**（译）函数般调用正则**

2008-3-3 下午 - [**JS/Ajax/AS/Flex**](http://www.planabc.net/category/javascript-ajax/) - [javascript](http://www.planabc.net/tag/javascript/) - [正则](http://www.planabc.net/tag/regexp/)

Firefox 包含了一个非标准的 JavaScript 扩展，使正则像函数一样可调用。这为调用正则的 [exec](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Global_Objects:RegExp:exec) 方法提供了便捷。例如，在 Firefox中，regex(”string”) 等同于 regex.exec(”string”)。曾经 ECMAScript 4 建议指出这个功能将会增加到 ES4 规范中，但后来的在 [ES4-discuss mailing list](https://mail.mozilla.org/listinfo/es4-discuss) 的讨论中，这个建议可能被废除。

然而，你可以通过增加 [call](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Global_Objects:Function:call) 和 [apply](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Global_Objects:Function:apply) 方法到 RegExp.prototype 中类似的现实这些方法。既有助于功能设计，又可实现对函数和正则表达式均有效的隐藏类型（duck-typed ）代码。因此，让我们增加这些方法。

RegExp.prototype.call = function (context, str) {  
        return this.exec(str);  
};  
RegExp.prototype.apply = function (context, args) {  
        return this.exec(args[0]);  
};

注意上面的两个方法完全忽略 context 参数，你可以提交 null 或者 任何其他作为 context 的对象，并且你将会类似的得到正则 exec 方法的返回值。使用上面的方法，无论在什么情况下，使我们正常地使用正则表达式和函数变得容易得多。一些很明显的例子，比如这些在 JavaScript 1.6 的数组迭代中很有用。下面的 [filter](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Global_Objects:Array:filter), [every](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Global_Objects:Array:every), [some](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Global_Objects:Array:some), 和 [map](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Global_Objects:Array:map) 方法的执行可以跨浏览器。

if (!Array.prototype.filter) {  
        // 返回一个数组，如果提供的过滤函数返回 true，则返回存在的数组中的元素。  
        Array.prototype.filter = function (func, context) {  
                var results = [];  
                for (var i = 0; i < this.length; i++) {  
                        if (i in this && func.call(context, this[i], i, this))  
                        results.push(this[i]);  
                }  
                return results;  
        };  
}  
  
if (!Array.prototype.every) {  
        // 返回 true ，如果数组中的每个元素满足提供的测试函数。  
        Array.prototype.every = function (func, context) {  
                for (var i = 0; i < this.length; i++) {  
                        if (i in this && !func.call(context, this[i], i, this))  
                        return false;  
                }  
                return true;  
        };  
}  
  
if (!Array.prototype.some) {  
        // 返回 true，如果数组中至少有一个元素满足提供的测试函数。  
        Array.prototype.some = function (func, context) {  
                for (var i = 0; i < this.length; i++) {  
                        if (i in this && func.call(context, this[i], i, this))  
                        return true;  
                }  
                return false;  
        };  
}  
  
if (!Array.prototype.map) {  
        // 返回一个数组，现有数组中的每个元素调用提供的函数的返回值。  
        Array.prototype.map = function (func, context) {  
                var results = [];  
                for (var i = 0; i < this.length; i++) {  
                        if (i in this)  
                        results[i] = func.call(context, this[i], i, this);  
                }  
                return results;  
        };  
}

因为exec 方法返回数组或 null 值，并会恰当的类型转换为 true 和 false，上面的代码允许我们像这样使用：["a","b","ab","ba"].filter(/^a/)，返回所有以“a”开始的值：["a","ab"]。

确实，在 Firefox 中已经实现了 Array.prototype.filter ，由于 exec 的间接调用已经在该浏览器中起作用了。但是如果 filter 没有添加 RegExp.prototype.call 方法，却无法跨浏览器执行。

原文地址：[《Regular Expressions As Functions》](http://blog.stevenlevithan.com/archives/regular-expressions-as-functions)

这篇文章发表2008年3月3日 下午 21:03，并被分类于[JS/Ajax/AS/Flex](http://www.planabc.net/category/javascript-ajax/)，已被 29987 人阅读。 您可以通过订阅 [RSS 2.0](http://www.planabc.net/2008/03/03/regular_expressions_as_functions/feed/) 跟踪对这篇文章的评论， 也可以[发表您的评论](http://www.planabc.net/2008/03/03/regular_expressions_as_functions/#respond)，或者在您自己的网站中 [引用（trackback）](http://www.planabc.net/2008/03/03/regular_expressions_as_functions/trackback/) 该篇日志。