Array.prototype.map()

map() 方法创建一个新数组，其结果是该数组中的每个元素都调用一个提供的函数后返回的结果。

```
let numbers = [1, 5, 10, 15];
let doubles = numbers.map((x) => {
   return x * 2;
});

// doubles is now [2, 10, 20, 30]
// numbers is still [1, 5, 10, 15]


let numbers = [1, 4, 9];
let roots = numbers.map(Math.sqrt);

// roots is now [1, 2, 3]
// numbers is still [1, 4, 9]
```

语法

```
let array = arr.map(function callback(currentValue, index, array) {
    // Return element for new_array
}[, thisArg])
```

参数

callback

生成新数组元素的函数，使用三个参数：

currentValue

callback 的第一个参数，数组中正在处理的当前元素。

index

callback 的第二个参数，数组中正在处理的当前元素的索引。

array

`callback` 的第三个参数， `map` 方法被调用的数组。

`thisArg`

可选的。执行 `callback` 函数时 使用的 `this` 值。

返回值

一个新数组，每个元素都是回调函数的结果。

描述

`map` 方法会给原数组中的每个元素都按顺序调用一次 `callback` 函数。 `callback` 每次执行后的返回值（包括 `undefined` ）组合起来形成一个新数组。 `callback` 函数只会在有值的索引上被调用；那些从来没被赋过值或者使用 `delete` 删除的索引则不会被调用。

`callback` 函数会被自动传入三个参数：数组元素，元素索引，原数组本身。

如果 `thisArg` 参数有值，则每次 `callback` 函数被调用的时候， `this` 都会指向 `thisArg` 参数上的这个对象。如果省略了 `thisArg` 参数, 或者赋值为 `null` 或 `undefined` ，则 this 指向全局对象 。

`map` 不修改调用它的原数组本身（当然可以在 `callback` 执行时改变原数组）。

使用 map 方法处理数组时，数组元素的范围是在 callback 方法第一次调用之前就已经确定了。在 map 方法执行的过程中：原数组中新增加的元素将不会被 callback 访问到；若已经存在的元素被改变或删除了，则它们的传递到 callback 的值是 map 方法遍历到它们的那一时刻的值；而被删除的元素将不会被访问到。

示例

求数组中每个元素的平方根

下面的代码创建了一个新数组，值为原数组中对应数字的平方根。

```
var numbers = [1, 4, 9];
var roots = numbers.map(Math.sqrt);
// roots的值为[1, 2, 3], numbers的值仍为[1, 4, 9]
```

使用 map 重新格式化数组中的对象

以下代码将一个包含对象的数组用以创建一个包含新重新格式化对象的新数组。

```
var kvArray = [{key: 1, value: 10},
               {key: 2, value: 20},
               {key: 3, value: 30}];

var reformattedArray = kvArray.map(function(obj) {
   var rObj = {};
   rObj[obj.key] = obj.value;
   return rObj;
});

// reformattedArray 数组为： [{1: 10}, {2: 20}, {3: 30}],

// kvArray 数组未被修改:
// [{key: 1, value: 10},
//  {key: 2, value: 20},
//  {key: 3, value: 30}]
```

用一个仅有一个参数的函数来mapping一个数字数组

下面的代码表示了当函数需要一个参数时map的工作方式。这个参数会遍历原始数组中的元素。

```
var numbers = [1, 4, 9];
var doubles = numbers.map(function(num) {
  return num * 2;
});

// doubles数组的值为： [2, 8, 18]
// numbers数组未被修改： [1, 4, 9]
```

一般的 map 方法

下面的例子演示如何在一个 `String` 上使用 map 方法获取字符串中每个字符所对应的 ASCII 码组成的数组：

```
var map = Array.prototype.map
var a = map.call("Hello World", function(x) {
  return x.charCodeAt(0);
})
// a的值为[72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100]
```

querySelectorAll 应用

下面代码展示了如何去遍历用 querySelectorAll 得到的动态对象集合。在这里，我们获得了文档里所有选中的选项，并将其打印：

```
var elems = document.querySelectorAll('select option:checked');
var values = Array.prototype.map.call(elems, function(obj) {
  return obj.value;
});
```

反转字符串

```
var str = '12345';
Array.prototype.map.call(str, function(x) {
  return x;
}).reverse().join('');

// 输出: '54321'
// Bonus: use '===' to test if original string was a palindrome
```

使用技巧案例

(原文地址)

通常情况下， map 方法中的 callback 函数只需要接受一个参数，就是正在被遍历的数组元素本身。但这并不意味着 map 只给 callback 传了一个参数。这个思维惯性可能会让我们犯一个很容易犯的错误。

```
// 下面的语句返回什么呢:
["1", "2", "3"].map(parseInt);
// 你可能觉的会是[1, 2, 3]
// 但实际的结果是 [1, NaN, NaN]

// 通常使用parseInt时,只需要传递一个参数.
// 但实际上,parseInt可以有两个参数.第二个参数是进制数.
// 可以通过语句"alert(parseInt.length)===2"来验证.
// map方法在调用callback函数时,会给它传递三个参数:当前正在遍历的元素,
// 元素索引, 原数组本身.
// 第三个参数parseInt会忽视, 但第二个参数不会,也就是说,
// parseInt把传过来的索引值当成进制数来使用.从而返回了NaN.
```

```
13
14 function returnInt(element) {
15   return parseInt(element, 10);
16 }
17
18 ['1', '2', '3'].map(returnInt); // [1, 2, 3]
19 // 意料之中的结果
20
21 // 也可以使用简单的箭头函数，结果同上
22 ['1', '2', '3'].map( str => parseInt(str) );
23
24 // 一个更简单的方式:
25 ['1', '2', '3'].map(Number); // [1, 2, 3]
26 // 与`parseInt` 不同，下面的结果会返回浮点数或指数:
27 ['1.1', '2.2e2', '3e300'].map(Number); // [1.1, 220, 3e+300]
```

兼容旧环境（Polyfill）

`map` 是在最近的 ECMA-262 标准中新添加的方法；所以一些旧版本的浏览器可能没有实现该方法。在那些没有原生支持 `map` 方法的浏览器中，你可以使用下面的 Javascript 代码来 实现它。 所使用的算法正是 ECMA-262，第 5 版规定的。假定 `Object`，`TypeError`，和 `Array` 有他们的原始值。而且 `callback.call` 的原始值也是 `Function.prototype.call`

```
1  // 实现 ECMA-262, Edition 5, 15.4.4.19
2  // 参考: http://es5.github.com/#x15.4.4.19
3  if (!Array.prototype.map) {
4    Array.prototype.map = function(callback, thisArg) {
5
6      var T, A, k;
7
8      if (this == null) {
9        throw new TypeError(" this is null or not defined");
10     }
11
12     // 1. 将O赋值为调用map方法的数组.
13     var O = Object(this);
14
15     // 2.将len赋值为数组O的长度.
16     var len = O.length >>> 0;
17
18     // 3.如果callback不是函数,则抛出TypeError异常.
19     if (Object.prototype.toString.call(callback) != "[object Function]") {
20       throw new TypeError(callback + " is not a function");
21     }
```

```
    // 4. 如果参数thisArg有值,则将T赋值为thisArg;否则T为undefined.
    if (thisArg) {
      T = thisArg;
    }

    // 5. 创建新数组A,长度为原数组O长度len
    A = new Array(len);

    // 6. 将k赋值为0
    k = 0;

    // 7. 当 k < len 时,执行循环.
    while(k < len) {

      var kValue, mappedValue;

      //遍历O,k为原数组索引
      if (k in O) {

        //kValue为索引k对应的值.
        kValue = O[ k ];

        // 执行callback,this指向T,参数有三个.分别是kValue:值,k:索引,O:原数组.
        mappedValue = callback.call(T, kValue, k, O);

        // 返回值添加到新数组A中.
        A[ k ] = mappedValue;
      }
      // k自增1
      k++;
    }

    // 8. 返回新数组A
    return A;
  };
}

```

规范

Specification	Status	Comment
ECMAScript 5.1 (ECMA-262) Array.prototype.map	ST Standard	Initial definition. Implemented in JavaScript 1.6
ECMAScript 2015 (6th Edition, ECMA-262) Array.prototype.map	ST Standard	

ECMAScript Latest Draft (ECMA-262) Array.prototype.map	LS Living Standard	

浏览器兼容性

Desktop Mobile

Feature	Firefox (Gecko)	Chrome	Internet Explorer	Opera	Safari
Basic support	(Yes)	1.5 (1.8)	9	(Yes)	(Yes)

相关链接

- `Array.prototype.forEach()`
- `Map` object
- `Array.from()`