JavaScript Array Methods - Complete Reference

■ Common Array Methods

Method	Definition	Syntax	Example
push()	Adds elements at end	arr.push(10)	let arr=[1,2]; arr.push(3); → [1,2,3]
pop()	Removes last element	arr.pop()	let arr=[1,2,3]; arr.pop(); → [1,2]
shift()	Removes first element	arr.shift()	let arr=[1,2,3]; arr.shift(); → [2,3]
unshift()	Adds at beginning	arr.unshift(0)	let arr=[1,2]; arr.unshift(0); → [0,1,2]
concat()	Merges arrays	arr1.concat(arr2)	let a=[1],b=[2]; a.concat(b); → [1,2]
join()	Joins elements as string	arr.join('-')	let arr=[1,2,3]; arr.join('-'); → '1-2-3'
slice()	Extracts section	arr.slice(1,3)	let arr=[1,2,3,4]; arr.slice(1,3); → [2,3]
splice()	Add/remove elements	arr.splice(1,1,'x')	let arr=[1,2,3]; arr.splice(1,1,'x'); → [1,'x',3]
toString()	Converts to string	arr.toString()	let arr=[1,2]; arr.toString(); → '1,2'

Important Points:

■ `push`, `pop` modify the end; `shift`, `unshift` modify the beginning.
■ `splice` changes the original array, `slice` does not.
■ `join` and `toString` return strings.

■ Searching & Finding Methods

Method	Definition	Syntax	Example
indexOf()	First index of element	arr.indexOf(2)	let arr=[1,2,3]; arr.indexOf(2); → 1
lastIndexOf()	Last index of element	arr.lastIndexOf(2)	let arr=[1,2,2]; arr.lastIndexOf(2); → 2
includes()	Checks presence	arr.includes(2)	let arr=[1,2,3]; arr.includes(2); → true
find()	First match	arr.find(x=>x>2)	let arr=[1,2,3,4]; arr.find(x=>x>2); → 3
findIndex()	Index of first match	arr.findIndex(x=>x>2)	let arr=[1,2,3,4]; arr.findIndex(x=>x>2); → 2
at()	Access element by index	arr.at(-1)	let arr=[1,2,3]; arr.at(-1); → 3

Important Points:

■ `indexOf`/`lastIndexOf` return -1 if not found.
■ `includes` is easier for true/false checks.
■ `at()` supports negative indexing.

■ Higher Order Methods

Method	Definition	Syntax	Example
forEach()	Executes for each element	arr.forEach(x=>...)	let arr=[1,2]; arr.forEach(x=>console.log(x))
map()	Creates new transformed array	arr.map(x=>x*2)	let arr=[1,2]; arr.map(x=>x*2); → [2,4]
filter()	Filters elements by condition	arr.filter(x=>x>2)	let arr=[1,2,3]; arr.filter(x=>x>2); → [3]
reduce()	Reduce to single value	arr.reduce((a,b)=>a+b,0)	let arr=[1,2,3]; reduce → 6
reduceRight()	Reduce right to left	arr.reduceRight((a,b)=>a-b)	let arr=[1,2,3]; → 0
some()	At least one passes test	arr.some(x=>x>2)	let arr=[1,2,3]; → true

every()	All pass test	arr.every(x=>x>0)	let arr=[1,2,3]; → true

Important Points:

- `map`, `filter` return new arrays, `forEach` returns undefined.
- `reduce` is powerful for sums, averages, etc.
- `some` and `every` return booleans.

■ Utility & Other Methods

Method	Definition	Syntax	Example
sort()	Sorts array	arr.sort()	let arr=[3,1,2]; arr.sort(); → [1,2,3]
reverse()	Reverses array	arr.reverse()	let arr=[1,2]; arr.reverse(); → [2,1]
fill()	Fill with static value	arr.fill(0)	let arr=[1,2,3]; arr.fill(0); → [0,0,0]
flat()	Flatten nested arrays	arr.flat(2)	let arr=[1,[2,[3]]]; arr.flat(2); → [1,2,3]
flatMap()	Map & flatten	arr.flatMap(x=>[x*2])	let arr=[1,2]; → [2,4]
copyWithin()	Copy part within array	arr.copyWithin(1,0)	let arr=[1,2,3]; arr.copyWithin(1,0); → [1,1,2]
entries()	Iterator of [index,value]	arr.entries()	for (let [i,v] of [10,20].entries())...
keys()	Iterator of indices	arr.keys()	for (let k of [10,20].keys())...
values()	Iterator of values	arr.values()	for (let v of [10,20].values())...

Important Points:

- `sort` sorts as strings unless compareFn is used.
- `flat` is useful for nested arrays.
- `copyWithin` modifies the array in place.
- `entries`, `keys`, `values` are used with loops.

■ Static Array Methods

Method	Definition	Syntax	Example
Array.isArray()	Check if value is array	Array.isArray([])	→ true
Array.from()	Create array from iterable	Array.from('abc')	→ ['a','b','c']
Array.of()	Create array from args	Array.of(7)	→ [7]

Important Points:

- `Array.isArray` is best to check arrays.
- `Array.from` converts sets, strings, arguments into arrays.
- `Array.of` avoids confusion with Array constructor.