Regular Expressions Cheat Sheet

A concise reference for regular expressions, covering syntax, metacharacters, common patterns, and usage examples for efficient text processing.

Regex Basics and Metacharacters

Core Metacharacters

`\`	Escapes a special character (e.g., `\.` matches a literal dot).
`.`	Matches any single character except newline.
`^`	Matches the start of the string or line (depending on multiline mode).
`$`	Matches the end of the string or line (depending on multiline mode).
`|`	Acts as an 'or' operator (e.g., `a|b` matches 'a' or 'b').
`[]`	Defines a character class (e.g., `[abc]` matches 'a', 'b', or 'c').

Quantifiers

`*`	Matches the preceding character zero or more times.
`+`	Matches the preceding character one or more times.
`?`	Matches the preceding character zero or one time (optional).
`{n}`	Matches the preceding character exactly n times.
`{n,}`	Matches the preceding character n or more times.
`{n,m}`	Matches the preceding character between n and m times (inclusive).

Character Classes

`\d`	Matches any digit (0-9).
`\D`	Matches any non-digit character.
`\w`	Matches any word character (a-z, A-Z, 0-9, and _).
`\W`	Matches any non-word character.
`\s`	Matches any whitespace character (space, tab, newline).
`\S`	Matches any non-whitespace character.

Anchors and Grouping

Anchors

`^`	Matches the beginning of the string. Inside a character class, it negates the class (e.g., `[^abc]` matches any character except a, b, or c).
`$`	Matches the end of the string.
`\b`	Matches a word boundary (the position between a word character and a non-word character).
`\B`	Matches a non-word boundary.

Grouping and Capturing

`()`	Groups parts of a regex together. Captures the matched group for backreferencing.
`(?:)`	Creates a non-capturing group. Useful for grouping without capturing the matched text.
`\1`, `\2`, etc.	Backreferences to the first, second, etc., captured groups in the regex.

Flags/Modifiers

`i`	Case-insensitive matching.
`g`	Global matching (find all matches rather than stopping after the first).
`m`	Multiline mode: `^` and `$` match the start and end of each line.
`s`	Dotall mode: `.` matches any character, including newline.

Lookarounds and Common Patterns

Lookarounds

`(?=pattern)`	Positive lookahead: Matches if `pattern` follows the current position, but doesn't include it in the match.
`(?!pattern)`	Negative lookahead: Matches if `pattern` does *not* follow the current position.
`(?<=pattern)`	Positive lookbehind: Matches if `pattern` precedes the current position, but doesn't include it in the match. (Not supported in all regex engines.)
`(?<!pattern)`	Negative lookbehind: Matches if `pattern` does *not* precede the current position. (Not supported in all regex engines.)

Common Patterns

Email Address: `[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}`

URL: `https?:\/\/(www\.)?[-a-zA-Z0-9@:%._\+~#?&//=]{2,256}\.[a-z]{2,4}\b(\/[-a-zA-Z0-9@:%._\+~#?&//=]*)?`

IP Address: `((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)`

Date (YYYY-MM-DD): `\d{4}-\d{2}-\d{2}`

Phone Number (US): `\d{3}-\d{3}-\d{4}`

POSIX Character Classes

POSIX Character Classes

`[[:alnum:]]`	Alphanumeric characters (a-z, A-Z, 0-9).
`[[:alpha:]]`	Alphabetic characters (a-z, A-Z).
`[[:blank:]]`	Space and tab characters.
`[[:cntrl:]]`	Control characters.
`[[:digit:]]`	Numeric characters (0-9); equivalent to `\d`.
`[[:graph:]]`	Visible characters (excluding spaces, control characters).

`[[:lower:]]`	Lowercase characters (a-z).
`[[:print:]]`	Printable characters (including spaces).
`[[:punct:]]`	Punctuation characters.
`[[:space:]]`	Whitespace characters (space, tab, newline, etc.); equivalent to `\s`.
`[[:upper:]]`	Uppercase characters (A-Z).
`[[:xdigit:]]`	Hexadecimal digits (0-9, a-f, A-F).