**Group分组**

分组(…)的含义和用法：

1.(…) 用来匹配符合条件的字符串。并且将此部分，打包放在一起，看做成一个组，group。

2. 而此group，可以被后续的（正则表达式中）匹配时，所引用。

此处我称其为 前向引用，即前面已经通过group定义好的字符串，你在后面需要引用。引用的方式，是通过\N，其中N是对应的group的编号。

3.group的编号

编号为0的group，始终代表匹配的整个字符串；你在正则表达式内所看到的，通过括号括起来的group，编号分别对应着1,2,3，…

4.如果你想要在正则表达式中，匹配左括号'(‘，右括号’)’，其字符本身，则通过添加反斜杠，即’\(‘，’\)’的方式来匹配。

示例：

表达式：<(\w+)\s\*(\w+(=('|").\*?\4)?\s\*)\*>.\*?<\/\1>

文本： <td id='td1' style="bgcolor:white"></td>

$0: <td id='td1' style="bgcolor:white"></td>

$1: td （(\w+)）

$2: style="bgcolor:white" （(\w+(=('|").\*?\4)?\s\*)）

$3: ="bgcolor:white" （('|").\*?\4)）

$4: " （('|")）

这里的\4、\1 匹配的是表达式里的分组规则，表示这里的规则和前面的匹配要相同($4、$1)。

**命名分组 (?<name>xxx)**

与普通分组一样的功能，并且将匹配的子字符串捕获到一个组名称或编号名称中。在获得匹配结果时，可通过分组名进行获取。

命名捕获根据左括号的从左到右的顺序按顺序编号（与非命名捕获类似），但在对所有非命名捕获进行计数之后才开始对命名捕获进行编号。如果两个命名相同分组之间有包含关系，那么被包含的那个分组将不进行捕获。

**非捕获组**

**non-capturing group 非捕获组**（?:），取消对表达式的分组匹配（regexp里的$0,$1,$2） 。可以理解为将对应的匹配的内容，弄成一个逻辑的整体，即group，但是实际上并没有记录内容为group组，即去掉了对它的编号

示例

/(?:#|0x)?(?:[0-9A-F]{2}){3,4}/ig 匹配的结果

#FF006C 99AAB7FF 0xF0F73611 #FFADAa

若(?:#|0x)去掉？: ，那么它代表的group1会匹配出 “#”和“0x”

若(?:[0-9A-F]{2})去掉?: ，那么它代表的group2会匹配出字符末尾两位 “6C”，“FF”，“11”，“Aa”

补充说明

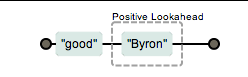
(?:[0-9A-F]{2}){3,4} 表示0-9或A-F的字符两个为一组匹配，且这个匹配重复3-4次，即表明6位到8位的由数字字母组成的字符串。

这样非捕获组匹配的结果是一个完整字符串主体如“#FF006C”而不是由$1和$2连个结果拼接出来的结果。

**前瞻表达式**



说的有些抽象，看个例子good(?=Byron)



代码如下:

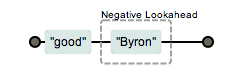
(/good(?=Byron)/).exec('goodByron123');['good']

(/good(?=Byron)/).exec('goodCasper123');//null

(/bad(?=Byron)/).exec('goodCasper123');//null

通过上面例子可以看出 exp1(?=exp2) 表达式会匹配exp1表达式，但只有其后面内容是exp2的时候才会匹配，也就是两个条件。

exp1(?!exp2) 比较类似

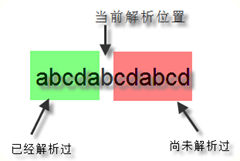


(/good(?!Byron)/).exec('goodByron123');//null

(/good(?!Byron)/).exec('goodCasper123');//['good']

(/bad(?!Byron)/).exec('goodCasper123');//null

正则表达式中有前瞻（Lookahead）和后顾（Lookbehind）的概念，这两个术语非常形象的描述了正则引擎的匹配行为。需要注意一点，正则表达式中的前和后和我们一般理解的前后有点不同。一段文本，我们一般习惯把文本开头的方向称作“前面”，文本末尾方向称为“后面”。但是对于正则表达式引擎来说，因为它是从文本头部向尾部开始解析的（可以通过正则选项控制解析方向），因此对于文本尾部方向，称为“前”，因为这个时候，正则引擎还没走到那块，而对文本头部方向，则称为“后”，因为正则引擎已经走过了那一块地方。如下图所示



所谓的前瞻就是在正则表达式匹配到某个字符的时候，往“尚未解析过的文本”预先看一下，看是不是符合/不符合匹配模式，而后顾，就是在正则引擎已经匹配过的文本看看是不是符合/不符合匹配模式。符合和不符合特定匹配模式我们又称为肯定式匹配和否定式匹配。

"1234567890".replace(/\B(?=(?:\d{3})+(?!\d))/g,",") // 结果：1,234,567,890，匹配的是后面是3\*n个数字的非单词边界(\B)

前瞻：

exp1(?=exp2) 查找exp2前面的exp1

后顾：

(?<=exp2)exp1 查找exp2后面的exp1

负前瞻：

exp1(?!exp2) 查找后面不是exp2的exp1

负后顾：

(?<!=exp2)exp1 查找前面不是exp2的exp1

举例：

"中国人".replace(/(?<=中国)人/, "rr") // 匹配中国人中的人，将其替换为rr，结果为 中国rr

"法国人".replace(/(?<=中国)人/, "rr") // 结果为 法国人，因为人前面不是中国，所以无法匹配到

要理解?:则需要理解捕获分组和非捕获分组的概念：

()表示捕获分组，()会把每个分组里的匹配的值保存起来，使用$n(n是一个数字，表示第n个捕获组的内容)

(?:)表示非捕获分组，和捕获分组唯一的区别在于，非捕获分组匹配的值不会保存起来