**关注”千锋互联”微信公众号( ID: qianfengjiaoyu ),更多学习资源(电子书/面试题/简历模板/视频/公开课等等)与你分享**



**必须知道的正则表达式**

在这篇文章中，我将开始解释，确切地说，正则表达式是什么，他们可以为你做什么。然后我将概述正则表达式最常见的特性（这一部分Perl爱好者们可以放心的跳过）。最后，我将用一个或两个实际的例子来帮助理解概念，解释在javascript中如何使用正则表达式。在这篇文章的结尾，你可能仍然是一个凡人，但你一定能用你新获得的文本技巧在聚会上给人留下好印象！

他们是什么 **正则表达式** 使用特殊的（在一开始，有点混乱）的代码来检测 **模式** 字符串。例如，如果你为你的网站游客提供了一个HTML表单输入他们的详细资料，你可能有一个字段来填入他们的电话号码。现在让我们面对这样的情景：一些网站游客比其他人更好的遵照说明。即使你在文本字段旁边放了一个小提示，指示电话号码所需的格式（例如：“(XXX) XXX-XXXX”北美地区的号码），一些人仍然会填错。写一个脚本来检查输入字符串的每一个字符，以确保所有的号码都是他们所属的地方，用圆括号和一个短线在正确的位置，将是相当繁琐的代码。同时一个电话号码是一个相当简单的问题！如果你不得不检查用户确实输入了一个电子邮件地址,或者更糟的是,一个URL，那又如何？

正则表达式提供了一个快速简单的方式去匹配一个字符串到一个模式。在我们电话号码的例子中，我们可以写一个简单的正则表达式并用它去检查-一个快速步骤-是否任何给定的字符串是一个正确格式化的电话号码。我们将进一步探讨这个例子，一旦我们已经注意了一些技术细节。

##他们看起来像什么## 正则表达式有时看起来想当复杂，但是当真正走近他时，他们实际上只是字符串。例如，下面是一个搜索文本‘JavaScript’的正则表达式（没有引号）：

`JavaScript`

它内容不多，对么？任何包含文本‘JavaScript’的字符串都匹配这个正则表达式。因此，这个正则表达式允许我们检测包含这个特定字符的字符串。好吧，坏消息是，它并不总是那么简单。

正如我上面提到的，正则表达式中可能会使用特殊代码。其中的一些可能彻头彻尾的混乱和难以记住，因此，如果你打算广泛使用它们，你可能希望为自己找到一个很好的参考。幸运地是，JavaScript支持和Perl非常相同的正则表达式语法，有大量网站记录Perl的正则表达式。在[Perl官方手册](http://perldoc.perl.org/perlre.html)中同样提供了文档。他们充满了神秘，所以你可能想要立刻参考他们。 Let’s work our way through a few examples to learn the basic regular expression syntax. 让我们通过几个例子来学习基本正则表达式语法。

**基本语法**

首先，尖冒号^可以用来表示字符串的开头，美元符号$可以用来标记结尾。

JavaScript //匹配"Isn't JavaScript great?"

^JavaScript //匹配以JavaScript为开头的字符串

//不能匹配 "What is JavaScript?"

JavaScript$ //匹配以JavaScript结尾的字符串

//不能匹配 "JavaScript is great!"

^JavaScript$ // 只匹配JavaScript

很明显，你有时候想要运用^，$或其他特殊字符在查询字符串中来代表相应的字符而不是正则表达中语法中定义的特殊含义。在字符前加反斜杠可以进行转义。

`\$\$\$ // 匹配"Show me the $$$!"

方括号可用于定义一组可匹配的字符集。例如，下面的正则表达式将匹配任何从1到5的数字

`[12345] // Matches "1" and "3", but not "a" or "12"` 匹配"1"和"3"，不匹配"12"

数字和字母的范围也可以指定

[1-5] //和上面的例子同理

[a-z] //匹配任何小写字母（来自英语字母表）

[0-9a-zA-Z] //匹配任何字母或数字

在方括号的开始位置插入一个^，你可以反转整个字符集，这意味着将匹配任何没有列出的字符：

`[^a-zA-Z] // 匹配字母以外的任意字符

?,+和\*也有特殊含义。具体来看，?意味着前面的字符是可选的（匹配零次或一次），+意味着前面的字符匹配一次或多次，\*意味着前面的字符匹配零次或多次

bana?na // 匹配"banana" 和 "banna"

// 不匹配"banaana"

bana+na // 匹配"banana" 和 "banaana"

// 不匹配"banna"

bana\*na // 匹配"banna", "banana", 和 "banaaana",

// 不匹配 "bnana"

^[a-zA-Z]+$ // 匹配任意字符一次或多次

// 只匹配字母

括号可用于将字符串分组，在运用?,+和&时将它们作为一个整体。

ba(na)+na // 匹配"banana" 和"banananana"

//不匹配"bana" 或 "banaana"

括号也可让你定义多个可能匹配的字符串，可采用|拆分它们。

^(ba|na)+$ // 匹配"banana", "nababa", "baba""nana", "ba", "na"和其他相关字符串

这里有一些特殊代码可用于在正则表达式中匹配字符串。

. // 除了换行符外的任意字符

\n // 换行符

\r // 回车符

\t // 制表符

\b //一个单词的边界（单词的开头或结尾）

\B //一个单词边界外的任意字符

\d //任意数字（等价于[0-9]）

\D // 除了数字外的任意字符 等价于[^0-9]

\s //空白字符（空格，制表符，换行符等）

\S //非空白符

\w //单词字符（等价于[A-Za-z0-9\_]）

\W //非单词字符（等价于[^A-Za-z0-9\_]）

注意，同我们之前看到的特殊字符对比，有些（如上面）需要加一个反斜杠来转化转移。

有更多特殊代码和正则表达式的语法技巧，所有的都应该涵盖任何完整的参考。现在，我们有足够多的目的。

在JavaScript中使用正则表达式

在JavaScript中使用正则表达式很容易，奇怪的是更多的人不知道这是可以做到的。你可以像下例一样创建一个正则表达式：

`var myRE = /regexp/;`

如上所述，regexp处是正则表达式代码。例如，下面的正则表达式创建了我在前一节提出的第一个示例，检测字符串JavaScript.

`var myRE = /JavaScript/;`

同样，这是如何创建最后一个例子

`var myRE = /^(ba|na)+$/;`

默认情况下,JavaScript正则表达式是大小写敏感的,并且在任何给定的字符串中只搜索第一个匹配项。通过在第二个/后添加g（全局）和i(不区分大小写)修饰符，你可以让一个正则表达式分别搜索所有匹配的字符串和不区分大小写。这里有一些正则表达式的例子。对于他们每一个，我将指明他们会匹配字符串"test1 Test2 TEST3"的哪个部分。 RegExp Match(es): 匹配 /Test[0-9]+/ “Test2” only 只匹配“Test2”

/Test[0-9]+/i “test1” only 只匹配“test1” /Test[0-9]+/gi “test1”, “Test2”, and “TEST3” 匹配“test1”, “Test2”, and “TEST3”

使用一个正则表达式是很容易的。每一个JavaScript变量包含一个文本字符串支持3个方法（或函数，如果你不习惯面向对象属于）处理正则表达式：match(), replace(), 和 search().

match()

match()用一个正则表达式作为参数，返回一个包含给出的字符串中发现的所有匹配的字符串组成的数组。如果没有发现任何匹配项，那么match()返回false.回到我们最初的例子，假设我们想要一个方法，它可以检查一个由用户输入的字符串作为她/他电话号是(XXX)XXX-XXX的形式.下面的代码就可以做到。

function checkPhoneNumber(phoneNo) {

var phoneRE = /^\(\d\d\d\) \d\d\d-\d\d\d\d$/;

if (phoneNo.match(phoneRE)) {

return true;

} else {

alert( "The phone number entered is invalid!" );

return false;

}

}

作为一个业务的第一单，这个方法定义了一个正则表达式。让我们把它分解一下，以理解它是如何工作的。这个正则表达式以^开头，表明任何匹配都必须在字符串开始处开始。接下来是\(，这将只匹配左括号。在正则表达式语法中我们在字母前加反斜杠作为前缀使他们进行转义（标记一组选择匹配的开始）。像上文提到的，\d是特殊字符匹配任何数字；因此\d\d\d匹配任意三位数字。我们可以写[0-9][0-9][0-9]来实现同样的效果，但是这样更短。剩下的部分不言自明。\)匹配右括号，空格匹配在电话号码中必须留下的空间，然后\d\d\d-\d\d\d\d匹配三位数字，短划线，四位数字。最后，$表明任何匹配必须在字符的结尾结束。 Incidentally, we could shorten this regular expression to the following, by using another shortcut that we did not mention above. If you can see how this works, you’re a natural! 顺带说一句，我们可以用以上没有提到的另一种快捷方式将这个正则表达式缩短至如下。如果你能看出它是如何工作的，你真是个天才！

`var phoneRE = /^\(\d{3}\) \d{3}-\d{4}$/;`

我们的方法接下来检查phoneNo.match(phoneRE)的结果是true还是false。换句话说，它检查phoneNo中包含的字符串是否匹配我们的正则表达式（因此返回一个数组，在JavaScript中评估为true）。如何发现一个匹配项，我们的方法返回true证明字符串确实是一个电话号码。如果没有，显示问题警告的消息并且方法返回false。 这种类型的方法最常见的用途是在允许表单提交前验证用户的输入。如果输入的信息格式不正确，我们可以通过在表单的onSubmit事件处理程序中调用我们的方法来中阻止表单提交。这是一个展示了使用我们checkPhoneNumber()方法的简单示例。

<form action="...">

<label>Enter phone number (e.g. (123) 456-7890):

<input type="text" name="phone">

</label>

<input type="submit">

</form>

<script>

var form = document.querySelector('form');

form.addEventListener('submit', function() {

return checkPhoneNumber(this.phone.value);

});

</script>

用户将无法提交表单，除非已经填入一个电话号码。任何尝试都将产生由我们checkPhoneNumber()方法生成的错误信息。 See the Pen [match() example](https://codepen.io/SitePoint/pen/QEwGoq/) by SitePoint ([@SitePoint](https://codepen.io/SitePoint)) on [CodePen](https://codepen.io/).

replace()

正如它的名字，replace()让你用一些新的字符串替换一个给定的正则表达式匹配项。假设你是一个难以正确拼写的人，并且想要遵守古老的拼写格言"I在E前，除非I前出现了C"来改正诸如"acheive"和“cieling”的拼写错误。我们需要一个方法，接收一个字符串并且执行2个操作，查询和替换。首先用"cei"替换“cie” Here’s the code:  
这里是代码：

`theString = theString.replace(/cie/gi,"cei");`

足够简单吧？第一个参数是我们搜索的正则表达式（注意，我们把它设为‘不区分大小写’和‘全局’模式，因此它可以找到所有的匹配项，而不是第一个），第二个参数是我们想要替换任意匹配项的字符串。 第二次更换更复杂。我们想要用在'x'是除去'c'以外任意字母的地方用"xie"替换“xei”。正则表达式检测的实体“xei”很容易理解。

`/[abd-z]ei/gi`

这个只检测了除了C以外的任意字母（包含‘a’,‘b’,‘d’到‘z’），其次是‘ei’,将它设为全局，并且不区分大小写模式。 复杂性在于定义我们的替换字符串。显然，我们想要用“xie”替换匹配想，但是困难在于写‘x’。记住，我们必须用匹配的字符串中的任意字母替换‘x’。要做到这一点，我们需要学习一个新的技巧。 之前，我向你们展示了如何使用括号来定义一组在一个正则表达式中的可选项（例如^(ba|na)+$）。事实证明，括号也有另外的含义。他们让我们‘记住’匹配的一部分，这样我们就可以在替换字符串中使用它。在这个案例中，我们要记住对应正则表达式中[abd-z]的匹配的部分。因此，我们用括号包裹他们。

`/([abd-z])ei/gi`

现在，当指定替换字符串时，我们把$1中插入到我们需要插入字符串的部分，对应正则表达式的括号部分。因此,执行需要替换的代码如下：

`theString = theString.replace(/([abd-z])ei/gi,"$1ie");`

总之，这是执行我们自我校对功能的完整方法。

function autoCorrect(theString) {

theString = theString.replace(/cie/gi,"cei");

theString = theString.replace(/([abd-z])ei/gi,"$1ie");

return theString;

}

See the Pen [replace() demo](https://codepen.io/SitePoint/pen/RRNoyw/) by SitePoint ([@SitePoint](https://codepen.io/SitePoint)) on [CodePen](https://codepen.io/).

在你去自己的页面使用这个方法之前，意识到这是“I在E前，除非I前出现了C”规则的例外。怪怪的，哈哈？

search()

search()方法类似于著名的indexOf()方法，除了它的参数是一个正则表达式而非一个字符串。然后搜索字符串针对给定的正则表达式的第一个匹配项并返回一个表明在字符串中位置的数字（例如，如果在字符串的开头匹配返回0，如果在字符串中匹配开始在第10位返回9），如果没有找到匹配项，方法返回-1。

var theString = "test1 Test2 TEST3";

theString.search(/Test[0-9]+/); // 6

总结 正则表达式是校验用户输入的宝贵工具。通过JavaScript中对正则表达式的应用，数据输入时就可以进行校验,提供流畅的用户体验（**注**：安全考虑，服务端校验仍然很必须，并且要注意在什么情况下JavaScript不可用）

**关注”千锋互联”微信公众号( ID: qianfengjiaoyu ),更多学习资源(电子书/面试题/简历模板/视频/公开课等等)与你分享**

