Regular Expression Reference: Special Groups

JGsoft | .NET

Feature	Syntax	Description
Comment	`(?#comment)`	Everything between `(?#` and `)` is ignored by the regex engine.
Branch reset group	`(?|regex)`	If the regex inside the branch reset group has multiple alternatives with capturin numbers are the same in all the alternatives.
Atomic group	`(?>regex)`	Atomic groups prevent the regex engine from backtracking back into the group group. If the remainder of the regex fails, the engine may backtrack over the gro optional. But it will not backtrack into the group to try other permutations of the
Positive lookahead	`(?=regex)`	Matches at a position where the pattern inside the lookahead can be matched. consume any characters or expand the match. In a pattern like `one(?=two)th` match at the position where the match of `one` ends.
Negative lookahead	`(?!regex)`	Similar to positive lookahead, except that negative lookahead only succeeds if match.
Positive lookbehind	`(?<=regex)`	Matches at a position if the pattern inside the lookbehind can be matched endin
Negative lookbehind	`(?<!regex)`	Matches at a position if the pattern inside the lookbehind cannot be matched en
Lookbehind	`(?<=regex|longer regex)`	Alternatives inside lookbehind can differ in length.
Lookbehind	`(?<=x{n,m})`	Quantifiers with a finite maximum number of repetitions can be used inside look
Lookbehind	`(?<=regex)`	The full regular expression syntax can be used inside lookbehind.
Lookbehind	`(group)(?<=\1)`	Backreferences can be used inside lookbehind. Syntax prohibited in lookbehind capturing group.
Keep text out of the regex match	`\K`	The text matched by the part of the regex to the left of the `\K` is omitted from th the regex is matched normally from left to right. Capturing groups to the left of t
Lookaround conditional	`(?(?=regex)then|else)` where `(?=regex)` is any valid lookaround and `then` and `else` are any valid regexes	If the lookaround succeeds, the "then" part must match for the overall regex to part must match for the overall regex to match. The lookaround is zero-length. their matches like normal regexes.
Implicit lookahead conditional	`(?(regex)then|else)` where `regex`, `then`, and `else` are any valid regexes and `regex` is not the name of a capturing group	If "regex" is not the name of a capturing group, then it is interpreted as a lookah `(?(?=regex)then|else)`. If the lookahead succeeds, the "then" part must m the lookahead fails, the "else" part must match for the overall regex to match. T "then" and "else" parts consume their matches like normal regexes.
Named conditional	`(?(name)then|else)` where `name` is the name of a capturing group and `then` and `else` are any valid regexes	If the capturing group with the given name took part in the match attempt thus f overall regex to match. If the capturing group did not take part in the match thus overall regex to match.
Named conditional	`(?(<name>)then|else)` where `name` is the name of a capturing group and `then` and `else` are any valid regexes	If the capturing group with the given name took part in the match attempt thus f overall regex to match. If the capturing group did not take part in the match thus overall regex to match.
Named conditional	`(?('name')then|else)` where `name` is the name of a capturing group and `then` and `else` are any valid regexes	If the capturing group with the given name took part in the match attempt thus f overall regex to match. If the capturing group did not take part in the match thus overall regex to match.
Conditional	`(?(1)then|else)` where 1 is	If the referenced capturing group took part in the match attempt thus far, the "th

	the number of a capturing group and `then` and `else` are any valid regexes	regex to match. If the capturing group did not take part in the match thus far, the regex to match.
Relative conditional	`(?(-1)then|else)` where `-1` is a negative integer and `then` and `else` are any valid regexes	Conditional that tests the capturing group that can be found by counting as man numbered capturing groups as specified by the number from right to left starting the referenced capturing group took part in the match attempt thus far, the "ther to match. If the capturing group did not take part in the match thus far, the "else to match.
Forward conditional	`(?(+1)then|else)` where `+1` is a positive integer and `then` and `else` are any valid regexes	Conditional that tests the capturing group that can be found by counting as man numbered capturing groups as specified by the number from left to right starting referenced capturing group took part in the match attempt thus far, the "then" pa match. If the capturing group did not take part in the match thus far, the "else" p match.
Conditional	`(?(+1)then|else)` where `1` is the number of a capturing group and `then` and `else` are any valid regexes	The + is ignored and the number is taken as an absolute reference to a capturir group took part in the match attempt thus far, the "then" part must match for the group did not take part in the match thus far, the "else" part must match for the
Feature	**Syntax**	**Description**

If you can imagine it, we will build you there.

Make a Donation

Did this website just save you a trip to the bookstore? Please make a donation to support this site, and you'll get a **lifetime of advertisement-free access** to this site!

Page URL: https://www.regular-expressions.info/refadv.html
Page last updated: 10 September 2018
Site last updated: 03 April 2019