JavaScript中Map和ForEach的区别

译者按： 惯用 Haskell 的我更爱 map。

- 原文: JavaScript — Map vs. ForEach - What's the difference between Map and ForEach in JavaScript?
- 译者: Fundebug

如果你已经有使用 JavaScript 的经验，你可能已经知道这两个看似相同的方法：`Array.prototype.map()`和`Array.prototype.forEach()`。

那么，它们到底有什么区别呢?

定义

我们首先来看一看 MDN 上对 Map 和 ForEach 的定义：

- `forEach()`: 针对每一个元素执行提供的函数(executes a provided function once for each array element)。
- `map()`: 创建一个新的数组，其中每一个元素由调用数组中的每一个元素执行提供的函数得来(creates a new array with the results of calling a provided function on every element in the calling array)。

到底有什么区别呢? `forEach()`方法不会返回执行结果，而是`undefined`。也就是说，`forEach()`会修改原来的数组。而`map()`方法会得到一个新的数组并返回。

示例

下方提供了一个数组，如果我们想将其中的每一个元素翻倍，我们可以使用`map`和`forEach`来达到目的。

```
let arr = [1, 2, 3, 4, 5];
```

ForEach

注意，`forEach`是不会返回有意义的值的。 我们在回调函数中直接修改`arr`的值。

```
arr.forEach((num, index) => {
    return (arr[index] = num * 2);
});
```

执行结果如下：

```
// arr = [2, 4, 6, 8, 10]
```

Map

```
let doubled = arr.map(num => {
    return num * 2;
});
```

执行结果如下：

```
// doubled = [2, 4, 6, 8, 10]
```

执行速度对比

jsPref是一个非常好的网站用来比较不同的 JavaScript 函数的执行速度。

这里是 `forEach()` 和 `map()` 的测试结果：

Testing in Chrome 61.0.3163 / Mac OS X 10.13.1		
Test		**Ops/sec**
ForEach	`arr.forEach((num, index) => { return arr[index] = num * 2; });`	1,072,096 ±1.13% 71% slower
Map	`let doubled = arr.map(num => { return num * 2; });`	3,650,087 ±1.32% fastest

可以看到，在我到电脑上 `forEach()` 的执行速度比 `map()` 慢了 70%。每个人的浏览器的执行结果会不一样。你可以使用下面的链接来测试一下: Map vs. forEach - jsPref。

函数式角度的理解

如果你习惯使用函数是编程，那么肯定喜欢使用 `map()` 。因为 `forEach()` 会改变原始的数组的值，而 `map()` 会返回一个全新的数组，原本的数组不受到影响。

哪个更好呢?

取决于你想要做什么。

`forEach` 适合于你并不打算改变数据的时候，而只是想用数据做一些事情 – 比如存入数据库或则打印出来。

```
let arr = ["a", "b", "c", "d"];
arr.forEach(letter => {
    console.log(letter);
});
// a
// b
// c
// d
```

`map()` 适用于你要改变数据值的时候。不仅仅在于它更快，而且返回一个新的数组。这样的优点在于你可以使用复合(composition)(map(), filter(), reduce()等组合使用)来玩出更多的花样。

```
let arr = [1, 2, 3, 4, 5];
let arr2 = arr.map(num => num * 2).filter(num => num > 5);
// arr2 = [6, 8, 10]
```

我们首先使用 map 将每一个元素乘以 2，然后紧接着筛选出那些大于 5 的元素。最终结果赋值给`arr2`。

核心要点

- 能用`forEach()`做到的，`map()`同样可以。反过来也是如此。
- `map()`会分配内存空间存储新数组并返回，`forEach()`不会返回数据。
- `forEach()`允许`callback`更改原始数组的元素。`map()`返回新的数组。