Does it mutate 😱

.concat

no mutation

Description

The concat() method is used to merge two or more arrays. This method does not change the existing arrays, but instead returns a new array.

```
Array.prototype.concat ( [ item1 [ , item2 [ , … ] ] ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/concat

Example

```
var array1 = ['a', 'b', 'c'];
var array2 = ['d', 'e', 'f'];

console.log(array1.concat(array2));
// expected output: Array ["a", "b", "c", "d", "e", "f"]
```

.copyWithin()

mutates

Description

The copyWithin() method shallow copies part of an array to another location in the same array and returns it, without modifying its size.

```
arr.copyWithin(target)
arr.copyWithin(target, start)
```

```
arr.copyWithin(target, start, end)
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/copyWithin

Example

```
var array1 = ['a', 'b', 'c', 'd', 'e'];

// copy to index 0 the element at index 3
console.log(array1.copyWithin(0, 3, 4));
// expected output: Array ["d", "b", "c", "d", "e"]

// copy to index 1 all elements from index 3 to the end
console.log(array1.copyWithin(1, 3));
// expected output: Array ["d", "d", "e", "d", "e"]
```

.entries()

no mutation

Description

The entries() method returns a new Array Iterator object that contains the key/value pairs for each index in the array.

```
a.entries()
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/entries

Example

```
var array1 = ['a', 'b', 'c'];

var iterator1 = array1.entries();

console.log(iterator1.next().value);
```

```
// expected output: Array [0, "a"]

console.log(iterator1.next().value);
// expected output: Array [1, "b"]
```

.every

no mutation

Description

The every() method tests whether all elements in the array pass the test implemented by the provided function.

```
Array.prototype.every ( callbackfn [ , thisArg ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/every

Example

```
function isBelowThreshold(currentValue) {
  return currentValue < 40;
}

var array1 = [1, 30, 39, 29, 10, 13];

console.log(array1.every(isBelowThreshold));
// expected output: true
```

.fill()

mutates

Description

The fill() method fills all the elements of an array from a start index to an end index with a static value.

```
arr.fill(value)
arr.fill(value, start)
arr.fill(value, start, end)
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/fill

Example

```
var 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]
```

.filter

no mutation

Description

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

```
Array.prototype.filter ( callbackfn [ , thisArg ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

Example

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

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

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

.find()

no mutation

Description

The find() method returns a value of the first element in the array that satisfies the provided testing function. Otherwise undefined is returned.

```
arr.find(callback[, thisArg])
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find

Example

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

var found = array1.find(function(element) {
  return element > 10;
});

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

.findIndex()

no mutation

Description

The findIndex() method returns an index of the first element in the array that satisfies the provided testing function. Otherwise -1 is returned.

```
arr.findIndex(callback[, thisArg])
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex

Example

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

function isLargeNumber(element) {
  return element > 13;
}

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

.flat

no mutation

Description

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

```
var newArray = arr.flat([depth]);
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/flat

Example

```
var arr1 = [1, 2, [3, 4]];
arr1.flat();
// [1, 2, 3, 4]

var arr2 = [1, 2, [3, 4, [5, 6]]];
arr2.flat();
// [1, 2, 3, 4, [5, 6]]

var arr3 = [1, 2, [3, 4, [5, 6]]];
arr3.flat(2);
// [1, 2, 3, 4, 5, 6]

var arr4 = [1, 2, [3, 4, [5, 6, [7, 8, [9, 10]]]]];
arr4.flat(Infinity);
// [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

.forEach

no mutation

Description

The forEach() method executes a provided function once per array element.

```
Array.prototype.forEach ( callbackfn [ , thisArg ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach

Example

```
var array1 = ['a', 'b', 'c'];

array1.forEach(function(element) {
  console.log(element);
});

// expected output: "a"
```

```
// expected output: "b"
// expected output: "c"
```

.includes()

no mutation

Description

The includes() method determines whether an array includes a certain element, returning true or false as appropriate.

```
arr.includes(searchElement)
arr.includes(searchElement, fromIndex)
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/includes

Example

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

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

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

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

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

.indexOf

no mutation

Description

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.

```
Array.prototype.indexOf ( searchElement [ , fromIndex ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/indexOf

Example

```
var 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
```

.join

no mutation

Description

The join() method joins all elements of an array into a string.

```
Array.prototype.join (separator)
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/join

Example

```
var elements = ['Fire', 'Air', 'Water'];

console.log(elements.join());
// expected output: "Fire,Air,Water"

console.log(elements.join(''));
// expected output: "FireAirWater"

console.log(elements.join('-'));
// expected output: "Fire-Air-Water"
```

.keys()

no mutation

Description

The keys() method returns a new Array Iterator that contains the keys for each index in the array.

```
arr.keys()
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/keys

Example

```
var array1 = ['a', 'b', 'c'];
var iterator = array1.keys();

for (let key of iterator) {
  console.log(key); // expected output: 0 1 2
}
```

.lastIndexOf

no mutation

Description

The lastIndexOf() method returns the last index at which a given element can be found in the array, or -1 if it is not present. The array is searched backwards, starting at fromIndex.

```
Array.prototype.lastIndexOf ( searchElement [ , fromIndex ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/lastIndexOf

Example

```
var animals = ['Dodo', 'Tiger', 'Penguin', 'Dodo'];

console.log(animals.lastIndexOf('Dodo'));
// expected output: 3

console.log(animals.lastIndexOf('Tiger'));
// expected output: 1
```

.map

no mutation

Description

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

```
Array.prototype.map ( callbackfn [ , thisArg ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

Example

```
var 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]
```

.pop

mutates

Description

The pop() method removes the last element from an array and returns that element. This method changes the length of the array.

```
Array.prototype.pop ( )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/pop

Example

```
var plants = ['broccoli', 'cauliflower', 'cabbage', 'kale', 'tomato'];

console.log(plants.pop());
// expected output: "tomato"

console.log(plants);
// expected output: Array ["broccoli", "cauliflower", "cabbage", "kale"]

plants.pop();

console.log(plants);
// expected output: Array ["broccoli", "cauliflower", "cabbage"]
```

.push

mutates

Description

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

```
Array.prototype.push ( [ item1 [ , item2 [ , … ] ] ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push

Example

```
var animals = ['pigs', 'goats', 'sheep'];

console.log(animals.push('cows'));
// expected output: 4

console.log(animals);
// expected output: Array ["pigs", "goats", "sheep", "cows"]

animals.push('chickens');

console.log(animals);
// expected output: Array ["pigs", "goats", "sheep", "cows", "chickens"]
```

.reduce

no mutation

Description

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

```
Array.prototype.reduce ( callbackfn [ , initialValue ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce

Example

```
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
```

.reduceRight

no mutation

Description

The reduceRight() method applies a function against an accumulator and each value of the array (from right-to-left) has to reduce it to a single value.

```
Array.prototype.reduceRight ( callbackfn [ , initialValue ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/ReduceRight

Example

```
const array1 = [[0, 1], [2, 3], [4, 5]].reduceRight(
  (accumulator, currentValue) => accumulator.concat(currentValue)
);

console.log(array1);
// expected output: Array [4, 5, 2, 3, 0, 1]
```

.reverse

mutates

Description

The reverse() method reverses an array in place. The first array element becomes the last, and the last array element becomes the first.

```
Array.prototype.reverse ( )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reverse

Example

```
var array1 = ['one', 'two', 'three'];
console.log('array1: ', array1);
// expected output: Array ['one', 'two', 'three']

var reversed = array1.reverse();
console.log('reversed: ', reversed);
// expected output: Array ['three', 'two', 'one']

/* Careful: reverse is destructive. It also changes
the original array */
console.log('array1: ', array1);
// expected output: Array ['three', 'two', 'one']
```

.shift

mutates

Description

The shift() method removes the first element from an array and returns that element. This method changes the length of the array.

```
Array.prototype.shift ( )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/shift

Example

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

var firstElement = array1.shift();

console.log(array1);
// expected output: Array [2, 3]

console.log(firstElement);
// expected output: 1
```

.slice

no mutation

Description

The slice() method returns a shallow copy of a portion of an array into a new array object selected from begin to end (end not included). The original array will not be modified.

```
Array.prototype.slice (start, end)
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice

Example

```
var 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"]
```

.some

no mutation

Description

The some() method tests whether some element in the array passes the test implemented by the provided function.

```
Array.prototype.some ( callbackfn [ , thisArg ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/some

Example

```
var array = [1, 2, 3, 4, 5];

var even = function(element) {
  // checks whether an element is even
  return element % 2 === 0;
};

console.log(array.some(even));
// expected output: true
```

.sort

mutates

Description

The sort() method sorts the elements of an array in place and returns the array. The sort is not necessarily stable. The default sort order is according to string Unicode code points.

```
Array.prototype.sort (comparefn)
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

Example

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

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

.splice

mutates

Description

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

```
Array.prototype.splice (start, deleteCount [ , item1 [ , item2 [ , … ] ] ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

Example

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

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

.toLocaleString

no mutation

Description

The toLocaleString() method returns a string representing the elements of the array. The elements are converted to Strings using their toLocaleString methods and these Strings are separated by a locale-specific String (such as a comma “,”).

```
Array.prototype.toLocaleString ( )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/toLocaleString

Example

```
var array1 = [1, 'a', new Date('21 Dec 1997 14:12:00 UTC')];
var localeString = array1.toLocaleString('en', {timeZone: "UTC"});
```

```
console.log(localeString);
// expected output: "1,a,12/21/1997, 2:12:00 PM",
// This assumes "en" locale and UTC timezone - your results may vary
```

.toSource()

no mutation

Description

The toSource() method returns a string representing the source code of the array.

```
arr.toSource()
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/toSource

Example

```
var alpha = new Array('a', 'b', 'c');

alpha.toSource();
//returns ['a', 'b', 'c']
```

.toString

no mutation

Description

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

```
Array.prototype.toString ( )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/toString

Example

```
var array1 = [1, 2, 'a', '1a'];

console.log(array1.toString());
// expected output: "1,2,a,1a"
```

.unshift

mutates

Description

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

```
Array.prototype.unshift ( [ item1 [ , item2 [ , … ] ] ] )
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift

Example

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

console.log(array1.unshift(4, 5));
// expected output: 5

console.log(array1);
// expected output: Array [4, 5, 1, 2, 3]
```

.values()

no mutation

Description

The values() method returns a new Array Iterator object that contains the values for each index in the array.

```
arr.values()
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/values

Example

```
const array1 = ['a', 'b', 'c'];
const iterator = array1.values();

for (const value of iterator) {
  console.log(value); // expected output: "a" "b" "c"
}
```

Made by @rem