REGULAR EXPRESSIONS QUICK REFERENCE

Patterns, quantifiers, groups, lookaheads, flags

Basic Patterns

Metacharacters

`.`	Any character (except newline)
`^`	Start of string / line
`$`	End of string / line
`*`	0 or more of previous
`+`	1 or more of previous
`?`	0 or 1 of previous (optional)
`\`	Escape metacharacter

Literal Matching

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

Character Classes

Bracket Expressions

`[abc]`	Match a, b, or c
`[^abc]`	Match anything except a, b, c
`[a-z]`	Lowercase letter
`[A-Z]`	Uppercase letter
`[0-9]`	Digit
`[a-zA-Z0-9]`	Alphanumeric

Shorthand Classes

`\d`	Digit `[0-9]`
`\D`	Non-digit `[^0-9]`
`\w`	Word char `[a-zA-Z0-9_]`
`\W`	Non-word char
`\s`	Whitespace `[ \t\n\r\f]`
`\S`	Non-whitespace

Quantifiers

Greedy Quantifiers

`*`	0 or more (greedy)
`+`	1 or more (greedy)
`?`	0 or 1 (greedy)
`{n}`	Exactly n times
`{n,}`	n or more times
`{n,m}`	Between n and m times

Lazy Quantifiers

`*?`	0 or more (lazy / non-greedy)
`+?`	1 or more (lazy)
`??`	0 or 1 (lazy)
`{n,m}?`	Between n and m (lazy)

Lazy quantifiers match as few characters as possible

Greedy vs Lazy

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

Anchors

`^`	Start of string (or line with `m` flag)
`$`	End of string (or line with `m` flag)
`\b`	Word boundary
`\B`	Non-word boundary
`\A`	Start of string (not affected by `m`)
`\Z`	End of string (not affected by `m`)

Anchor Examples

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

Groups & Alternation

Capturing Groups

```
(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"
```

Group Types

`(pattern)`	Capturing group
`(?:pattern)`	Non-capturing group
`(?P<name>pat)`	Named group (Python)
`(?<name>pat)`	Named group (JS, .NET)
`\1` `\2`	Backreference to group 1, 2
`a|b`	Alternation: a or b

Lookahead & Lookbehind

`(?=pattern)`	Positive lookahead
`(?!pattern)`	Negative lookahead
`(?<=pattern)`	Positive lookbehind
`(?<!pattern)`	Negative lookbehind

Lookaround Examples

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

Lookarounds match a position without consuming characters

Common Patterns

`\d{1,3}(\.\d{1,3}){3}`	IPv4 address (basic)
`[\w.+-]+@[\w-]+\.[\w.]+`	Email (basic)
`https?://[\w./\-?&#=]+`	URL (basic)
`\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}`	US phone number
`\d{4}-\d{2}-\d{2}`	Date (YYYY-MM-DD)
`#?[0-9a-fA-F]{6}`	Hex color code

These are simplified patterns; production use may need stricter validation

Flags

`g`	Global: find all matches, not just first
`i`	Case-insensitive matching
`m`	Multiline: `^` / `$` match line boundaries
`s`	Dotall: `.` matches newline too
`x`	Verbose: ignore whitespace, allow comments
`u`	Unicode: full Unicode support

Flag Usage by Language

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