

Array.prototype.forEach()

forEach() 方法对数组的每个元素执行一次提供的函数。

```
1  let a = ['a', 'b', 'c'];
2
3  a.forEach(function(element) {
4      console.log(element);
5  });
6
7  // a
8  // b
9  // c
```

语法

```
array.forEach(callback(currentValue, index, array){
    //do something
}, this)

array.forEach(callback[, thisArg])
```

参数

callback

为数组中每个元素执行的函数，该函数接收三个参数：

currentValue(当前值)

数组中正在处理的当前元素。

index(索引)

数组中正在处理的当前元素的索引。

array

forEach()方法正在操作的数组。

thisArg | 可选

可选参数。当执行回调 函数时 用作 `this` 的 值(参考对象)。

返回值

`undefined`。

描述

`forEach` 方法按升序为数组中含有效值的每一项执行一次 `callback` 函数，那些已删除（使用 `delete` 方法等情况）或者未初始化的项将被跳过（但不包括那些值为 `undefined` 的项）（例如在稀疏数组上）。


`callback` 函数会被依次传入三个参数：

- 数组当前项的值
- 数组当前项的索引
- 数组对象本身

如果 给`forEach`传递了`thisArg` 参数，当调用时，它将被传给 `callback` 函数，作为它的`this`值。否则，将会传入 `undefined` 作为它的`this`值。`callback`函数最终可观察到`this`值，这取决于 [函数观察到 `this` 的常用规则](#)。

`forEach` 遍历的范围在第一次调用 `callback` 前就会确定。调用 `forEach` 后添加到数组中的项不会被 `callback` 访问到。如果已经存在的值被改变，则传递给 `callback` 的值是 `forEach` 遍历到他们那一刻的值。已删除的项不会被遍历到。如果已访问的元素在迭代时被删除了(例如使用 `shift()`)，之后的元素将被跳过 - 参见下面的示例。

`forEach()` 为每个数组元素执行`callback`函数；不像 `map()` 或者 `reduce()`，它总是返回 `undefined` 值，并且不可链式调用。典型用例是在一个链的最后执行副作用。

 **注意：** 没有办法中止或者跳出 `forEach` 循环，除了抛出一个异常。如果你需要这样，使用`forEach()`方法是错误的，你可以用一个简单的循环作为替代。如果您正在测试一个数组里的元素是否符合某条件，且需要返回一个布尔值，那么可使用 `Array.every` 或 `Array.some`。如果可用，新方法 `find()` 或者 `findIndex()` 也可被用于真值测试的提早终止。

示例

打印出数组的内容

下面的代码会为每一个数组元素输出一行记录：

```
1 function logArrayElements(element, index, array) {
2     console.log("a[" + index + "] = " + element);
3 }
4
5 // 注意索引2被跳过了，因为在数组的这个位置没有项
6 [2, 5, ,9].forEach(logArrayElements);
7
8 // a[0] = 2
9 // a[1] = 5
10 // a[3] = 9
11
12 [2, 5, "", 9].forEach(logArrayElements);
13 // a[0] = 2
14 // a[1] = 5
15 // a[2] =
16 // a[3] = 9
17
18 [2, 5, undefined, 9].forEach(logArrayElements);
19 // a[0] = 2
20 // a[1] = 5
21 // a[2] = undefined
22 // a[3] = 9
23
24
25 let xxx;
26 // undefined
27
28 [2, 5, xxx, 9].forEach(logArrayElements);
29 // a[0] = 2
30 // a[1] = 5
31 // a[2] = undefined
32 // a[3] = 9
```

使用thisArg

举个勉强的例子，从每个数组中的元素值中更新一个对象的属性：

```
1 function Counter() {
2     this.sum = 0;
3     this.count = 0;
4 }
```

```
5
6 Counter.prototype.add = function(array) {
7     array.forEach(function(entry) {
8         this.sum += entry;
9         ++this.count;
10    }, this);
11    //console.log(this);
12 };
13
14 var obj = new Counter();
15 obj.add([1, 3, 5, 7]);
16
17 obj.count;
18 // 4 === (1+1+1+1)
19 obj.sum;
20 // 16 === (1+3+5+7)
```

因为 `thisArg` 参数 (`this`) 传给了 `forEach()`，每次调用时，它都被传给 `callback` 函数，作为它的 `this` 值。

❏ **注意：**如果使用[箭头函数表达式](#)传入函数参数，`thisArg` 参数会被忽略，因为箭头函数在词法上绑定了 `this` 值。

对象复制函数

下面的代码会创建一个给定对象的副本。创建对象的副本有不同的方法，以下是只是一种方法，并解释了 `Array.prototype.forEach()` 是如何使用 ECMAScript 5 `Object.*` 元属性（meta property）函数工作的。

```
1 function copy(obj) {
2     var copy = Object.create(Object.getPrototypeOf(obj));
3     var propNames = Object.getOwnPropertyNames(obj);
4
5     propNames.forEach(function(name) {
6         var desc = Object.getOwnPropertyDescriptor(obj, name);
7         Object.defineProperty(copy, name, desc);
8     });
9
10    return copy;
11 }
12
```

```
13 | var obj1 = { a: 1, b: 2 };
14 | var obj2 = copy(obj1); // obj2 looks like obj1 now
```

如果数组在迭代时被修改了，则其他元素会被跳过。

下面的例子输出"one", "two", "four"。当到达包含值"two"的项时，整个数组的第一个项被移除了，这导致所有剩下的项上移一个位置。因为元素 "four"现在在数组更前的位置，"three"会被跳过。

`forEach()` 不会在迭代之前创建数组的副本。

```
1 | var words = ["one", "two", "three", "four"];
2 | words.forEach(function(word) {
3 |     console.log(word);
4 |     if (word === "two") {
5 |         words.shift();
6 |     }
7 | });
8 | // one
9 | // two
10 | // four
```

兼容旧环境 (Polyfill)

`forEach` 是在第五版本里被添加到 ECMA-262 标准的；这样它可能在标准的其他实现中不存在，你可以在你调用 `forEach` 之前 插入下面的代码，在本地不支持的情况下使用 `forEach()` 。该算法是 ECMA-262 第5版中指定的算法。算法假定 `Object` 和 `TypeError` 拥有它们的初始值。 `callback.call` 等价于 `Function.prototype.call()` 。

```
1 | // Production steps of ECMA-262, Edition 5, 15.4.4.18
2 | // Reference: http://es5.github.io/#x15.4.4.18
3 | if (!Array.prototype.forEach) {
4 |
5 |     Array.prototype.forEach = function(callback, thisArg) {
6 |
7 |         var T, k;
8 |
9 |         if (this == null) {
10 |             throw new TypeError(' this is null or not defined');
11 |         }
12 |
13 |         // 1. Let 0 be the result of calling toObject() passing the
14 |         // |this| value as the argument.
```

```
15     var 0 = Object(this);
16
17     // 2. Let lenValue be the result of calling the Get() internal
18     // method of 0 with the argument "length".
19     // 3. Let len be toUint32(lenValue).
20     var len = 0.length >>> 0;
21
22     // 4. If isCallable(callback) is false, throw a TypeError exception.
23     // See: http://es5.github.com/#x9.11
24     if (typeof callback !== "function") {
25         throw new TypeError(callback + ' is not a function');
26     }
27
28     // 5. If thisArg was supplied, let T be thisArg; else let
29     // T be undefined.
30     if (arguments.length > 1) {
31         T = thisArg;
32     }
33
34     // 6. Let k be 0
35     k = 0;
36
37     // 7. Repeat, while k < len
38     while (k < len) {
39
40         var kValue;
41
42         // a. Let Pk be ToString(k).
43         // This is implicit for LHS operands of the in operator
44         // b. Let kPresent be the result of calling the HasProperty
45         // internal method of 0 with argument Pk.
46         // This step can be combined with c
47         // c. If kPresent is true, then
48         if (k in 0) {
49
50             // i. Let kValue be the result of calling the Get internal
51             // method of 0 with argument Pk.
52             kValue = 0[k];
53
54             // ii. Call the Call internal method of callback with T as
55             // the this value and argument list containing kValue, k, and 0.
56             callback.call(T, kValue, k, 0);
57         }
```

```
58      // d. Increase k by 1.
59      k++;
60  }
61  // 8. return undefined
62  };
63 }
```

规范

规范	状态	说明
ECMAScript 5.1 (ECMA-262) Array.prototype.forEach	<div><div></div>ST Standard</div>	初始定义。在JavaScript 1.6中实现。
ECMAScript 2015 (6th Edition, ECMA-262) Array.prototype.forEach	<div><div></div>ST Standard</div>	

浏览器兼容性

	Desktop	Mobile			
Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
基本支持	(Yes)	1.5 (1.8)	9	(Yes)	(Yes)

相关链接

- Array.prototype.find()
- Array.prototype.findIndex()
- Array.prototype.map()
- Array.prototype.every()
- Array.prototype.some()
- Map.prototype.forEach()

- `Set.prototype.forEach()`
- <http://www.webhek.com/javascript-loop-foreach-for-in-for-of>

