isNaN(**NaN**) *//=> true*

#### Number.parseInt()

Accepts a string as its first argument and parses it as an integer. The second argument is the base that should be used in parsing (e.g., 2 for binary or 10 for decimal). For example, 100 is 100 in decimal but 4 in binary

Number.parseInt('100', 10); *//=> 100*

Number.parseInt('100', 2); *//=> 4*

Number.parseFloat('3.14159'); *//=> 3.14159*

1. Quotient is Math.floor(a/b), remainder is (a % b)

# Common tips and tricks

1. **Reverse a string**

return **Array**.**from**(S).**reverse**().**join**()

1. **Get max from array**

Math.**max**(...A)

1. Divisor and remainder

Quotient = Math.floor(a/b)

Remainder = a % b

1. Convert to binary

const n = 1234

let binary = n.**toString**(2)

1. Unique value array

const set = new **Set**([1, 2, 2, 3, 4, 5, 5])

const arr3 = **Array**.**of**(...set)

or

const arrayWithUniqueItems = [...new Set([1, 2, 3, 3,])]

1. Convert array to object with indices as keys of object

Const obj = {…arr}

1. Swap key / values of object

const swapObj = {}

**Object**.**entries**(obj).**forEach**(([key, value])=>{

        swapObj[value] = key

    })

# Regexp

* The search() method returns the position of match

const str = ‘visit GeeksforGeeks’

Str.search(/geeksforgeeks/i); case insensitive search result: 6

The match() returns array of matches found

The test() returns true / false

* [...] checks if any one character between the brackets is present in the searched string
* [^...] checks if any one character between the brackets is not present in the searched string
* Samples [0-9], [a-z],[A-Z],[a-zA-Z]
* p+ matches any string containing one or more p's. p\* matches any string containing zero or more p's. p? matches any string containing zero or more p's. p{N} matches any string containing a sequence of N p's
* p$ matches any string containing p at the end of it. ^p matches any string containing p at the start of it
* [^a-zA-Z] matches any string not containing any of the characters ranging from a through z and A through Z
* \0 is null, \t is tab, \n is new line, \r is carriage return
* Meta chars: \s matches a whitespace character (space, tab, newline), \S matches a non-whitespace character, \d matches a digit (0-9), \D matches a non-digit, \w matches a word character (a-z, A-Z, 0-9, \_), \W matches a non-word character.

[aeiou] matches a single character in the given set

[^aeiou] matches a single character outside the given set

(foo|bar|baz) matches any of the alternatives specified. /(foo|bar|baz)/g