Array.prototype.slice()

slice() 方法返回一个从开始到结束（不包括结束）选择的数组的一部分浅拷贝到一个新数组对象。原始数组不会被修改。

```
var a = ['zero', 'one', 'two', 'three'];
var sliced = a.slice(1, 3);

console.log(a);      // ['zero', 'one', 'two', 'three']
console.log(sliced); // ['one', 'two']
```

语法

```
arr.slice();
// [0, end]

arr.slice(begin);
// [begin, end]

arr.slice(begin, end);
// [begin, end)
```

参数

begin 可选

从该索引处开始提取原数组中的元素（从0开始）。

如果该参数为负数， 则表示从原数组中的倒数第几个元素开始提取， `slice(-2)` 表示提取原数组中的倒数第二个元素到最后一个元素（包含最后一个元素）。

如果省略 `begin`，则 `slice` 从索引 0 开始。

end 可选

在该索引处结束提取原数组元素（从0开始）。 `slice` 会提取原数组中索引从 `begin` 到 `end` 的所有元素（包含begin，但不包含end）。

`slice(1,4)` 提取原数组中的第二个元素开始直到第四个元素的所有元素 （索引为 1, 2, 3的元素）。

如果该参数为负数， 则它表示在原数组中的倒数第几个元素结束抽取 。 `slice(-2,-1)` 表示抽取了原数组中的倒数第二个元素到最后一个元素（不包含最后一个元素，也就是只有倒数第二个元素）。

如果 `end` 被省略， 则`slice` 会一直提取到原数组末尾。

如果 `end` 大于数组长度，`slice` 也会一直提取到原数组末尾。

返回值

一个含有提取元素的新数组

描述

`slice` 不修改原数组，只会返回一个浅复制了原数组中的元素的一个新数组。原数组的元素会按照下述规则拷贝：

- 如果该元素是个对象引用 （不是实际的对象）， `slice` 会拷贝这个对象引用到新的数组里。两个对象引用都引用了同一个对象。如果被引用的对象发生改变，则新的和原来的数组中的这个元素也会发生改变。

- 对于字符串、数字及布尔值来说（不是 `String` 、 `Number` 或者 `Boolean` 对象）， `slice` 会拷贝这些值到新的数组里。在别的数组里修改这些字符串或数字或是布尔值，将不会影响另一个数组。

如果向两个数组任一中添加了新元素，则另一个不会受到影响。

示例

返回现有数组的一部分

```
var fruits = ['Banana', 'Orange', 'Lemon', 'Apple', 'Mango'];
var citrus = fruits.slice(1, 3);

// fruits contains ['Banana', 'Orange', 'Lemon', 'Apple', 'Mango']
// citrus contains ['Orange','Lemon']
```

使用 slice

在下例中, slice从 myCar中创建了一个新数组 newCar .两个数组都包含了一个 myHonda 对象的引用. 当 myHonda 的color属性改变为purple, 则两个数组中的对应元素都会随之改变.

```
// 使用slice方法从myCar中创建一个newCar.
var myHonda = { color: 'red', wheels: 4, engine: { cylinders: 4, size: 2.
var myCar = [myHonda, 2, "cherry condition", "purchased 1997"];
var newCar = myCar.slice(0, 2);

// 输出myCar, newCar,以及各自的myHonda对象引用的color属性.
console.log('myCar = ' + JSON.stringify(myCar));
console.log('newCar = ' + JSON.stringify(newCar));
console.log('myCar[0].color = ' + JSON.stringify(myCar[0].color));
console.log('newCar[0].color = ' + JSON.stringify(newCar[0].color));

// 改变myHonda对象的color属性.
myHonda.color = 'purple';
console.log('The new color of my Honda is ' + myHonda.color);

//输出myCar, newCar中各自的myHonda对象引用的color属性.
console.log('myCar[0].color = ' + myCar[0].color);
console.log('newCar[0].color = ' + newCar[0].color);
```

上述代码输出:

```
myCar = [{color: 'red', wheels: 4, engine: {cylinders: 4, size: 2.2}}, 2,
       'cherry condition', 'purchased 1997']
newCar = [{color: 'red', wheels: 4, engine: {cylinders: 4, size: 2.2}}, 2
myCar[0].color = red
newCar[0].color = red
The new color of my Honda is purple
myCar[0].color = purple
newCar[0].color = purple
```

类似数组（Array-like）对象

slice 方法可以用来将一个类数组（Array-like）对象/集合转换成一个数组。你只需将该方法绑定到这个对象上。下述代码中 list 函数中的 arguments 就是一个类数组对象。

```
function list() {
  return Array.prototype.slice.call(arguments);
}

var list1 = list(1, 2, 3); // [1, 2, 3]
```

除了使用 `Array.prototype.slice.call( arguments )`，你也可以简单的使用 `[].slice.call(arguments)` 来代替。另外，你可以使用 `bind` 来简化该过程。

```
var unboundSlice = Array.prototype.slice;
var slice = Function.prototype.call.bind(unboundSlice);

function list() {
  return slice(arguments);
}

var list1 = list(1, 2, 3); // [1, 2, 3]
```

精简跨浏览器行为

根据规范，使用 `Array.prototype.slice` 转换宿主对象（如 DOM 对象）时不必遵循 Mozilla 的默认行为，即可以转化任何符合条件的伪数组宿主对象为数组，IE < 9 没有遵循，而 IE9 + 遵循这个行为，但是稍加改造可以使其在跨浏览器使用时更可靠。只要其他现代浏览器继续支持该行为，目前 IE 9+、FireFox、Chrome、Safari 以及 Opera 都支持，开发者在使用下面代码时遍历 DOM 时就不会被该方法的字面意义误导，即 IE < 9 不能转化 DOM Collections。开发者可以安全地根据语义知道该方法的实际上的标准行为。（下面的代码还修正了 IE 中 `slice()` 方法第二个参数不允许为显式的 `null` / `undefined` 值的问题，其他现代浏览器，包括IE9+都允许）。

```
/**
* Shim for "fixing" IE's lack of support (IE < 9) for applying slice
* on host objects like NamedNodeMap, NodeList, and HTMLCollection
* (technically, since host objects have been implementation-dependent,
* at least before ES6, IE hasn't needed to work this way).
* Also works on strings, fixes IE < 9 to allow an explicit undefined
* for the 2nd argument (as in Firefox), and prevents errors when
* called on other DOM objects.
*/
(function () {
    'use strict';
    var _slice = Array.prototype.slice;
```

```
13
14      try {
15          // Can't be used with DOM elements in IE < 9
16          _slice.call(document.documentElement);
17      } catch (e) { // Fails in IE < 9
18          // This will work for genuine arrays, array-like objects,
19          // NamedNodeMap (attributes, entities, notations),
20          // NodeList (e.g., getElementsByTagName), HTMLCollection (e.g., c
21          // and will not fail on other DOM objects (as do DOM elements in
22          Array.prototype.slice = function (begin, end) {
23              // IE < 9 gets unhappy with an undefined end argument
24              end = (typeof end !== 'undefined') ? end : this.length;
25
26              // For native Array objects, we use the native slice functior
27              if (Object.prototype.toString.call(this) === '[object Array]
28                  return _slice.call(this, begin, end);
29              }
30
31              // For array like object we handle it ourselves.
32              var i, cloned = [],
33                  size, len = this.length;
34
35              // Handle negative value for "begin"
36              var start = begin || 0;
37              start = (start >= 0) ? start: len + start;
38
39              // Handle negative value for "end"
40              var upTo = (end) ? end : len;
41              if (end < 0) {
42                  upTo = len + end;
43              }
44
45              // Actual expected size of the slice
46              size = upTo - start;
47
48              if (size > 0) {
49                  cloned = new Array(size);
50                  if (this.charAt) {
51                      for (i = 0; i < size; i++) {
52                          cloned[i] = this.charAt(start + i);
53                      }
54                  } else {
55                      for (i = 0; i < size; i++) {
```

```
                        cloned[i] = this[start + i];
                    }
                }
            }

            return cloned;
        };
    }
}());
```

规范

Specification	Status	Comment
ECMAScript 3rd Edition (ECMA-262)	ST Standard	Initial definition. Implemented in JavaScript 1.2
ECMAScript 5.1 (ECMA-262) Array.prototype.slice	ST Standard	
ECMAScript 2015 (6th Edition, ECMA-262) Array.prototype.slice	ST Standard	
ECMAScript Latest Draft (ECMA-262) Array.prototype.slice	LS Living Standard	

浏览器兼容性

Desktop Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	1.0	1.0 (1.7 or earlier)	(Yes)	(Yes)	(Yes)

相关链接

- `Array.prototype.splice()`
- `Function.prototype.call`
- `Function.prototype.bind`
- es6-in-deepth