正则表达式

> https://imweb.io/topic/56e804ef1a5f05dc50643106
>
> https://louiszhai.github.io/2016/06/13/regexp
>
> https://regex101.com
>
> https://www.cnblogs.com/shunyao8210/archive/2008/11/13/1332591.html
>
> 《精通正则表达式》

字符

- 元字符

元字符	描述
.	匹配除换行符以外的任意字符
\d	匹配数字, 等价于字符组[0-9]
\w	匹配字母, 数字, 下划线
\s	匹配任意的空白符(包括制表符,空格,换行等)
\b	匹配单词开始或结束的位置
^	匹配行首
$	匹配行尾

- 反义元字符

元字符	描述
\D	匹配非数字的任意字符, 等价于[^0-9]
\W	匹配除字母,数字,下划线之外的任意字符
\S	匹配非空白的任意字符
\B	匹配非单词开始或结束的位置
[^x]	匹配除x以外的任意字符

- 限定符

限定符	描述
*	x>=0
+	x>=1
?	x=0 or x=1
{n}	x=n
{n,}	x>=n
{n,m}	n<=x<=m

- 其他

字符类型	字符	描述
字符组	[…]	[…] 匹配中括号内字符之一. 如: [xyz] 匹配字符 x, y 或 z. 如果中括号中包含元字符, 则元字符降级为普通字符, 不再具有元字符的功能, 如 [+.?] 匹配 加号, 点号或问号.
排除性字符组	[^…]	[^…] 匹配任何未列出的字符,. 如: [^x] 匹配除x以外的任意字符.
多选结构	…|…|…	就是或的意思, 表示两者中的一个. 如: a
括号	(ab)+	括号 常用来界定重复限定符的范围, 以及将字符**分组**. 如: (ab)+ 可以匹配 abab..等, 其中 ab 便是一个分组.
转义字符	\ * + ?{ [( )] }^$.# 空白	\ 即转义字符, 通常 **\ * + ?

优先级（依次递减）

1. \ 转义符
2. (), (?:), (?=), [] 圆括号或方括号
3. *, +, ?, {n}, {n,}, {n,m} 限定符

4. ^, $ 位置
5. | “或” 操作

修饰符

修饰符	描述
g	全局匹配，有多少匹配多少
i	大小写不敏感
m	多行查找

分组

- 捕获型分组

```
reg = /([abc])/g // 其中的$1代表([abc])中捕获到的内容。
```

- 反向引用

```
reg = /([abc])\1/ // 其中 \1 就是反向引用部分
```

正则表达式匹配时，各个捕获性分组匹配到的内容，会依次保存在内存中一个特定的组里，通过`\+数字`的方式可以在正则中引用组里的内容，这种引用称作**反向引用**。捕获性分组匹配成功之前，它的内容的是不确定的，一旦匹配成功，组里的内容也就确定了。

反向引用常用来匹配重复出现的字符串，而不是重复出现的子表达式，这点要尤为注意。

- 非捕获型分组

非捕获性分组, 通常由一对括号加上”?:”加上子表达式组成, 非捕获性分组不会创建反向引用, 就好像没有括号一样. 如下:

```
var color = "#808080";
var output = color.replace(/#(?:\d+)/,"$1"+"~~");
console.log(RegExp.$1);//""
console.log(output);//$1~~
1234
```

以上, (?:\d+) 表示一个非捕获性分组, 由于分组不捕获任何内容, 所以, `RegExp.$1` 就指向了空字符串。同时, 由于`$1` 的反向引用不存在, 因此最终它被当成了普通字符串进行替换。

实际上, 捕获性分组和无捕获性分组在搜索效率方面也没什么不同, 没有哪一个比另一个更快。

*注意

- 字符组 和 多选结构

这里来看两个比较相似的例子：

regA: gr[ea]y

regB: gr(e|a)y

这两个正则表达式在匹域是相同的（匹配结果相同）。不过前者用的叫做"字符组"，后者是"多选结构"。需要注意两者的区别在于，"字符组"**只能**匹配目标文本中的**单个字符**，而"多选结构"中自身都可能是一个完整的正则表达式，都可以匹配**任意长度**的文本。

- 修饰符 `m`

`m` 用来表示可以多行匹配，进一步的解释可以说成是，**将每一行看作独立的单行**。只有在使用`^`和`$`的时候才有作用，其他时候不影响。

JS中的正则方法

RegExp：

方法	描述
`exec`	一个在字符串中执行查找匹配的RegExp方法，它返回一个数组（未匹配到则返回null）。
`test`	一个在字符串中测试是否匹配的RegExp方法，它返回true或false。

String：

方法	描述
`match`	一个在字符串中执行查找匹配的String方法，它返回一个数组或者在未匹配到时返回null。
`search`	一个在字符串中测试匹配的String方法，它返回匹配到的位置索引，或者在失败时返回-1。
`replace`	一个在字符串中执行查找匹配的String方法，并且使用替换字符串替换掉匹配到的子字符串。
`split`	一个使用正则表达式或者一个固定字符串分隔一个字符串，并将分隔后的子字符串存储到数组中的String方法。