

正则表达式快速参考

模式、量词、分组、前瞻断言与标志

基础模式

元字符

- `.` 任意字符 (换行除外)
- `^` 字符串/行的起始位置
- `$` 字符串/行的结束位置
- `*` 0个或多个前置字符
- `+` 1个或多个前置字符
- `?` 0个或1个前置字符 (可选)
- `\` 转义元字符

字面匹配

```
hello # matches "hello" exactly
a.c # matches "abc", "alc", "a-c", etc.
.txt # matches literal ".txt"
```

字符类

方括号表达式

- `[abc]` 匹配 a、b 或 c
- `[^abc]` 匹配除 a、b、c 以外的任意字符
- `[a-z]` 小写字母
- `[A-Z]` 大写字母
- `[0-9]` 数字
- `[a-zA-Z0-9]` 字母数字

简写字符类

- `\d` 数字 `[0-9]`
- `\D` 非数字 `[^0-9]`
- `\w` 单词字符 `[a-zA-Z0-9_]`
- `\W` 非单词字符
- `\s` 空白字符 `[\t\n\r\f]`
- `\S` 非空白字符

量词

贪婪量词

- `*` 0个或多个 (贪婪)
- `+` 1个或多个 (贪婪)
- `?` 0个或1个 (贪婪)
- `{n}` 恰好 n 次
- `{n,}` n 次或更多
- `{n,m}` n 到 m 次

懒惰量词

- `*?` 0个或多个 (懒惰/非贪婪)
 - `+?` 1个或多个 (懒惰)
 - `??` 0个或1个 (懒惰)
 - `{n,m}?` n 到 m 次 (懒惰)
- 懒惰量词尽可能少地匹配字符

贪婪 vs 懒惰

```
<.+> # greedy: "<b>bold</b>"
<.+?> # lazy: " <b>"
```

锚点

- `^` 字符串起始 (启用 `m` 标志时匹配行首)
- `$` 字符串结束 (启用 `m` 标志时匹配行尾)
- `\b` 单词边界
- `\B` 非单词边界
- `\A` 字符串起始 (不受 `m` 影响)
- `\Z` 字符串结束 (不受 `m` 影响)

锚点示例

```
"Hello" # starts with "Hello"
worlds # ends with "world"
\bword\b # "word" as whole word
\bword\B # "word" inside another word
```

分组与交替

捕获组

```
(abc) # capture group: match "abc"
(a|b|c) # alternation: a or b or c
(cat|dog) # match "cat" or "dog"
\d{3}-\d{4} # groups: "123-4567"
```

分组类型

- `(pattern)` 捕获组
- `(?:pattern)` 非捕获组
- `(?P<name>pat)` 命名组 (Python)
- `(?<name>pat)` 命名组 (JS、.NET)
- `\1 \2` 反向引用第 1、2 组
- `a|b` 交替: 匹配 a 或 b

前瞻与后瞻断言

- `(?=pattern)` 正向前瞻
- `(?!pattern)` 负向前瞻
- `(?<=pattern)` 正向后瞻
- `(?<!pattern)` 负向后瞻

断言示例

```
\d+(?= USD) # digits followed by " USD"
\d+(?! USD) # digits NOT followed by " USD"
(?<=s)\d+ # digits preceded by "s"
(?<!s)\d+ # digits NOT preceded by "s"
```

断言匹配的是位置, 不消耗字符

常用模式

- `\d{1,3}(\.\d{1,3}){3}` IPv4 地址 (基础)
- `[lw.+-]+@[lw.-]+\.[lw.]+` Email (基础)
- `https?://[w.-/\-?&#]=1+` URL (基础)
- `\(? \d{3}?\)?[-. \s]? \d{3}[-. \s]? \d{4}` 美国电话号码
- `\d{4}-\d{2}-\d{2}` 日期 (YYYY-MM-DD)
- `#?[0-9a-fA-F]{6}` 十六进制颜色码

以上为简化模式, 生产环境可能需要更严格的验证

标志

- `g` 全局: 查找所有匹配, 而非只找第一个
- `i` 忽略大小写
- `m` 多行: `^`、`$` 匹配行边界
- `s` 单行模式: `.` 也匹配换行符
- `x` 扩展模式: 忽略空白, 允许注释
- `u` Unicode: 完整的 Unicode 支持

各语言的标志用法

```
/pattern/gi # JavaScript
re.compile(r'pat', re.I | re.M) # Python
grep -iE "pattern" # grep (extended)
```