JavaScript Array Methods

push()

Adds one or more elements to the end of an array.

```
const arr = [1, 2, 3]; arr.push(4); // arr is now [1, 2, 3, 4]
```

pop()

Removes the last element from an array.

```
const arr = [1, 2, 3]; arr.pop(); // arr is now [1, 2]
```

shift()

Removes the first element from an array.

```
const arr = [1, 2, 3]; arr.shift(); // arr is now [2, 3]
```

unshift()

Adds one or more elements to the beginning of an array.

```
const arr = [1, 2, 3]; arr.unshift(0); // arr is now [0, 1, 2, 3]
```

concat()

Merges two or more arrays.

```
const arr = [1, 2]; const newArr = arr.concat([3, 4]); // newArr is [1, 2, 3, 4]
```

join()

Joins all elements of an array into a string.

```
const arr = [1, 2, 3]; const str = arr.join('-'); // str is '1-2-3'
```

slice()

Returns a shallow copy of a portion of an array.

```
const arr = [1, 2, 3, 4]; const newArr = arr.slice(1, 3); // newArr is [2, 3]
```

splice()

Adds/removes elements from an array.

```
const arr = [1, 2, 3, 4]; arr.splice(1, 2); // arr is now [1, 4]
```

indexOf()

Returns the first index at which a given element can be found.

```
const arr = [1, 2, 3]; const index = arr.indexOf(2); // index is 1
```

lastIndexOf()

Returns the last index at which a given element can be found.

```
const arr = [1, 2, 3, 2]; const index = arr.lastIndexOf(2); // index is 3
```

includes()

Determines whether an array includes a certain value.

```
const arr = [1, 2, 3]; const hasTwo = arr.includes(2); // hasTwo is true
```

forEach()

Executes a provided function once for each array element.

```
const arr = [1, 2, 3]; arr.forEach(el => console.log(el)); // logs 1, 2, 3
```

map()

Creates a new array with the results of calling a function on every element.

```
const arr = [1, 2, 3]; const newArr = arr.map(x => x * 2); // newArr is [2, 4, 6]
```

filter()

Creates a new array with all elements that pass the test implemented by the provided function.

```
const arr = [1, 2, 3]; const newArr = arr.filter(x => x > 1); // newArr is [2, 3]
```

reduce()

Executes a reducer function on each element, resulting in a single output value.

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

reduceRight()

Executes a reducer function on each element of the array (from right-to-left) to reduce it to a single value.

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

some()

Tests whether at least one element in the array passes the test implemented by the provided function.

```
const arr = [1, 2, 3]; const hasEven = arr.some(x => x % 2 === 0); // hasEven is true
```

every()

Tests whether all elements in the array pass the test implemented by the provided function.

```
const arr = [1, 2, 3]; const allEven = arr.every(x => x % 2 === 0); // allEven is false
```

find()

Returns the first element that satisfies the provided testing function.

const arr = [1, 2, 3]; const found = arr.find(x => x > 1); // found is 2

findIndex()

Returns the index of the first element that satisfies the provided testing function.

const arr = [1, 2, 3]; const index = arr.findIndex(x => x > 1); // index is 1

sort()

Sorts the elements of an array in place and returns the array.

const arr = [3, 1, 2]; arr.sort(); // arr is now [1, 2, 3]

reverse()

Reverses an array in place.

const arr = [1, 2, 3]; arr.reverse(); // arr is now [3, 2, 1]

fill()

Fills all the elements of an array from a start index to an end index with a static value.

const arr = [1, 2, 3]; arr.fill(0, 1, 2); // arr is now [1, 0, 3]

copyWithin()

Shallow copies part of an array to another location in the same array.

const arr = [1, 2, 3, 4]; arr.copyWithin(1, 2); // arr is now [1, 3, 4, 4]

flat()

Creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.

const arr = [1, [2, [3]]]; const flatArr = arr.flat(2); // flatArr is [1, 2, 3]

flatMap()

Maps each element using a mapping function, then flattens the result into a new array.

const arr = [1, 2, 3]; const newArr = arr.flatMap(x => [x, x * 2]); // newArr is [1, 2, 2, 4, 3, 6]

from()

Creates a new array instance from an array-like or iterable object.

const str = '123'; const arr = Array.from(str); // arr is ['1', '2', '3']

isArray()

Determines whether the passed value is an array.

const isArray = Array.isArray([1, 2, 3]); // isArray is true

of()

Creates a new Array instance with a variable number of arguments.

const arr = Array.of(1, 2, 3); // arr is [1, 2, 3]

entries()

Returns a new Array Iterator object that contains the key/value pairs for each index in the array.

const arr = ['a', 'b', 'c']; const iterator = arr.entries(); console.log(iterator.next().value); // [0, 'a']

keys()

Returns a new Array Iterator that contains the keys for each index in the array.

const arr = ['a', 'b', 'c']; const iterator = arr.keys(); console.log(iterator.next().value); // 0

values()

Returns a new Array Iterator object that contains the values for each index in the array.

const arr = ['a', 'b', 'c']; const iterator = arr.values(); console.log(iterator.next().value); // 'a'

findLast()

Returns the last element in the array that satisfies the provided testing function.

const arr = [1, 2, 3, 2]; const found = arr.findLast(x => x > 1); // found is 2

findLastIndex()

Returns the index of the last element in the array that satisfies the provided testing function.

const arr = [1, 2, 3, 2]; const index = arr.findLastIndex(x > 1); // index is 3

toLocaleString()

Returns a string representing the elements of the array using their toLocaleString methods.

const arr = [1, 'a', new Date()]; const str = arr.toLocaleString(); // str depends on locale, e.g., '1,a,12/31/2023, 10:00:00 AM'

toString()

Returns a string representing the array and its elements.

const arr = [1, 2, 3]; const str = arr.toString(); // str is '1,2,3'

with()

Returns a new array with the given index replaced by a new value.

```
const arr = [1, 2, 3]; const newArr = arr.with(1, 4); // newArr is [1, 4, 3]
```

at()

Takes an integer value and returns the item at that index.

```
const arr = [1, 2, 3]; const val = arr.at(-1); // val is 3
```