

# JavaScript

---

# Array methods

---

JavaScript Array methods with examples

# JavaScript Array Methods

## at() Method

The `at()` method takes an integer value and returns the item at that index, allowing for positive and negative integers. Negative integers count back from the last item in the array.



```
1 const array1 = [5, 12, 8, 130, 44];
2 let index = 2;
3 console.log(`Using an index of ${index} the item returned is ${array1.at(index)}`);
4 // Expected output: "Using an index of 2 the item returned is 8"
5 index = -2;
6 console.log(`Using an index of ${index} item returned is ${array1.at(index)}`);
7 // Expected output: "Using an index of -2 item returned is 130"
```

# JavaScript Array Methods

## concat() Method

The concat() method merges two or more arrays into a new array without modifying the original arrays.



```
1 const arr1 = [1, 2];
2 const arr2 = [3, 4];
3 const merged = arr1.concat(arr2);
4 console.log(merged); // Output: [1, 2, 3, 4]
```

# JavaScript Array Methods

## copyWithin() Method

The `copyWithin()` method copies a portion of an array to another location within the same array.



```
1 const arr = [1, 2, 3, 4];
2 arr.copyWithin(0, 2);
3 console.log(arr); // Output: [3, 4, 3, 4]
```

# JavaScript Array Methods

## entries() method

The `entries()` method returns an iterator object for the key-value pairs (index and value) in an array.



```
1 const arr = ['a', 'b'];
2 const iterator = arr.entries();
3 console.log(iterator.next().value); // Output: [0, "a"]
```

# JavaScript Array Methods

## every() Method

The `every()` method tests if all elements in the array pass a provided function's test.



```
1 const arr = [1, 2, 3];
2 const allGreaterThanZero = arr.every(x => x > 0);
3 console.log(allGreaterThanZero); // Output: true
```

# JavaScript Array Methods

## fill() method

The fill() method changes all elements in an array to a specified value.



```
1 const arr = [1, 2, 3];
2 arr.fill(0);
3 console.log(arr); // Output: [0, 0, 0]
```

# JavaScript Array Methods

## filter() method

The filter() method creates a new array with all elements that pass a provided function's test.



```
1 const arr = [1, 2, 3];
2 const filtered = arr.filter(x => x > 1);
3 console.log(filtered); // Output: [2, 3]
```

# JavaScript Array Methods

## find() method

The `find()` method returns the first element in an array that satisfies a provided testing function.



```
1 const arr = [1, 2, 3];
2 const found = arr.find(x => x > 1);
3 console.log(found); // Output: 2
```

# JavaScript Array Methods

## findIndex() method

The `findIndex()` method returns the index of the first element in an array that satisfies a provided testing function.



```
1 const arr = [1, 2, 3];
2 const index = arr.findIndex(x => x > 1);
3 console.log(index); // Output: 1
```

# JavaScript Array Methods

## findLast() method

The `findLast()` method iterates the array in reverse order and returns the value of the first element that satisfies the provided testing function. If no elements satisfy the testing function, `undefined` is returned.



```
1 const array1 = [5, 12, 50, 130, 44];
2 const found = array1.findLast((element) => element > 45);
3 console.log(found);
4 // Expected output: 130
```

# JavaScript Array Methods

## findLastIndex() method

The `findLastIndex()` method iterates the array in reverse order and returns the index of the first element that satisfies the provided testing function. If no elements satisfy the testing function, `-1` is returned.



```
1 const array1 = [5, 12, 50, 130, 44];
2 const isLargeNumber = (element) => element > 45;
3 console.log(array1.findLastIndex(isLargeNumber));
4 // Expected output: 3
5 // Index of element with value: 130
```

# JavaScript Array Methods

## flat() method

The flat() method creates a new array with all sub-array elements concatenated recursively up to the specified depth.



```
1 const arr = [1, [2, [3]]];  
2 const flattened = arr.flat();  
3 console.log(flattened); // Output: [1, 2, [3]]
```

# JavaScript Array Methods

## flatMap() method

The flatMap() method first maps each element using a mapping function, then flattens the result into a new array.



```
1 const arr = [1, 2, 3];
2 const mapped = arr.flatMap(x => [x * 2]);
3 console.log(mapped); // Output: [2, 4, 6]
```

# JavaScript Array Methods

## forEach() method

The `forEach()` method executes a provided function once for each array element.



```
1 const arr = [1, 2, 3];
2 arr.forEach(x => console.log(x)); // Output: 1, 2, 3
```

# JavaScript Array Methods

## from() method

The `from()` method creates a new, shallow-copied array from an iterable object or array-like object.



```
1 const arr = Array.from('hello');  
2 console.log(arr); // Output: ['h', 'e', 'l', 'l', 'o']
```

# JavaScript Array Methods

## fromAsync() method

The `fromAsync()` static method creates a new, shallow-copied `Array` instance from an `async iterable`, `iterable`, or `array-like` object.



```
1 const asyncIterable = (async function* () {  
2   for (let i = 0; i < 5; i++) {  
3     await new Promise((resolve) => setTimeout(resolve, 10 * i));  
4     yield i;  
5   }  
6 })();  
7  
8 Array.fromAsync(asyncIterable).then((array) => console.log(array));  
9 // [0, 1, 2, 3, 4]
```

# JavaScript Array Methods

## includes() method

The includes() method checks if an array includes a certain element and returns true or false.



```
1 const arr = [1, 2, 3];
2 console.log(arr.includes(2)); // Output: true
```

# JavaScript Array Methods

## indexOf() method

The indexOf() method returns the first index at which a given element can be found in the array, or -1 if it's not present.



```
1 const arr = [1, 2, 3];
2 console.log(arr.indexOf(2)); // Output: 1
```

# JavaScript Array Methods

## isArray() method

The isArray() method checks if an object is an array.



```
1 console.log(Array.isArray([1, 2, 3])); // Output: true
```

# JavaScript Array Methods

## join() method

The join() method joins all elements of an array into a string, separated by a specified separator.



```
1 const arr = ['h', 'e', 'l', 'l', 'o'];
2 console.log(arr.join(''));
```

// Output: "hello"

# JavaScript Array Methods

## keys() method

The `keys()` method returns an iterator that generates keys (indices) for each element in the array.



```
1 const arr = ['a', 'b'];
2 const iterator = arr.keys();
3 console.log(iterator.next().value); // Output
```

# JavaScript Array Methods

## lastIndexOf() method

The `lastIndexOf()` method returns the last index at which a given element can be found in the array, or `-1` if it's not present.



```
1 const arr = [1, 2, 1];
2 console.log(arr.lastIndexOf(1)); // Output: 2
```

# JavaScript Array Methods

## map() method

The `map()` method creates a new array with the results of calling a provided function on every element in the array.



```
1 const arr = [1, 2, 3];
2 const mapped = arr.map(x => x * 2);
3 console.log(mapped); // Output: [2, 4, 6]
```

# JavaScript Array Methods

## of() method

The `of()` method creates a new array from the provided elements.



```
1 const arr = Array.of(1, 2, 3);
2 console.log(arr); // Output: [1, 2, 3]
```

# JavaScript Array Methods

## pop() method

The `pop()` method removes and returns the last element from an array.



```
1 const arr = [1, 2, 3];
2 const popped = arr.pop();
3 console.log(popped); // Output: 3
```

# JavaScript Array Methods

## push() method

The `push()` method adds one or more elements to the end of an array and returns the new length of the array.



```
1 const arr = [1, 2, 3];
2 const length = arr.push(4);
3 console.log(length); // Output: 4
4 console.log(arr); // Output: [1, 2, 3, 4]
```

# JavaScript Array Methods

## reduce() method

The reduce() method applies a function against an accumulator and each element in the array (from left to right) to reduce it to a single value.



```
1 const arr = [1, 2, 3];
2 const sum = arr.reduce((acc, val) => acc + val, 0);
3 console.log(sum); // Output: 6
```

# JavaScript Array Methods

## reduceRight() method

The reduceRight() method is similar to reduce(), but it applies the function from right to left.



```
1 const arr = [1, 2, 3];
2 const sum = arr.reduceRight((acc, val) => acc + val, 0);
3 console.log(sum); // Output: 6
```

# JavaScript Array Methods

## reverse() method

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



```
1 const arr = [1, 2, 3];
2 arr.reverse();
3 console.log(arr); // Output: [3, 2, 1]
```

# JavaScript Array Methods

## shift() method

The `shift()` method removes and returns the first element from an array.



```
1 const arr = [1, 2, 3];
2 const shifted = arr.shift();
3 console.log(shifted); // Output: 1
4 console.log(arr); // Output: [2, 3]
```

# JavaScript Array Methods

## slice() method

The slice() method extracts a section of an array and returns a new array.



```
1 const arr = [1, 2, 3];
2 const sliced = arr.slice(0, 2);
3 console.log(sliced); // Output: [1, 2]
```

# JavaScript Array Methods

## some() method

The `some()` method tests if at least one element in the array passes a provided function's test.



```
1 const arr = [1, 2, 3];
2 const someGreaterThanOrEqualToTwo = arr.some(x => x > 2);
3 console.log(someGreaterThanOrEqualToTwo); // Output: true
```

# JavaScript Array Methods

## sort() method

The `sort()` method sorts the elements of an array in place and returns the sorted array.



```
1 const arr = [3, 1, 2];
2 arr.sort();
3 console.log(arr); // Output: [1, 2, 3]
```

# JavaScript Array Methods

## splice() method

The `splice()` method changes the contents of an array by removing or replacing elements at a specified index.



```
1 const arr = [1, 2, 3];
2 arr.splice(1, 0, 4);
3 console.log(arr); // Output: [1, 4, 2, 3]
```

# JavaScript Array Methods

## toLocaleString() method

The `toLocaleString()` method returns a string with a language-sensitive representation of the array.



```
1 const arr = [1, 'a', new Date('21 Dec 1997 14:12:00 UTC')];  
2 const localeString = arr.toLocaleString('en', { timeZone: 'UTC' });  
3 console.log(localeString); // Output: "1,a,12/21/1997, 2:12:00 PM"
```

# JavaScript Array Methods

## toReversed() method

The `toReversed()` method is the copying counterpart of the `reverse()` method. It returns a new array with the elements in reversed order.



```
1 const items = [1, 2, 3];
2 console.log(items); // [1, 2, 3]
3
4 const reversedItems = items.toReversed();
5 console.log(reversedItems); // [3, 2, 1]
6 console.log(items); // [1, 2, 3]
```

# JavaScript Array Methods

## toSorted() method

The `toSorted()` method returns a new array with the elements sorted in ascending order.



```
1 const values = [1, 10, 21, 2];
2 const sortedValues = values.toSorted((a, b) => a - b);
3 console.log(sortedValues); // [1, 2, 10, 21]
4 console.log(values); // [1, 10, 21, 2]
```

# JavaScript Array Methods

## toSpliced() method

The `toLocaleString()` method returns a string with a language-sensitive representation of the array.



```
1 const months = ["Jan", "Mar", "Apr", "May"];
2
3 // Inserting an element at index 1
4 const months2 = months.toSpliced(1, 0, "Feb");
5 console.log(months2); // ["Jan", "Feb", "Mar", "Apr", "May"]
```

# JavaScript Array Methods

## toString() method

The `toString()` method returns a string representing the array and its elements.



```
1 const arr = [1, 2, 3];
2 console.log(arr.toString()); // Output: "1,2,3"
```

# JavaScript Array Methods

## unshift() method

The `unshift()` method adds one or more elements to the beginning of an array and returns the new length of the array.



```
1 const arr = [1, 2, 3];
2 const length = arr.unshift(0);
3 console.log(length); // Output: 4
4 console.log(arr); //
```

# JavaScript Array Methods

## values() method

The values() method returns an iterator that generates values for each element in the array.



```
1 const arr = ["a", "b", "c", "d", "e"];
2 const iterator = arr.values();
3
4 for (const letter of iterator) {
5   console.log(letter);
6 } // "a" "b" "c" "d" "e"
```

# JavaScript Array Methods

## with() method

The values() method returns an iterator that generates values for each element in the array.



```
1 const arr = [1, 2, 3, 4, 5];
2 console.log(arr.with(2, 6)); // [1, 2, 6, 4, 5]
```

# JavaScript Array Methods

---

**Which of these JavaScript  
Array methods do you  
use in your projects?**

---

Let's connect and expand our  
professional network together