Regular Expressions Cheat Sheet

A quick reference guide to regular expressions, covering syntax, metacharacters, common patterns, and usage examples for text manipulation.

Regex Fundamentals

Basic Metacharacters

`.` (Dot)	Matches any single character except newline. **Example:** `a.c` matches "abc", "adc", "a1c", etc.
`^` (Caret)	Matches the beginning of the string. **Example:** `^abc` matches "abc" only if it's at the start of the string.
`$` (Dollar)	Matches the end of the string. **Example:** `xyz$` matches "xyz" only if it's at the end of the string.
`[]` (Square brackets)	Defines a character class, matching any character within the brackets. **Example:** `[aeiou]` matches any vowel.
`[^]` (Negated square brackets)	Matches any character *not* within the brackets. **Example:** `[^0-9]` matches any non-digit character.
`|` (Pipe)	Acts as an "OR" operator, matching either the expression before or after the pipe. **Example:** `cat|dog` matches either "cat" or "dog".
`()` (Parentheses)	Groups parts of the regex together and captures the matched substring. **Example:** `(abc)+` matches one or more occurrences of "abc" and captures "abc".

Quantifiers

`*` (Asterisk)	Matches the preceding character zero or more times. **Example:** `ab*c` matches "ac", "abc", "abbc", "abbbc", etc.
`+` (Plus)	Matches the preceding character one or more times. **Example:** `ab+c` matches "abc", "abbc", "abbbc", etc., but *not* "ac".
`?` (Question mark)	Matches the preceding character zero or one time. **Example:** `ab?c` matches "ac" or "abc".
`{n}`	Matches the preceding character exactly `n` times. **Example:** `a{3}` matches "aaa".
`{n,}`	Matches the preceding character `n` or more times. **Example:** `a{2,}` matches "aa", "aaa", "aaaa", etc.
`{n,m}`	Matches the preceding character between `n` and `m` times (inclusive). **Example:** `a{2,4}` matches "aa", "aaa", or "aaaa".

Character Classes & Anchors

Predefined Character Classes

`\d`	Matches any digit (0-9). Equivalent to `[0-9]`.
`\D`	Matches any non-digit character. Equivalent to `[^0-9]`.
`\w`	Matches any word character (alphanumeric and underscore). Equivalent to `[a-zA-Z0-9_]`.
`\W`	Matches any non-word character. Equivalent to `[^a-zA-Z0-9_]`.
`\s`	Matches any whitespace character (space, tab, newline, etc.).
`\S`	Matches any non-whitespace character.

Anchors

`^`	Matches the beginning of a line. **Example:** `^Hello` matches "Hello world" but not "World Hello".
`$`	Matches the end of a line. **Example:** `world$` matches "Hello world" but not "world Hello".
`\b`	Matches a word boundary (the position between a word character and a non-word character). **Example:** `\bcat\b` matches "cat" in "The cat sat" but not in "cattle".
`\B`	Matches a non-word boundary. **Example:** `\Bcat\B` matches "cat" in "cattle" but not in "The cat sat".

Groups and Lookarounds

Capturing Groups

(...)	Captures the matched portion of the string. The captured group can be referenced later (e.g., in backreferences or when extracting matches). **Example:** `(abc)` captures the substring “abc”.
`\1`, `\2`, etc.	Backreferences to previously captured groups. `\1` refers to the first captured group, `\2` to the second, and so on. **Example:** `(.)(.)\2\1` matches palindromes like “abba”.

Non-Capturing Groups

(?:...)	Groups the pattern but does *not* capture the matched substring. Useful for grouping without the overhead of capturing. **Example:** `(?:abc)+` matches one or more occurrences of “abc” but does not capture it.

Lookarounds (Zero-Width Assertions)

(?=...)	Positive lookahead. Asserts that the pattern is followed by the given subpattern, without consuming the subpattern. **Example:** `\w+(?=\d)` matches a word followed by a digit, but the digit is not included in the match.
(?!...)	Negative lookahead. Asserts that the pattern is *not* followed by the given subpattern. **Example:** `\w+(?!\d)` matches a word not followed by a digit.
(?<=...)	Positive lookbehind. Asserts that the pattern is preceded by the given subpattern, without consuming the subpattern. *Note: Not supported in all regex engines, and lookbehind assertions often have length restrictions.* **Example:** `(?<=\$)(\d+)` matches digits preceded by a dollar sign, but the dollar sign is not included in the match.
(?<!...)	Negative lookbehind. Asserts that the pattern is *not* preceded by the given subpattern. *Note: Not supported in all regex engines, and lookbehind assertions often have length restrictions.* **Example:** `(?<!\$)(\d+)` matches digits not preceded by a dollar sign.

Flags & Common Patterns

Common Flags (Modifiers)

`i`	Case-insensitive matching. **Example:** `/abc/i` matches “abc”, “Abc”, “ABC”, etc.
`g`	Global matching. Finds all matches rather than stopping after the first. **Example:** `/abc/g` finds all occurrences of “abc” in the string.
`m`	Multiline mode. `^` and `$` match the beginning and end of each line (separated by newlines) rather than the beginning and end of the entire string. **Example:** `/^abc$/m` matches “abc” at the beginning of a line.
`s`	Dotall mode. The `.` metacharacter matches any character, *including* newline characters. Without this flag, `.` matches any character *except* newline. **Example:** `/a.b/s` matches “a\nb”.
`x`	Verbose mode. Allows whitespace and comments within the regex pattern (for better readability). Whitespace is ignored unless escaped or within a character class. Comments start with `#`. **Example:** `/abc # Matches abc` `  def/x` ```` ``` matches `abcdef`. ````

Common Regex Patterns

Matching an email address:
`[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}`

Matching a URL:
`(https?:\/\/(?:www\.|(?!www))[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-9]\.[^\s]{2,}|www\.[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-9]\.[^\s]{2,}|https?:\/\/(?:www\.|(?!www))[a-zA-Z0-9]+\.[^\s]{2,}|www\.[a-zA-Z0-9]+\.[^\s]{2,})`

Matching an 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]?)`

Matching a date (YYYY-MM-DD):
`\d{4}-\d{2}-\d{2}`

Matching a US phone number:
`\d{3}-\d{3}-\d{4}` or `(\(\d{3}\))?\s*\d{3}-\d{4}`

Matching HTML tags:
`<[^>]+>`