

Array.prototype.copyWithin()

Web technology for developers ▸ JavaScript ▸ JavaScript reference ▸ Standard built-in objects ▸ Array ▸ Array.prototype.copyWithin()

English ▾

On this Page

[Syntax](#)

[Description](#)

[Examples](#)

[Polyfill](#)

[Specifications](#)

[Browser compatibility](#)

[See also](#)

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

JavaScript Demo: Array.copyWithin()

```
1 const array1 = ['a', 'b', 'c', 'd', 'e'];
2
3 // copy to index 0 the element at index 3
4 console.log(array1.copyWithin(0, 3, 4));
5 // expected output: Array ["d", "b", "c", "d", "e"]
6
7 // copy to index 1 all elements from index 3 to the end
8 console.log(array1.copyWithin(1, 3));
9 // expected output: Array ["d", "d", "e", "d", "e"]
10
```

Run ›

Reset

```
> Array ["d", "b", "c", "d", "e"]
> Array ["d", "d", "e", "d", "e"]
```

Related Topics

Standard built-in objects

Array

Properties


[Array.length](#)

[Array.prototype\[\[@@unscopables\]\(#\)\]](#)

Methods

[Array.from\(\)](#)

[Array.isArray\(\)](#)

 [Array.observe\(\)](#)

[Array.of\(\)](#)

[Array.prototype.concat\(\)](#)

[Array.prototype.copyWithin\(\)](#)

[Array.prototype.entries\(\)](#)

[Array.prototype.every\(\)](#)

[Array.prototype.fill\(\)](#)

[Array.prototype.filter\(\)](#)

[Array.prototype.find\(\)](#)

Syntax

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

Parameters

target

`Array.prototype.findIndex()`

`Array.prototype.flat()`

`Array.prototype.flatMap()`

`Array.prototype.forEach()`

`Array.prototype.includes()`

`Array.prototype.indexOf()`

`Array.prototype.join()`

`Array.prototype.keys()`

`Array.prototype.lastIndexOf()`

`Array.prototype.map()`

`Array.prototype.pop()`

`Array.prototype.push()`

`Array.prototype.reduce()`

`Array.prototype.reduceRight()`

`Array.prototype.reverse()`

`Array.prototype.shift()`

`Array.prototype.slice()`

`Array.prototype.some()`

`Array.prototype.sort()`

`Array.prototype.splice()`

`Array.prototype.toLocaleString()`

⚠️ `Array.prototype.toSource()`

`Array.prototype.toString()`

`Array.prototype.unshift()`

`Array.prototype.values()`

`Array.prototype[@@iterator]()`

📦 `Array.unobserve()`

`get Array[@@species]`

Inheritance:

Function

Properties

🗨️ `Function.arguments`

📦 `Function.arity`

⚠️ `Function.caller`

⚠️ `Function.displayName`

`Function.length`

Zero-based index at which to copy the sequence to. If negative, `target` will be counted from the end.

If `target` is at or greater than `arr.length`, nothing will be copied. If `target` is positioned after `start`, the copied sequence will be trimmed to fit `arr.length`.

start | Optional

Zero-based index at which to start copying elements from. If negative, `start` will be counted from the end.

If `start` is omitted, `copyWithin` will copy from index 0.

end | Optional

Zero-based index at which to end copying elements from. `copyWithin` copies up to but not including `end`. If negative, `end` will be counted from the end.

If `end` is omitted, `copyWithin` will copy until the last index (default to `arr.length`).

Return value

The modified array.

Description

The `copyWithin` works like C and C++'s `memmove`, and is a high-performance method to shift the data of an `Array`. This especially applies to the `TypedArray` method of the same name. The sequence is copied and pasted as one operation; pasted sequence will have the copied values even when the copy and paste region overlap.

The `copyWithin` function is intentionally *generic*, it does not require that its `this` value be an `Array` object.

The `copyWithin` method is a mutable method. It does not alter the length of `this`, but it will change its content and create new properties, if necessary.

Examples

```
const arr = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'];
```

Function.name

Function.prototype

Methods

Function.prototype.apply()

Function.prototype.bind()

Function.prototype.call()

⚠️ Function.prototype.isGenerator()

⚠️ Function.prototype.toSource()

Function.prototype.toString()

Object

Properties

⚠️ Object.prototype.__count__

⚠️ Object.prototype.__noSuchMethod__

⚠️ Object.prototype.__parent__

👤 Object.prototype.__proto__

Object.prototype.constructor

Methods

👤 Object.prototype.__defineGetter__()

👤 Object.prototype.__defineSetter__()

👤 Object.prototype.__lookupGetter__()

👤 Object.prototype.__lookupSetter__()

Object.prototype.hasOwnProperty()

Object.prototype.isPrototypeOf()

Object.prototype.propertyIsEnumerable()

Object.prototype.toLocaleString()

⚠️ Object.prototype.toSource()

Object.prototype.toString()

👤 ⚠️ Object.prototype.unwatch()

Object.prototype.valueOf()

👤 ⚠️ Object.prototype.watch()

Object.setPrototypeOf()

```
1 [1, 2, 3, 4, 5].copyWithin(-2);
2 // [1, 2, 3, 1, 2]
3
4 [1, 2, 3, 4, 5].copyWithin(0, 3);
5 // [4, 5, 3, 4, 5]
6
7 [1, 2, 3, 4, 5].copyWithin(0, 3, 4);
8 // [4, 2, 3, 4, 5]
9
10 [1, 2, 3, 4, 5].copyWithin(-2, -3, -1);
11 // [1, 2, 3, 3, 4]
12
13 [].copyWithin.call({length: 5, 3: 1}, 0, 3);
14 // {0: 1, 3: 1, length: 5}
15
16 // ES2015 Typed Arrays are subclasses of Array
17 var i32a = new Int32Array([1, 2, 3, 4, 5]);
18
19 i32a.copyWithin(0, 2);
20 // Int32Array [3, 4, 5, 4, 5]
21
22 // On platforms that are not yet ES2015 compliant:
23 [].copyWithin.call(new Int32Array([1, 2, 3, 4, 5]), 0, 3, 4);
24 // Int32Array [4, 2, 3, 4, 5]
```

Polyfill

```
1 if (!Array.prototype.copyWithin) {
2   Object.defineProperty(Array.prototype, 'copyWithin', {
3     value: function(target, start/*, end*/) {
4       // Steps 1-2.
5       if (this == null) {
6         throw new TypeError('this is null or not defined');
7       }
8
9       var 0 = Object(this);
10
11       // Steps 3-5.
12       var len = 0.length >>> 0;
13
14       // Steps 6-8.
15       var relativeTarget = target >> 0;
16
```

```
17     var to = relativeTarget < 0 ?
18         Math.max(len + relativeTarget, 0) :
19         Math.min(relativeTarget, len);
20
21     // Steps 9-11.
22     var relativeStart = start >> 0;
23
24     var from = relativeStart < 0 ?
25         Math.max(len + relativeStart, 0) :
26         Math.min(relativeStart, len);
27
28     // Steps 12-14.
29     var end = arguments[2];
30     var relativeEnd = end === undefined ? len : end >> 0;
31
32     var final = relativeEnd < 0 ?
33         Math.max(len + relativeEnd, 0) :
34         Math.min(relativeEnd, len);
35
36     // Step 15.
37     var count = Math.min(final - from, len - to);
38
39     // Steps 16-17.
40     var direction = 1;
41
42     if (from < to && to < (from + count)) {
43         direction = -1;
44         from += count - 1;
45         to += count - 1;
46     }
47
48     // Step 18.
49     while (count > 0) {
50         if (from in O) {
51             O[to] = O[from];
52         } else {
53             delete O[to];
54         }
55
56         from += direction;
57         to += direction;
58         count--;
59     }
60
61     // Step 19.
62     return 0;
63 },
64 configurable: true,
```

















```
65     writable: true
66   });
67 }
```

Specifications


Specification	Status	Comment
ECMAScript 2015 (6th Edition, ECMA-262) The definition of 'Array.prototype.copyWithin' in that specification.	ST Standard	Initial definition.
ECMAScript 2016 (ECMA-262) The definition of 'Array.prototype.copyWithin' in that specification.	ST Standard	
ECMAScript Latest Draft (ECMA-262) The definition of 'Array.prototype.copyWithin' in that specification.	D Draft	


Browser compatibility

[Update compatibility data on GitHub](#)

													
	 Chrome	 Edge	 Firefox	 Internet Explorer	 Opera	 Safari	 Android webview	 Chrome for Android	 Firefox for Android	 Opera for Android	 Safari on iOS	 Samsung Internet	 Node.js
copyWithin	45	12	32	No	32	9	Yes	45	32	Yes	Yes	Yes	4.0.0

What are we missing?

 Full support

 No support

See also

- [Array](#)

Last modified: May 4, 2019, by MDN contributors

Learn the best of web development

Get the latest and greatest from MDN delivered straight to your inbox.

[Sign up now](#)



moz://a

[Terms](#) [Privacy](#) [Cookies](#)

© 2005-2019 Mozilla and individual contributors. Content is available under [these licenses](#).

MDN

[Web Technologies](#)

[Learn Web Development](#)

[About MDN](#)

[Feedback](#)



Mozilla

[About](#)

[Contact Us](#)

[Firefox](#)

