**Масив**

поредица от елементи подредени един до друг. Правим референция към първият елемент и от там и следващите. Масива знае само първата си стойност и всичко останало сочи за отместване спрямо първият елемент, затова и индексите започват от 0 до length – 1. Масивите не се дефинират с точна дължина. Инициализира се с [ ]. Масив от масива е матрица.

# indexOf

# <https://www.w3schools.com/jsref/jsref_indexof.asp>

The indexOf() method returns the first index (position) of a specified value.

The indexOf() method returns -1 if the value is not found.

The indexOf() method starts at a specified index and searches from left to right.

## **Syntax**

*string*.indexOf(searchvalue, start)

.**push()**

[**https://www.w3schools.com/jsref/jsref\_push.asp**](https://www.w3schools.com/jsref/jsref_push.asp)

- поставя елемента от скобите на последна позиция в един масив

join() - съединява всички елементи в един масив ползвайки конкретен сепаратор, който сме избрали (“ - “);

## **Syntax**

*array*.push(item1, item2, ..., itemX)

**for of**

[**https://www.w3schools.com/js/js\_loop\_forof.asp**](https://www.w3schools.com/js/js_loop_forof.asp)

– алтернатива на for цикъла

- не му казваме до кога да итерира, той си знае кога да спре. Не следи индексите на елементите.

### **Syntax**

for (let el of arr) {  
  // *code block to be executed*  
}

**.shift()**

[**https://www.w3schools.com/jsref/jsref\_shift.asp**](https://www.w3schools.com/jsref/jsref_shift.asp)

- взима първият елемент и го маха/ изрязва

ако го запишем към даден масив arr.shift() - само го взимаме без да го махаме.

## **Syntax**

*array*.shift()

**.repeat(count)** <https://www.w3schools.com/jsref/jsref_repeat.asp>

– повтаря N на брой пъти определен код

The Continue statement "jumps over" one iteration in the loop.

The Continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

## **Syntax**

*string*.repeat(count)

.pop <https://www.w3schools.com/jsref/jsref_pop.asp>

The pop() method removes (pops) the last element of an array.

The pop() method changes the original array.

The pop() method returns the removed element.

**Syntax**

*array*.pop()

.unshift <https://www.w3schools.com/jsref/jsref_unshift.asp>

The unshift() method adds new elements to the beginning of an array.

The unshift() method overwrites the original array.

## **Syntax**

*array*.unshift(item1, item2, ..., itemX)

.filter <https://www.w3schools.com/jsref/jsref_filter.asp>

The filter() method creates a new array filled with elements that pass a test provided by a function.

The filter() method does not execute the function for empty elements.

The filter() method does not change the original array.

## **Syntax**

*array*.filter(function(currentValue, index, arr), thisValue)

.slice <https://www.w3schools.com/jsref/jsref_slice_array.asp>

The slice() method returns selected elements in an array, as a new array.

The slice() method selects from a given start, up to a (not inclusive) given end.

The slice() method does not change the original array.

## **Syntax**

*array*.slice(start, end)

.splice <https://www.w3schools.com/jsref/jsref_splice.asp>

The splice() method adds and/or removes array elements.

The splice() method overwrites the original array.

## **Syntax**

*array*.splice(index, howmany, item1, ....., itemX)

.map <https://www.w3schools.com/jsref/jsref_map.asp>

map() creates a new array from calling a function for every array element.

map() calls a function once for each element in an array.

map() does not execute the function for empty elements.

map() does not change the original array.

## **Syntax**

*array*.map(function(currentValue, index, arr), thisValue)

.reverse <https://www.w3schools.com/jsref/jsref_reverse.asp>

The reverse() method reverses the order of the elements in an array.

The reverse() method overwrites the original array.

**Syntax**

*array*.reverse()

.sort <https://www.w3schools.com/js/js_array_sort.asp>

The sort() sorts the elements of an array.

The sort() overwrites the original array.

The sort() sorts the elements as strings in alphabetical and ascending order.

## [Syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#syntax)

// Functionless

sort()

// Arrow function

sort((a, b) => { /\* ... \*/ } )

// Compare function

sort(compareFn)

// Inline compare function

sort(function compareFn(a, b) { /\* ... \*/ })

.localeCompare <https://www.w3schools.com/jsref/jsref_localecompare.asp>

The localeCompare() method compares two strings in the current locale.

The localeCompare() method returns sort order -1, 1, or 0 (for before, after, or equal).

## **Syntax**

*string*.localeCompare(compareString).

.split()

<https://www.w3schools.com/jsref/jsref_split.asp>

The split() method splits a string into an array of substrings.

The split() method returns the new array.

The split() method does not change the original string.

If (" ") is used as separator, the string is split between words.

## **Syntax**

*string*.split(separator, limit).