深入浅出ES6（二）：迭代器和for-of循环

作者：Vanessa
原文链接：https://ld246.com/article/1436236287834
来源网站：链滴

编者按：ECMAScript 6已经正式发布了，作为它最重要的方言，Javascript也即将迎来语法上的大变革，InfoQ特开设“<a href="http://www.infoq.com/cn/es6-in-depth/">深入浅出ES6/a>”专栏，来看一下ES6将给我们带来哪些新内容。本专栏文章来自Mozilla Web开发者博客，由作者授权翻译并发布。
我们如何遍历数组中的元素？20年前JavaScript刚萌生时，你可能这样实现数组遍历：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">for (var index = 0; index < myArray.length; index++) {
console.log(myArray[index]);
}</code></pre>
自ES5正式发布后，你可以使用内建的forEach方法来遍历数组：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
myArray.forEach(function (value) {
console.log(value)
});</code></pre>
这段代码看起来更加简洁，但这种方法也有一个小缺陷：你不能使用break语句中断环，也不能使用return语句返回到外层函数。
当然，如果只用for循环的语法来遍历数组元素也很不错。
那么，你一定想尝试一下for-in循环：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
for (var index in myArray) { // 千万别这样做
 console.log(myArray[index]);
}</code></pre>
这绝对是一个糟糕的选择，为什么呢？

在这段代码中，赋给index的值不是实际的数字，而是字符串“0”、“1”、“2”，此时很可能在无意之间进行字符串算数计算，例如：“2” + 1 == “21”，这给编码过程带来极大的不便。

作用于数组的for-in循环体除了遍历数组元素外，还会遍历自定义属性。举个例子，如果你的数组中有一个可枚举性myArray.name，循环将额外执行一次，遍历到名为“name”的索引。就连数组原链上的属性都能被访问到。
最让人震惊的是，在某些情况下，这段代码可能按照随机顺序遍历数组元素。
简而言之，for-in是为普通对象设计的，你可以遍历得到字符串类型的键，因此不适用于数组遍。

<h2>强大的for-of循环</h2>
还记得在《入浅出ES6（一）：ES6是什么》中我向你们承诺过的话么？ES6不会破坏你已经写好的JS代码目前看来，成千上万的Web网站依赖for-in循环，其中一些网站甚至将其用于数组遍历。如果想通过正for-in循环增加数组遍历支持会让这一切变得更加混乱，因此，标准委员会在ES6中增加了一种新循环语法来解决目前的问题。
就像这样：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
for (var value of myArray) {
  console.log(value);
}</code></pre>
是的，与之前的内建方法相比，这种循环方式看起来是否有些眼熟？那好，我们将要探究一下for-of循环的外表下隐藏着哪些强大的功能。现在，只需记住：

这是最简洁、最直接的遍历数组元素的语法
这个方法避开了for-in循环的所有缺陷
与forEach()不同的是，它可以正确响应break、continue和return语句

for-in循环用来遍历对象属性。
for-of循环用来遍历数据—例如数组中的值。
但是，不仅如此！
<h2>for-of循环也可以遍历其它的集合</h2>
for-of循环不仅支持数组，还支持大多数类数组对象，例如DOM NodeList对象。
for-of循环也支持字符串遍历，它将字符串视为一系列的Unicode字符来进行遍历：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
for (var chr of "") {
  alert(chr);
}</code></pre>
它同样支持Map和Set对象遍历。
对不起，你一定没听说过Map和Set对象。他们是ES6中新增的类型。我们将在后续的文章讲解两个新的类型。如果你曾在其它语言中使用过Map和Set，你会发现ES6中的并无太大出入。
举个例子，Set对象可以自动排除重复项：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
// 基于单词数组创建一个set对象

var uniqueWords = new Set(words);</code></pre>
生成Set对象后，你可以轻松遍历它所包含的内容：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
for (var word of uniqueWords) {
  console.log(word);
}</code></pre>
Map对象稍有不同：内含的数据由键值对组成，所以你需要使用解构（destructuring）来将键对拆解为两个独立的变量：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
for (var [key, value] of phoneBookMap) {
  console.log(key + "'s phone number is: " + value);
}</code></pre>
解构也是ES6的新特性，我们将在另一篇文章中讲解。看来我应该记录这些优秀的主题，未来有多的新内容需要一一剖析。
现在，你只需记住：未来的JS可以使用一些新型的集合类，甚至会有更多的类型陆续诞生，而for of就是为遍历所有这些集合特别设计的循环语句。
for-of循环不支持普通对象，但如果你想迭代一个对象的属性，你可以用for-in循环（这也是它本职工作）或内建的Object.keys()方法：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
// 向控制台输出对象的可枚举属性
for (var key of Object.keys(someObject)) {
  console.log(key + ": " + someObject[key]);
}</code></pre>
<h2>深入理解</h2>
<blockquote>
“能工摹形，巧匠窃意。”——巴勃罗·毕卡
</blockquote>

ES6始终坚持这样的宗旨：凡是新加入的特性，势必已在其它语言中得到强有力的实用性证明。
举个例子，新加入的for-of循环像极了C++、Java、C#以及Python中的循环语句。与它们一样这里的for-of循环支持语言和标准库中提供的几种不同的数据结构。它同样也是这门语言中的一个扩点（译注：关于扩展点，建议参考 1. 浅析扩展点 2. What are extensions and extension points?）。
正如其它语言中的for/foreach语句一样，for-of循环语句通过法调用来遍历各种集合。数组、Maps对象、Sets对象以及其它在我们讨论的对象有一个同点，它们都有一个迭代器方法。
你可以给任意类型的对象添加迭代器方法。
当你为对象添加myObject.toString()方法后，就可以将对象转化为字符串，同样地，当你向任对象添加myObject[Symbol.iterator]()方法，就可以遍历这个对象了。
举个例子，假设你正在使用jQuery，尽管你非常钟情于里面的.each()方法，但你还是想让jQuer对象也支持for-of循环，你可以这样做：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
// 因为jQuery对象与数组相似
// 可以为其添加与数组一致的迭代器方法
jQuery.prototype[Symbol.iterator] = Array.prototype[Symbol.iterator];</code></pre>
好的，我知道你在想什么，那个[Symbol.iterator]语法看起来很奇怪，这段代码到底做了什么呢这里通过Symbol处理了一下方法的名称。标准委员会可以把这个方法命名为.iterator()方法，但是如你的代码中的对象可能也有一些.iterator()方法，这一定会让你感到非常困惑。于是在ES6标准中使用symbol来作为方法名，而不是使用字符串。
你大概也猜到了，Symbols是ES6中的新类型，我们会在后续的文章中讲解。现在，你需要记住基于新标准，你可以定义一个全新的symbol，就像Symbol.iterator，如此一来可以保证不与任何已代码产生冲突。这样做的代价是，这段代码的语法看起来会略显生硬，但是这微乎其微代价却可以为带来如此多的新特性和新功能，并且你所做的这一切可以完美地向后兼容。
所有拥有[Symbol.iterator]()的对象被称为可迭代的。在接下来的文章中你会发现，可迭代对象概念几乎贯穿于整门语言之中，不仅是for-of循环，还有Map和Set构造函数、解构赋值，以及新的开操作符。
<h2>迭代器对象</h2>
现在，你将无须亲自从零开始实现一个对象迭代器，我们会在下一篇文章详细讲解。为了帮助你解本文，我们简单了解一下迭代器（如果你跳过这一章，你将错过非常精彩的技术细节）。
for-of循环首先调用集合的[Symbol.iterator]()方法，紧接着返回一个新的迭代器对象。迭代器象可以是任意具有.next()方法的对象；for-of循环将重复调用这个方法，每次循环调用一次。举个例，这段代码是我能想出来的最简单的迭代器：
<pre class=" language-javascript"><code class=" language-javascript" data-language="javascript">
var zeroesForeverIterator = {
 [Symbol.iterator]: function () {
  return this;
 },

```
  next<span class="token punctuation">:</span> <span class="token keyword">function</s
an> <span class="token punctuation">(</span><span class="token punctuation">)</span>
<span class="token punctuation">{</span>
  <span class="token keyword">return</span> <span class="token punctuation">{</span>
one<span class="token punctuation">:</span> <span class="token keyword">false</span
<span class="token punctuation">,</span> value<span class="token punctuation">:</span
 <span class="token number">0</span><span class="token punctuation">}</span><span c
ass="token punctuation">;</span>
 <span class="token punctuation">}</span>
<span class="token punctuation">}</span><span class="token punctuation">;</span></co
e></pre>
<p>每一次调用.next()方法，它都返回相同的结果，返回给for-of循环的结果有两种可能：(a) 我们未完成迭代；(b) 下一个值为0。这意味着(value of zeroesForeverIterator) {}将会是一个无限循环。然，一般来说迭代器不会如此简单。</p>
<p>这个迭代器的设计，以及它的.done和.value属性，从表面上看与其它语言中的迭代器不太一样在Java中，迭代器有分离的.hasNext()和.next()方法。在Python中，他们只有一个.next() 方法，当有更多值时抛出StopIteration异常。但是所有这三种设计从根本上讲都返回了相同的信息。</p>
<p>迭代器对象也可以实现可选的.return()和.throw(exc)方法。如果for-of循环过早退出会调用.retur
()方法，异常、break语句或return语句均可触发过早退出。如果迭代器需要执行一些清洁或释放资源操作，可以在.return()方法中实现。大多数迭代器方法无须实现这一方法。.throw(exc)方法的使用场就更特殊了：for-of循环永远不会调用它。但是我们还是会在下一篇文章更详细地讲解它的作用。</p

<p>现在我们已了解所有细节，可以写一个简单的for-of循环然后按照下面的方法调用重写被迭代的象。</p>
<p>首先是for-of循环：</p>
<pre class=" language-javascript"><code class=" language-javascript" data-language="javas
ript">
<span class="token keyword">for</span> <span class="token punctuation">(</span>VAR
f ITERABLE<span class="token punctuation">)</span> <span class="token punctuation">{</
pan>
  一些语句
<span class="token punctuation">}</span></code></pre>
<p>然后是一个使用以下方法和少许临时变量实现的与之前大致相当的示例，：</p>
<pre class=" language-javascript"><code class=" language-javascript" data-language="javas
ript">
<span class="token keyword">var</span> $iterator <span class="token operator">=</span
 ITERABLE<span class="token punctuation">[</span>Symbol<span class="token punctuatio
">.</span>iterator<span class="token punctuation">]</span><span class="token punctuati
n">(</span><span class="token punctuation">)</span><span class="token punctuation">;
/span>
<span class="token keyword">var</span> $result <span class="token operator">=</span>
$iterator<span class="token punctuation">.</span><span class="token function">next<spa
 class="token punctuation">(</span></span><span class="token punctuation">)</span><
pan class="token punctuation">;</span>
<span class="token keyword">while</span> <span class="token punctuation">(</span><s
an class="token operator">!</span>$result<span class="token punctuation">.</span>done
span class="token punctuation">)</span> <span class="token punctuation">{</span>
  VAR <span class="token operator">=</span> $result<span class="token punctuation">.</
pan>value<span class="token punctuation">;</span>
  一些语句
  $result <span class="token operator">=</span> $iterator<span class="token punctuation
>.</span><span class="token function">next<span class="token punctuation">(</span></
pan><span class="token punctuation">)</span><span class="token punctuation">;</span>
 <span class="token punctuation">}</span></code></pre>
```

这段代码没有展示.return()方法是如何处理的，我们可以添加这部分代码，但我认为这对于我们在讲解的内容来说过于复杂了。for-of循环用起来很简单，但是其背后有着非常复杂的机制。
<h2>我何时可以开始使用这一新特性？</h2>
目前，对于for-of循环新特性，所有最新版本Firefox都（部分）支持（译注：从FF 13开始陆续持相关功能，FF 36 - FF 40基本支持大部分特性），在Chrome中可以通过访问 chrome://flags 并用“实验性JavaScript”来支持。微软的Spartan浏览器支持，但是IE不支持。如果你想web环境中使用这种新语法，同时需要支持IE和Safari，你可以使用Bael或Google的Trceur这些编译器来将你的ES6代码翻译为Web友好的ES5代码。
而在服务端，你不需要类似的编译器，io.js中默认支持ES6新语法（部分），在Node中需要添加-harmony选项来启用相关特性。
{done: true}
喲！
好的，我们今天的讲解就到这里，但是对于for-of循环的使用远没有结束。
在ES6中有一种新的对象与for-of循环配合使用非常契合，我没有提及它因为它是我们下周文章主题，我认为这种新特性是ES6种最梦幻的地方，如果你尚未在类似Python和C#的语言中遇到它，一开始很可能会发现它令人难以置信，但是这是编写迭代器最简单的方式，在重构中非常有用，并且很可能改变我们书写异步代码的方式，无论是在浏览器环境还是服务器环境，所以，下周的深入浅出S6 中，请务必一起来仔细看看 ES6 的生成器：generators。
转自：http://www.infoq.com/cn/articles/es6-in-depth-iterators-and-the-for-of-loop?utm_source=infoq&utm_medium=related_content_link&utm_campaign=relatedContent_articles_clk
想要快速了解 ECMAScript6 可以参见 http://babeljs.io/docs/learn-es2015/