正規表現 クイックリファレンス

パターン、量指定子、グループ、先読み、フラグ

基本パターン

メタ文字

`.`	任意の 1 文字（改行を除く）
`^`	文字列/行の先頭
`$`	文字列/行の末尾
`*`	直前の 0 回以上
`+`	直前の 1 回以上
`?`	直前の 0 または 1 回（省略可能）
`\`	メタ文字のエスケープ

リテラルマッチング

```
hello     # matches "hello" exactly
a.c       # matches "abc", "a1c", "a-c", etc.
\.txt     # matches literal ".txt"
```

文字クラス

ブラケット表現

`[abc]`	a、b、c にマッチ
`[^abc]`	a、b、c 以外にマッチ
`[a-z]`	小文字アルファベット
`[A-Z]`	大文字アルファベット
`[0-9]`	数字
`[a-zA-Z0-9]`	英数字

省略クラス

`\d`	数字 `[0-9]`
`\D`	非数字 `[^0-9]`
`\w`	単語文字 `[a-zA-Z0-9_]`
`\W`	非単語文字
`\s`	空白 `[ \t\n\r\f]`
`\S`	非空白

量指定子

貪欲な量指定子

`*`	0 回以上（貪欲）
`+`	1 回以上（貪欲）
`?`	0 または 1 回（貪欲）
`{n}`	ちょうど n 回
`{n,}`	n 回以上
`{n,m}`	n 回以上 m 回以下

怠惰な量指定子

`*?`	0 回以上（怠惰/非貪欲）
`+?`	1 回以上（怠惰）
`??`	0 または 1 回（怠惰）
`{n,m}?`	n 回以上 m 回以下（怠惰）

怠惰な量指定子はできるだけ少ない文字にマッチする

貪欲 vs 怠惰

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

アンカー

`^`	文字列の先頭（`m`フラグで行の先頭）
`$`	文字列の末尾（`m`フラグで行の末尾）
`\b`	単語境界
`\B`	非単語境界
`\A`	文字列の先頭（`m`の影響を受けない）
`\Z`	文字列の末尾（`m`の影響を受けない）

アンカーの例

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

グループと選択

キャプチャグループ

```
(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"
```

グループの種類

`(pattern)`	キャプチャグループ
`(?:pattern)`	非キャプチャグループ
`(?P<name>pat)`	名前付きグループ（Python）
`(?<name>pat)`	名前付きグループ（JS、.NET）
`\1` `\2`	グループ 1、2 への後方参照
`a|b`	選択：a または b

先読みと後読み

`(?=pattern)`	肯定先読み
`(?!pattern)`	否定先読み
`(?<=pattern)`	肯定後読み
`(?<!pattern)`	否定後読み

先読み/後読みの例

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

先読み/後読みは文字を消費せずに位置にマッチする

よくあるパターン

`\d{1,3}(\.\d{1,3}){3}`	IPv4 アドレス（基本）
`[\w.+-]+@[\w-]+\.[\w.]+`	メールアドレス（基本）
`https?://[\w./\-?&#=]+`	URL（基本）
`\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}`	米国の電話番号
`\d{4}-\d{2}-\d{2}`	日付（YYYY-MM-DD）
`#?[0-9a-fA-F]{6}`	16 進カラーコード

これらは簡略化されたパターン。本番環境ではより厳密なバリデーションが必要な場合がある

フラグ

`g`	グローバル：最初だけでなく全マッチを検索
`i`	大文字小文字を区別しないマッチング
`m`	複数行：`^`/`$`が行境界にマッチ
`s`	ドットオール：`.`が改行にもマッチ
`x`	詳細モード：空白を無視、コメントを許可
`u`	Unicode：完全な Unicode サポート

言語別フラグの使用

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