**为什么使用正则表达式？**

典型的搜索和替换操作要求您提供与预期的搜索结果匹配的确切文本。虽然这种技术对于对静态文本执行简单搜索和替换任务可能已经足够了，但它缺乏灵活性，若采用这种方法搜索动态文本，即使不是不可能，至少也会变得很困难。

通过使用正则表达式，可以：

测试字符串内的模式。  
例如，可以测试输入字符串，以查看字符串内是否出现电话号码模式或信用卡号码模式。这称为数据验证。

替换文本。  
可以使用正则表达式来识别文档中的特定文本，完全删除该文本或者用其他文本替换它。

基于模式匹配从字符串中提取子字符串。  
可以查找文档内或输入域内特定的文本。

例如，您可能需要搜索整个网站，删除过时的材料，以及替换某些 HTML 格式标记。在这种情况下，可以使用正则表达式来确定在每个文件中是否出现该材料或该 HTML 格式标记。此过程将受影响的文件列表缩小到包含需要删除或更改的材料的那些文件。然后可以使用正则表达式来删除过时的材料。最后，可以使用正则表达式来搜索和替换标记。

/^[0-9]+xy$/

^ 开始标志，

[0-9]+ 匹配一个或者多个数字，

[0-9] 匹配单个数字，

+ 匹配一个或多个，

xy 匹配xy，

$ 结束标志

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>正则表达式的使用</**title**>  
</**head**>  
<**script**> **var *str*** = **"1111xy"**;  
 **var *patt1*** = /^[0-9]+xy$/;  
 ***document***.write(***str***.match(***patt1***));  
</**script**>  
<**body**>  
  
  
</**body**>  
</**html**>

结果：

**1111xy**

**------------**

**var *str*** = **"1111xy#"**;   
**var *patt1*** = /^[0-9]+xy$/;  
***document***.write(***str***.match(***patt1***));

结果：

**null**

**---------------**

**var *str*** = **"0"**;  
**var *patt1*** = /^[1-9]\*$/;  
***document***.write(***str***.match(***patt1***));

结果：

**null**

**---------------**

**var *str*** = **"199"**;  
**var *patt2*** = /^[1-9][0-9]{0,}$/;  
***document***.write(***str***.match(***patt2***));*//199*

**--------------**

**var *str*** = **"199"**;  
**var *patt2*** = /^[1-9][0-9]{0,1}$/;  
***document***.write(***str***.match(***patt2***));*//null*

**--------------**

**var *str*** = **"10"**;  
**var *patt2*** = /^[1-9][0-9]{0,1}$/;  
***document***.write(***str***.match(***patt2***));*//10*

**--------------**

**var *str*** = **"1"**;  
**var *patt2*** = /^[1-9][0-9]+$/;  
***document***.write(***str***.match(***patt2***));*//null*

***-------------***

**var *str*** = **"10"**;  
**var *patt2*** = /^[1-9][0-9]+$/;  
***document***.write(***str***.match(***patt2***));*//10*

***-------------***

**var *str*** = **"<h1>正则表达式的使用</h1>"**;  
**var *patt2*** = /<.\*>/;  
***document***.write(***str***.match(***patt2***));*//正则表达式的使用*

***------------***

## 特殊字符

|  |  |
| --- | --- |
| 特别字符 | 描述 |
| $ | 匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 \$。 |
| ( ) | 标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 \( 和 \)。 |
| \* | 匹配前面的子表达式零次或多次。要匹配 \* 字符，请使用 \\*。 |
| + | 匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。 |
| . | 匹配除换行符 \n 之外的任何单字符。要匹配 . ，请使用 \. 。 |
| [ | 标记一个中括号表达式的开始。要匹配 [，请使用 \[。 |
| ? | 匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。 |
| \ | 将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 '\(' 则匹配 "("。 |
| ^ | 匹配输入字符串的开始位置，除非在方括号表达式中使用，当该符号在方括号表达式中使用时，表示不接受该方括号表达式中的字符集合。要匹配 ^ 字符本身，请使用 \^。 |
| { | 标记限定符表达式的开始。要匹配 {，请使用 \{。 |
| | | 指明两项之间的一个选择。要匹配 |，请使用 \|。 |

## 限定符

|  |  |
| --- | --- |
| **字符** | **描述** |
| \* | 匹配前面的子表达式零次或多次。例如，zo\* 能匹配 "z" 以及 "zoo"。\* 等价于{0,}。 |
| + | 匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。 |
| ? | 匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 、 "does" 中的 "does" 、 "doxy" 中的 "do" 。? 等价于 {0,1}。 |
| {n} | n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。 |
| {n,} | n 是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o\*'。 |
| {n,m} | m 和 n 均为非负整数，其中。n <= m最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。 |

**\*、+** **限定符都是贪婪的，因为它们会尽可能多的匹配文字，只有在它们的后面加上一个?就可以实现非贪婪或最小匹配。**

**var *str*** = **"<h1>正则表达式的使用</h1>"**;  
**var *patt2*** = /<.\*?>/;  
***document***.write(***str***.match(***patt2***));*//<h1>  
 //</h1>*

****-----------------****

## 定位符

定位符使您能够将正则表达式固定到行首或行尾。它们还使您能够创建这样的正则表达式，这些正则表达式出现在一个单词内、在一个单词的开头或者一个单词的结尾。

定位符用来描述字符串或单词的边界，**^** 和 **$** 分别指字符串的开始与结束，**\b** 描述单词的前或后边界，**\B** 表示非单词边界。

正则表达式的定位符有：

|  |  |
| --- | --- |
| ^ | 匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性，^ 还会与 \n 或 \r 之后的位置匹配。 |
| $ | 匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，$ 还会与 \n 或 \r 之前的位置匹配。 |
| \b | 匹配一个单词边界，即字与空格间的位置。 |
| \B | 非单词边界匹配。 |