锚 — ^ 以及 $

The 匹配任意字符串以 The 为开头-> **试一下!**

end$ 匹配任意字符串以 end 为结尾

^The end$ 匹配字符串的(开头和结尾分别是 The end)

量词 — * + ? 以及 {}

abc* 匹配一个字符串具有 ab 其后有0个或者多个 c->试一下!

abc+ 匹配一个字符串具有 ab 其后有1个或者多个 c

abc? 匹配一个字符串具有 ab 其后有0个或者1个 c

abc{2} 匹配一个字符串具有 ab 其后有2个 c

abc{2,} 匹配一个字符串具有 ab 其后有2个或者多个 c

abc{2,5} 匹配一个字符串具有 ab 其后有2到5个 c

a(bc)* 匹配一个字符串具有 a 其后有0到多个 bc 的副本

a(bc){2,5} 匹配一个字符串具有 a 其后有0到5个 bc 的副本

OR 操作符 — | 或 []

a(b|c) 匹配一个字符串具有 a 其后有 b 或者 c -> 试一下!

a[bc] 与上一条相同

字符类 — \d \w \s 以及 .

\d 匹配一个数字字符-> 试一下!

\w 匹配一个单词字符(字母以及下划线) -> 试一下!

\s 匹配空白字符(包括 tab 以及换行)

. 匹配任意字符->试一下!

标志

- **g**(全局的) 在第一匹配之后不会立即返回，从前面匹配之后继续搜索
- **m** (多行的) 当使用 `^` 以及 `$` 的时候将会匹配行首和行尾而不是整个字符串
- **i** (大小写不敏感的) 让整个表达式大小写不敏感（比如 `/aBc/i` 将匹配 `Abc`）

分组以及捕获 — ()

a(bc) 括号产生一个值为 bc 的捕获分组-> 试一下!

a(?:bc)* 我么可以使用 ?: 让捕获分组不起作用->试一下!

a(?bc) 我们可以使用 ? 将名字放在分组中 -> 试一下!

方括号表达式 — []

[abc] 匹配一个具有 要么一个 a 或者一个 b 或者一个 c 的字符串-> 等同于 `a|b|c` -> 试一下!

[a-c] 与前一条相同

[a-fA-F0-9] 字符串代表一个十六进制数，大小写不敏感-> 试一下!

[0-9]% 一个具有从0到9其后后一个 % 符号

[^a-zA-Z] 一个不是大小写字母的字符串。在这种情况下，^ 被用为 表达式的否定。->试一下!

边界 — \b 以及 \B

\babc\b 执行“仅限整个单词”搜索->试一下!

返回引用— \1

([abc])\1 使用 `\1` 将会匹配与第一个捕获分组相同的文本 -> 试一下!

([abc])([de])\2\1 我们可以使用 \2 (\3, \4, 等等)来获取被第二个(第三个, 第四个, 等等.)捕获分组相同的文本 -> 试一下!

(?[abc])\k 我们将分组名称命名为 `foo` 并随后使用 `(\k<foo>)` 来进行引用。结果与第一个正则表达式相同 -> 试一下!

前瞻和后瞻 — (?=) 以及 (?<=)

d(?=r) 匹配一个 `d` 并且其后有一个 `r`, 但是 `r` 将不会是整个正则表达式匹配的一部分-> 试一下!

(?<=r)d 匹配一个 `d` 并且前面有一个 `r`, 但是 `r` 将不会是整个正则表达式匹配的一部分-> 试一下!

我们也可以使用否定符号 !

d(?!r) 匹配一个 `d` 并且其后不是一个 `r`, 但是 `r` 将不会是整个正则表达式匹配的一部分-> 试一下!

(?<!r)d 匹配一个 `d` 并且前面不是一个 `r`, 但是 `r` 将不会是整个正则表达式匹配的一部分-> 试一下!