迭代方法

ECMAScript 5 为数组定义了 5 个迭代方法。每个方法都接收两个参数：要在每一项上运行的函数和（可选的）运行该函数的作用域对象——影响 this 的值。传入这些方法中的函数会接收三个参数：数组项的值、该项在数组中的位置和数组对象本身。根据使用的方法不同，这个函数执行后的返回值可能会也可能不会影响方法的返回值。以下是这 5 个迭代方法的作用。

传入这些方法中的函数会接收三个参数：数组项的值、该项在数组中的位置和数组对象本身。

- filter()：对数组中的每一项运行给定函数，返回该函数会返回 true 的项组成的数组。

下面再看一看 filter()函数，它利用指定的函数确定是否在返回的数组中包含某一项。例如，要返回一个所有数值都大于 2 的数组，可以使用以下代码。

```
var numbers = [1,2,3,4,5,4,3,2,1];

var filterResult = numbers.filter(function(item, index, array){
    return (item > 2);
});

alert(filterResult);       //[3,4,5,4,3]
```

- forEach()：对数组中的每一项运行给定函数。这个方法没有返回值。

最后一个方法是 forEach()，它只是对数组中的每一项运行传入的函数。这个方法没有返回值，本质上与使用 for 循环迭代数组一样。来看一个例子。

```
var numbers = [1,2,3,4,5,4,3,2,1];

numbers.forEach(function(item, index, array){
    //执行某些操作
});
```

- map()：对数组中的每一项运行给定函数，返回每次函数调用的结果组成的数组。

map()也返回一个数组，而这个数组的每一项都是在原始数组中的对应项上运行传入函数的结果。例如，可以给数组中的每一项乘以 2，然后返回这些乘积组成的数组，如下所示。

```
var numbers = [1,2,3,4,5,4,3,2,1];

var mapResult = numbers.map(function(item, index, array){
    return item * 2;
});

alert(mapResult);  //[2,4,6,8,10,8,6,4,2]
```

- every()：对数组中的每一项运行给定函数，如果该函数对每一项都返回 true，则返回 true。
- some()：对数组中的每一项运行给定函数，如果该函数对任一项返回 true，则返回 true。

以上方法都不会修改数组中的包含的值。

在这些方法中，最相似的是 every()和 some()，它们都用于查询数组中的项是否满足某个条件。对 every()来说，传入的函数必须对每一项都返回 true，这个方法才返回 true；否则，它就返回 false。而 some()方法则是只要传入的函数对数组中的某一项返回 true，就会返回 true。请看以下例子。

```
var numbers = [1,2,3,4,5,4,3,2,1];

var everyResult = numbers.every(function(item, index, array){
    return (item > 2);
});

alert(everyResult);     //false

var someResult = numbers.some(function(item, index, array){
    return (item > 2);
});

alert(someResult);      //true
```