Specifying Modes Inside The Regular Expression

Normally, matching modes are specified outside the regular expression. In a programming language, you pass them as a flag to the regex constructor or append them to the regex literal. In an application, you'd toggle the appropriate buttons or checkboxes. You can find the specifics in the tools and languages section of this website.

Sometimes, the tool or language does not provide the ability to specify matching options. The handy `String.matches()` method in Java does not take a parameter for matching options like `Pattern.compile()` does. Or, the regex flavor may support matching modes that aren't exposed as external flags. The regex functions in R have `ignore.case` as their only option, even though the underlying PCRE library has more matching modes than any other discussed in this tutorial.

In those situation, you can add the following mode modifiers to the start of the regex. To specify multiple modes, simply put them together as in `(?ismx)`.

- `(?i)` makes the regex case insensitive.
- `(?c)` makes the regex case sensitive. Only supported by Tcl.
- `(?x)` turn on free-spacing mode.
- `(?t)` turn off free-spacing mode. Only supported by Tcl.
- `(?xx)` turn on free-spacing mode, also in character classes. Supported by Perl 5.26 and PCRE2 10.30.
- `(?s)` for "single line mode" makes the dot match all characters, including line breaks. Not supported by Ruby or JavaScript. In Tcl, `(?s)` also makes the caret and dollar match at the start and end of the string only.
- `(?m)` for "multi-line mode" makes the caret and dollar match at the start and end of each line in the subject string. In Ruby, `(?m)` makes the dot match all characters, without affecting the caret and dollar which always match at the start and end of each line in Ruby. In Tcl, `(?m)` also prevents the dot from matching line breaks.
- `(?p)` in Tcl makes the caret and dollar match at the start and the end of each line, and makes the dot match line breaks.
- `(?w)` in Tcl makes the caret and dollar match only at the start and the end of the subject string, and prevents the dot from matching line breaks.
- `(?n)` turns all unnamed groups into non-capturing groups. Only supported by .NET, XRegExp, and the JGsoft flavor. In Tcl, `(?n)` is the same as `(?m)`.
- `(?J)` allows duplicate group names. Only supported by PCRE and languages that use it such as Delphi, PHP and R.
- `(?U)` turns on "ungreedy mode", which switches the syntax for greedy and lazy quantifiers. So `(?U)a*` is lazy and `(?U)a*?` is greedy. Only supported by PCRE and languages that use it. It's use is strongly discouraged because it confuses the meaning of the standard quantifier syntax.
- `(?d)` corresponds with UNIX_LINES in Java, which makes the dot, caret, and dollar treat only the newline character `\n` as a line break, instead of recognizing all line break characters from the Unicode standard. Whether they match or don't match (at) line breaks depends on `(?s)` and `(?m)`.
- `(?b)` makes Tcl interpret the regex as a POSIX BRE.
- `(?e)` makes Tcl interpret the regex as a POSIX ERE.
- `(?q)` makes Tcl interpret the regex as a literal string (minus the `(?q)` characters).
- `(?X)` makes escaping letters with a backslash an error if that combination is not a valid regex token. Only supported by PCRE and languages that use it.

Turning Modes On and Off for Only Part of The Regular Expression

Modern regex flavors allow you to apply modifiers to only part of the regular expression. If you insert the modifier `(?ism)` in the middle of the regex then the modifier only applies to the part of the regex to the right of the modifier.

With these flavors, you can turn off modes by preceding them with a minus sign. All modes after the minus sign will be turned off. E.g. `(?i-sm)` turns on case insensitivity, and turns off both single-line mode and multi-line mode.

Flavors that can't apply modifiers to only part of the regex treat a modifiers in the middle of the regex as an error. Python is an exception to this. In Python, putting a modifier in the middle of the regex affects the whole regex. So in Python, `(?i)caseless` and `caseless(?i)` are both case insensitive. In all other flavors, the trailing mode modifier either has no effect or is an error.

You can quickly test how the regex flavor you're using handles mode modifiers. The regex `(?i)te(?-i)st` should match `test` and `TEst`, but not `teST` or `TEST`.

Modifier Spans

Instead of using two modifiers, one to turn an option on, and one to turn it off, you use a modifier span. `(?i)caseless(?-i)cased(?i)caseless` is equivalent to `(?i)caseless(?-i:cased)caseless`. This syntax resembles that of the non-capturing group `(?:group)`. You could think of a non-capturing group as a modifier span that does not change any modifiers. But there are flavors, like JavaScript, Python, and Tcl that support non-capturing groups even though they do not support modifier spans. Like a non-capturing group, the modifier span does not create a backreference.

Modifier spans are supported by all regex flavors that allow you to use mode modifiers in the middle of the regular expression, and by those flavors only. These include the JGsoft engine, .NET, Java, Perl and PCRE, PHP, Delphi, and R.

Make a Donation

Page URL: https://www.regular-expressions.info/modifiers.html
Page last updated: 22 November 2019
Site last updated: 08 April 2020