Array.prototype.copyWithin()

copyWithin() 方法浅复制数组的一部分到同一数组中的另一个位置，并返回它，而不修改其大小。

```
["alpha", "beta", "copy", "delta"].copyWithin(1, 2, 3);
// 0:"alpha" 1:"beta" 2:"copy" 3:"delta"
// ["alpha", "copy", "copy", "delta"]
// 0:"alpha" 1:"copy" 2:"copy" 3:"delta"

// target === 1:"beta"
// start === 2:"copy",
// end === 3:"delta"

// 1:"beta" => 1:"copy"

['alpha', 'bravo', 'charlie', 'delta'].copyWithin(2, 0);

// results in ["alpha", "bravo", "alpha", "bravo"]
```

语法

```
arr.copyWithin(target)

arr.copyWithin(target, start)

arr.copyWithin(target, start, end)

arr.copyWithin(目标索引, [源开始索引], [结束源索引])
```

参数

target

0 为基底的索引，复制序列到该位置。如果是负数， `target` 将从末尾开始计算。

如果 `target` 大于等于 `arr.length` ，将会不发生拷贝。如果 `target` 在 `start` 之后，复制的序列将被修改以符合 `arr.length` 。

start

0 为基底的索引，开始复制元素的起始位置。如果是负数， `start` 将从末尾开始计算。

如果 `start` 被忽略， `copyWithin` 将会从0开始复制。

`end`

0 为基底的索引，开始复制元素的结束位置。 `copyWithin` 将会拷贝到该位置，但不包括 `end` 这个位置的元素。如果是负数， `end` 将从末尾开始计算。

如果 `end` 被忽略， copyWithin 将会复制到 `arr.length` 。

返回值

改变了的数组。

描述

参数target,start和end 必须为整数。

如果start为负，则其指定的索引位置等同于length+start，length为数组的长度。end也是如此。

copyWithin方法不要求其this值必须是一个数组对象；除此之外，copyWithin是一个可变方法，它可以改变this对象本身，并且返回它，而不仅仅是它的拷贝。

copyWithin 就像 C 和 C++ 的 `memcpy` 函数一样，且它是用来移动 `Array` 或者 `TypedArray` 数据的一个高性能的方法。复制以及粘贴序列这两者是为一体的操作;即使复制和粘贴区域重叠，粘贴的序列也会有拷贝来的值。

copyWithin 函数是设计为通用的，其不要求其 `this` 值必须是一个 数组 对象。

The **copyWithin** 是一个可变方法，它不会改变 this 的 length，但是会改变 this 本身的内容，且需要时会创建新的属性。

例子

```
[1, 2, 3, 4, 5].copyWithin(-2);
// [1, 2, 3, 1, 2]

[1, 2, 3, 4, 5].copyWithin(0, 3);
// [4, 5, 3, 4, 5]

[1, 2, 3, 4, 5].copyWithin(0, 3, 4);
// [4, 2, 3, 4, 5]
```

```
[1, 2, 3, 4, 5].copyWithin(-2, -3, -1);
// [1, 2, 3, 3, 4]

[].copyWithin.call({length: 5, 3: 1}, 0, 3);
// {0: 1, 3: 1, length: 5}

// ES2015 Typed Arrays are subclasses of Array
var i32a = new Int32Array([1, 2, 3, 4, 5]);

i32a.copyWithin(0, 2);
// Int32Array [3, 4, 5, 4, 5]

// On platforms that are not yet ES2015 compliant:
[].copyWithin.call(new Int32Array([1, 2, 3, 4, 5]), 0, 3, 4);
// Int32Array [4, 2, 3, 4, 5]
```

Polyfill

```
if (!Array.prototype.copyWithin) {
  Array.prototype.copyWithin = function(target, start/*, end*/) {
    // Steps 1-2.
    if (this == null) {
      throw new TypeError('this is null or not defined');
    }

    var O = Object(this);

    // Steps 3-5.
    var len = O.length >>> 0;

    // Steps 6-8.
    var relativeTarget = target >> 0;

    var to = relativeTarget < 0 ?
      Math.max(len + relativeTarget, 0) :
      Math.min(relativeTarget, len);

    // Steps 9-11.
    var relativeStart = start >> 0;

    var from = relativeStart < 0 ?
```

```
      Math.max(len + relativeStart, 0) :
      Math.min(relativeStart, len);

    // Steps 12-14.
    var end = arguments[2];
    var relativeEnd = end === undefined ? len : end >> 0;

    var final = relativeEnd < 0 ?
      Math.max(len + relativeEnd, 0) :
      Math.min(relativeEnd, len);

    // Step 15.
    var count = Math.min(final - from, len - to);

    // Steps 16-17.
    var direction = 1;

    if (from < to && to < (from + count)) {
      direction = -1;
      from += count - 1;
      to += count - 1;
    }

    // Step 18.
    while (count > 0) {
      if (from in O) {
        O[to] = O[from];
      } else {
        delete O[to];
      }

      from += direction;
      to += direction;
      count--;
    }

    // Step 19.
    return O;
  };
}
```

规范

Specification	Status	Comment
ECMAScript 2015 (6th Edition, ECMA-262) Array.prototype.copyWithin	ST Standard	Initial definition.
ECMAScript 2016 (ECMA-262) Array.prototype.copyWithin	ST Standard	
ECMAScript Latest Draft (ECMA-262) Array.prototype.copyWithin	LS Living Standard	

浏览器支持

Desktop Mobile

Feature	Chrome	Edge	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	45	12	32 (32)	未实现	32	9.0

相关链接

- `Array`