Arrays

methods

▼ The official way to access the last value

```
let last = fruits[fruits.length - 1]
```

▼ How to loop over an array with forEach()?

```
fruits.forEach(function(item, index, array) {
  console.log(item, index)
})
// Apple 0
// Banana 1
```

▼ How to add an item to the end of an Array?

push

```
let newLength = fruits.push('Orange')
// ["Apple", "Banana", "Orange"]
```

▼ How to remove an item from the end of an Array?

Removes the last element from an array and returns that element.

pop

```
let last = fruits.pop() // remove Orange (from the end)
// ["Apple", "Banana"]
```

▼ How to remove an item from the **beginning** of an Array?

shift

```
let first = fruits.shift() // remove Apple from the front
// ["Banana"]
```

▼ How to add an item to the beginning of an Array?

```
let newLength = fruits.unshift('Strawberry') // add to the front
// ["Strawberry", "Banana"]
```

▼ How to find the index of an item in the Array?

```
let pos = fruits.indexOf('Banana')
// 1
```

▼ How to remove an item by index position?

```
let removedItem = fruits.splice(pos, 1) // this is how to remove an item

// ["Strawberry", "Mango"]
```

▼ How to make a shallow copy of an array?

```
let shallowCopy = fruits.slice() // this is how to make a copy
// ["Strawberry", "Mango"]
```

▼ How to remove items from an index position?

```
let pos = 1
let n = 2

let removedItems = vegetables.splice(pos, n)
// this is how to remove items, n defines the number of items to be removed,
// starting at the index position specified by pos and progressing toward the end of array.
```

▼ What is the Array constructor?

new Array() creates a new array.

```
// construct from elements
new Array(element0, element1, ..., elementN)
```

▼ What does Array.prototype.concat() do?

The concat() method is used to merge two or more arrays.

```
const array1 = ['a', 'b', 'c'];
const array2 = ['d', 'e', 'f'];
const array3 = array1.concat(array2);

console.log(array3);
// expected output: Array ["a", "b", "c", "d", "e", "f"]
```

▼ What does Array.prototype.fill() do?

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

```
const array1 = [1, 2, 3, 4];

// fill with 0 from position 2 until position 4
console.log(array1.fill(0, 2, 4));
// expected output: [1, 2, 0, 0]

// fill with 5 from position 1
console.log(array1.fill(5, 1));
// expected output: [1, 5, 5, 5]

console.log(array1.fill(6));
// expected output: [6, 6, 6, 6]
```

▼ What does Array.prototype.filter() do?

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

```
const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];

const result = words.filter(word => word.length > 6);
```

```
console.log(result);
// expected output: Array ["exuberant", "destruction", "present"]
```

▼ What does Array.prototype.find() do?

The find() method returns the value of the **first** element in the provided array that satisfies the provided testing function.

```
const array1 = [5, 12, 8, 130, 44];

const found = array1.find(element => element > 10);

console.log(found);
// expected output: 12
```

▼ What does Array.prototype.findIndex() do?

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

```
const array1 = [5, 12, 8, 130, 44];

const isLargeNumber = (element) => element > 13;

console.log(array1.findIndex(isLargeNumber));
// expected output: 3
```

▼ What does Array.prototype.includes() do?

The includes() method determines whether an array includes a certain value among its entries, returning true or false as appropriate.

```
const array1 = [1, 2, 3];

console.log(array1.includes(2));
// expected output: true

const pets = ['cat', 'dog', 'bat'];

console.log(pets.includes('cat'));
// expected output: true

console.log(pets.includes('at'));
// expected output: false
```

▼ What does Array.prototype.indexOf() do?

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

```
const beasts = ['ant', 'bison', 'camel', 'duck', 'bison'];

console.log(beasts.indexOf('bison'));
// expected output: 1

// start from index 2
console.log(beasts.indexOf('bison', 2));
// expected output: 4

console.log(beasts.indexOf('giraffe'));
// expected output: -1
```

▼ What does Array.prototype.reduce() do?

The reduce() method executes a reducer function (that you provide) on each element of the array, resulting in a single output value.

```
const array1 = [1, 2, 3, 4];
const reducer = (accumulator, currentValue) => accumulator + currentValue;

// 1 + 2 + 3 + 4
console.log(array1.reduce(reducer));
// expected output: 10

// 5 + 1 + 2 + 3 + 4
console.log(array1.reduce(reducer, 5));
// expected output: 15
```

▼ What does Array.prototype.slice() do?

The slice() method returns a shallow copy of a portion of an array into a new array object selected from start to end (end not included) where start and end represent the index of items in that array.

The original array will not be modified.

```
const animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];

console.log(animals.slice(2));
```

```
// expected output: Array ["camel", "duck", "elephant"]

console.log(animals.slice(2, 4));
// expected output: Array ["camel", "duck"]

console.log(animals.slice(1, 5));
// expected output: Array ["bison", "camel", "duck", "elephant"]

console.log(animals.slice(-2));
// expected output: Array ["duck", "elephant"]

console.log(animals.slice(2, -1));
// expected output: Array ["camel", "duck"]
```

▼ What does Array.prototype.sort() do?

The sort() method sorts the elements of an array in place and returns the sorted array. The default sort order is ascending.

```
const months = ['March', 'Jan', 'Feb', 'Dec'];
months.sort();
console.log(months);
// expected output: Array ["Dec", "Feb", "Jan", "March"]

const array1 = [1, 30, 4, 21, 100000];
array1.sort();
console.log(array1);
// expected output: Array [1, 100000, 21, 30, 4]
```

▼ What does Array.prototype.splice() do? - modify an array

The splice() method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.

```
const months = ['Jan', 'March', 'April', 'June'];
months.splice(1, 0, 'Feb');
// inserts at index 1
console.log(months);
// expected output: Array ["Jan", "Feb", "March", "April", "June"]

months.splice(4, 1, 'May');
// replaces 1 element at index 4
console.log(months);
// expected output: Array ["Jan", "Feb", "March", "April", "May"]
```

▼ What does Array.prototype.map() do?

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

```
const array1 = [1, 4, 9, 16];

// pass a function to map
const map1 = array1.map(x => x * 2);

console.log(map1);
// expected output: Array [2, 8, 18, 32]
```