Advanced Regular Expressions Cheat Sheet

A concise guide to advanced regular expression patterns and techniques, including lookarounds, backreferences, and conditional matching, designed to help you master complex text manipulation.

Lookarounds

Positive Lookahead

`(?:pattern)`	Matches a group without capturing it. Useful when you need to group parts of a regex but don't need to refer back to them. **Example:** `(?:https?|ftp)://.*` Matches a URL but doesn't capture the protocol.
`(?=pattern)`	Asserts that the regex matches the `pattern` that follows, but doesn't include the `pattern` in the match. **Example:** `\w+(?=\sInc\.)` Matches a word followed by ' Inc.', without including ' Inc.' in the matched text.
`X(?=Y)`	Find "X" only if followed by "Y".
Example	`foo(?=bar)` Matches 'foo' only if it's followed by 'bar', but 'bar' is not part of the match.
Use cases	Validating password strength, parsing structured data, and conditional replacements.
Real-world example	Extract the version number from 'app-1.2.3.zip' using `app-(?=\d+(?:\.\d+)*\.zip)`. This will only match 'app-' if it's followed by a version number pattern and '.zip'.

Negative Lookahead

`X(?!Y)`	Find "X" only if not followed by "Y".
Example	`foo(?!bar)` Matches 'foo' only if it's NOT followed by 'bar'.
`(?<!pattern)`	Asserts that the regex matches if the `pattern` does not precede the current position. The `pattern` is not included in the match. **Example:** `(?<!\d)%\w+` Matches '%word' only if it is not preceded by a digit.
Use cases	Filtering log files, validating data formats, and advanced search functionalities.
Real-world example	Find all words that are not preceded by a number using `(?<!\d)\b\w+\b`. This helps to exclude words that are part of a numbered list.
`(?<!pattern)X`	Asserts that the regex matches if the `pattern` does not precede the current position. The `pattern` is not included in the match. **Example:** `(?<![A-Z])\d+` Matches a one or more digits if not preceded by a capital letter

Positive Lookbehind

`(?<=pattern)`	Asserts that the regex matches the `pattern` that precedes, but doesn't include the `pattern` in the match. **Example:** `(?<=USD)\d+\.?\d*` Matches a number preceded by 'USD', without including 'USD' in the matched number.
`(?<=X)Y`	Find "Y" only if preceded by "X".
Example	`(?<=bar)foo` Matches 'foo' only if it's preceded by 'bar', but 'bar' is not part of the match.
Use cases	Extracting data from specific contexts, validating formatted input, and data sanitization.
Real-world example	Extract file sizes (numbers) only when they are indicated in kilobytes (KB) using `(?<=KB )\d+`. This targets only the file sizes specified in KB.
Note	Not supported in all regex engines.

Negative Lookbehind

`(?<!pattern)X`	Asserts that the regex matches if the `pattern` does not precede the current position. The `pattern` is not included in the match. **Example:** `(?<!\d)%\w+` Matches '%word' only if it is not preceded by a digit.
`(?<!X)Y`	Find "Y" only if not preceded by "X".
Example	`(?<!bar)foo` Matches 'foo' only if it's NOT preceded by 'bar'.
Use cases	Filtering data based on context, excluding unwanted patterns, and refining search results.
Real-world example	Find function names that are not part of a class method definition using `(?<!\.)\b\w+\b`. This helps to identify standalone functions.
Note	Not supported in all regex engines.

Backreferences

Basic Backreference

`\1`, `\2`, etc.	Refers to the text matched by the 1st, 2nd, etc. capturing group. **Example:** `(\w+)\s\1` Matches a repeated word, like 'the the'.
Use cases	Finding duplicate words, validating symmetrical patterns, and complex text replacements.
Example	Find duplicated words in a text: `(\b\w+)\s+\1`. This will match 'word word' and is case-sensitive.
Common mistake	Forgetting that backreferences refer to the exact matched text, not the pattern.
Real-world example	Correct HTML tag pairing using `<(.*?)>.*?</\1>`. This ensures that the closing tag matches the opening tag (e.g., `<h1>...</h1>`).
Note	Backreferences can significantly increase the complexity (and processing time) of regex matching.

Named Capture Groups

`(?<name>pattern)` (PCRE/Python)	Defines a named capture group. **Example:** `(?<year>\d{4})-(?<month>\d{2})-(?<day>\d{2})` Matches a date and names the groups 'year', 'month', and 'day'.
`(?'name'pattern)` (.NET)	Alternative syntax for defining named capture groups in .NET.
`\k<name>` (PCRE/Python)	Refers to a named capture group. **Example:** `(?<word>\w+)\s+\k<word>` Matches repeated words using the named group 'word'.
Use cases	Parsing complex data structures, extracting specific parts of a string, and making regexes more readable.
Real-world example	Extract specific parts of a log entry like timestamp, log level, and message using named groups for better clarity and maintainability.
Note	Named groups improve readability but might not be supported in all regex engines.

Backreference in Replacement

`$1`, `$2`, etc. (Most engines)	Refers to captured groups in the replacement string.
	Example:
	`Find: (\w+),(\s)(\w+)`
	`Replace: $3,$2$1`
	Swaps the first and last word separated by a comma and space.

`\1`, `\2`, etc. (Some engines)	Alternative syntax for backreferences in replacement strings, especially in languages like Python.
Use cases	Reformatting data, swapping fields, and complex string manipulations.
Example	Reformat phone numbers from '123-456-7890' to '(123) 456-7890' using `(\d{3})-(\d{3})-(\d{4})` as the find pattern and `($1) $2-$3` as the replace pattern.
Note	Ensure that the backreference number matches the intended capture group to avoid unexpected results.
Real-world example	Swap first name and last name in a CSV file, where names are separated by a comma, using backreferences in the replacement string.

Conditional Matching

If-Then-Else Conditionals

`?(?(condition)then|else)`	Matches either the `then` pattern if the `condition` is true, or the `else` pattern if the `condition` is false.
Condition syntax	(?(1)then|else) - Condition based on whether group 1 matched.
Example	`(<)?(\w+@\w+(?:\.\w+)+)(?(1)>)` Matches email addresses, optionally enclosed in angle brackets.
Use cases	Handling optional elements, validating complex data formats, and adapting matching based on context.
Real-world example	Parse data entries where some fields are optional but depend on the presence of others, such as address fields in a contact database.
Note	Not supported in all regex engines, and syntax may vary.

If-Then Conditionals

`?(?(condition)then)`	Matches the `then` pattern only if the `condition` is true.
Condition syntax	(?(name)then) - Condition based on whether named group 'name' matched.
Example	`(\()?\d+(?(1)\))` Matches a number, optionally enclosed in parentheses, but only if both parentheses are present.
Use cases	Validating paired elements, handling different formats, and ensuring data consistency.
Real-world example	Process log entries that may or may not include a timestamp, but require specific handling if the timestamp is present.
Note	Like If-Then-Else, If-Then conditionals have limited support across regex engines.

Recursion

Recursive Patterns

`(?R)` or `(?0)`	Recurses the entire regular expression. **Example:** `\(([^()]|(?R))*\)` Matches nested parentheses.
`(?n)`	Recurses the nth subpattern.
Use cases	Matching nested structures, parsing markup languages, and validating complex syntax.
Note	Recursion is powerful but can lead to performance issues or stack overflow errors with deeply nested structures. Not supported in all regex engines.

Example	Match nested HTML tags like `<div><div>...</div></div>` using recursion to ensure proper nesting.
Real-world example	Parse nested JSON or XML structures, ensuring that all opening tags have corresponding closing tags.