Free-Spacing Regular Expressions

Most modern regex flavors support a variant of the regular expression syntax called free-spacing mode. This mode allows for regular expressions that are much easier for people to read. Of the flavors discussed in this tutorial, only XML Schema and the POSIX and GNU flavors don't support it. Plain JavaScript doesn't either, but XRegExp does. The mode is usually enabled by setting an option or flag outside the regex. With flavors that support mode modifiers, you can put `(?x)` the very start of the regex to make the remainder of the regex free-spacing.

In free-spacing mode, whitespace between regular expression tokens is ignored. Whitespace includes spaces, tabs, and line breaks. Note that only whitespace *between* tokens is ignored. `a b c` is the same as `abc` in free-spacing mode. But `\ d` and `\d` are not the same. The former matches ` d`, while the latter matches a digit. `\d` is a single regex token composed of a backslash and a "d". Breaking up the token with a space gives you an escaped space (which matches a space), and a literal "d".

Likewise, grouping modifiers cannot be broken up. `(?>atomic)` is the same as `(?> ato mic )` and as `( ?>ato mic)`. They all match the same atomic group. They're not the same as `(? >atomic)`. The latter is a syntax error. The `?>` grouping modifier is a single element in the regex syntax, and must stay together. This is true for all such constructs, including lookaround, named groups, etc.

Exactly which spaces and line breaks are ignored depends on the regex flavor. All flavors discussed in this tutorial ignore the ASCII space, tab, line feed, carriage return, and form feed characters. JGsoft V2 and Boost are the only flavors that ignore all Unicode spaces and line breaks. JGsoft V1 almost does but misses the next line control character (U+0085). Perl always treats non-ASCII spaces as literals. Perl 5.22 and later ignore non-ASCII line breaks. Perl 5.16 and prior treat them as literals. Perl 5.18 and 5.20 treated unescaped non-ASCII line breaks as errors in free-spacing mode to give developers a transition period.

Free-Spacing in Character Classes

A character class is generally treated as a single token. `[abc]` is not the same as `[ a b c ]`. The former matches one of three letters, while the latter matches those three letters or a space. In other words: free-spacing mode has no effect inside character classes. Spaces and line breaks inside character classes will be included in the character class. This means that in free-spacing mode, you can use `\ ` or `[ ]` to match a single space. Use whichever you find more readable. The hexadecimal escape `\x20` also works, of course.

Java, however, does not treat a character class as a single token in free-spacing mode. Java does ignore spaces, line breaks, and comments inside character classes. So in Java's free-spacing mode, `[abc]` is identical to `[ a b c ]`. To add a space to a character class, you'll have to escape it with a backslash. But even in free-spacing mode, the negating caret must appear immediately after the opening bracket. `[ ^ a b c ]` matches any of the four characters `^`, `a`, `b` or `c` just like `[abc^]` would. With the negating caret in the proper place, `[^ a b c ]` matches any character that is not `a`, `b` or `c`.

Perl 5.26 offers limited free-spacing within character classes as an option. The `/x` flag enables free-spacing outside character classes only, as in previous versions of Perl. The double `/xx` flag additionally makes Perl 5.26 treat unescaped spaces and tabs inside character classes as free whitespace. Line breaks are still literals inside character classes. PCRE2 10.30 supports the same /xx mode as Perl 5.26 if you pass the flag PCRE2_EXTENDED_MORE to pcre2_compile().

Perl 5.26 and PCRE 10.30 also add a new mode modifier `(?xx)` which enables free-spacing both inside and outside character classes. `(?x)` turns on free-spacing outside character classes like before, but also turns off free-spacing inside character classes. `(?-x)` and `(?-xx)` both completely turn off free-spacing.

Java treats the `^` in `[ ^ a ]` as a literal. Even when spaces are ignored they still break the special meaning of the caret in Java. Perl 5.26 and PCRE2 10.30 treat `^` in `[ ^ a ]` as a negation caret in `/xx` mode. Perl 5.26 and PCRE2 10.30 totally ignore free whitespace. They still consider the caret to be at the start of the character class.

Comments in Free-Spacing Mode

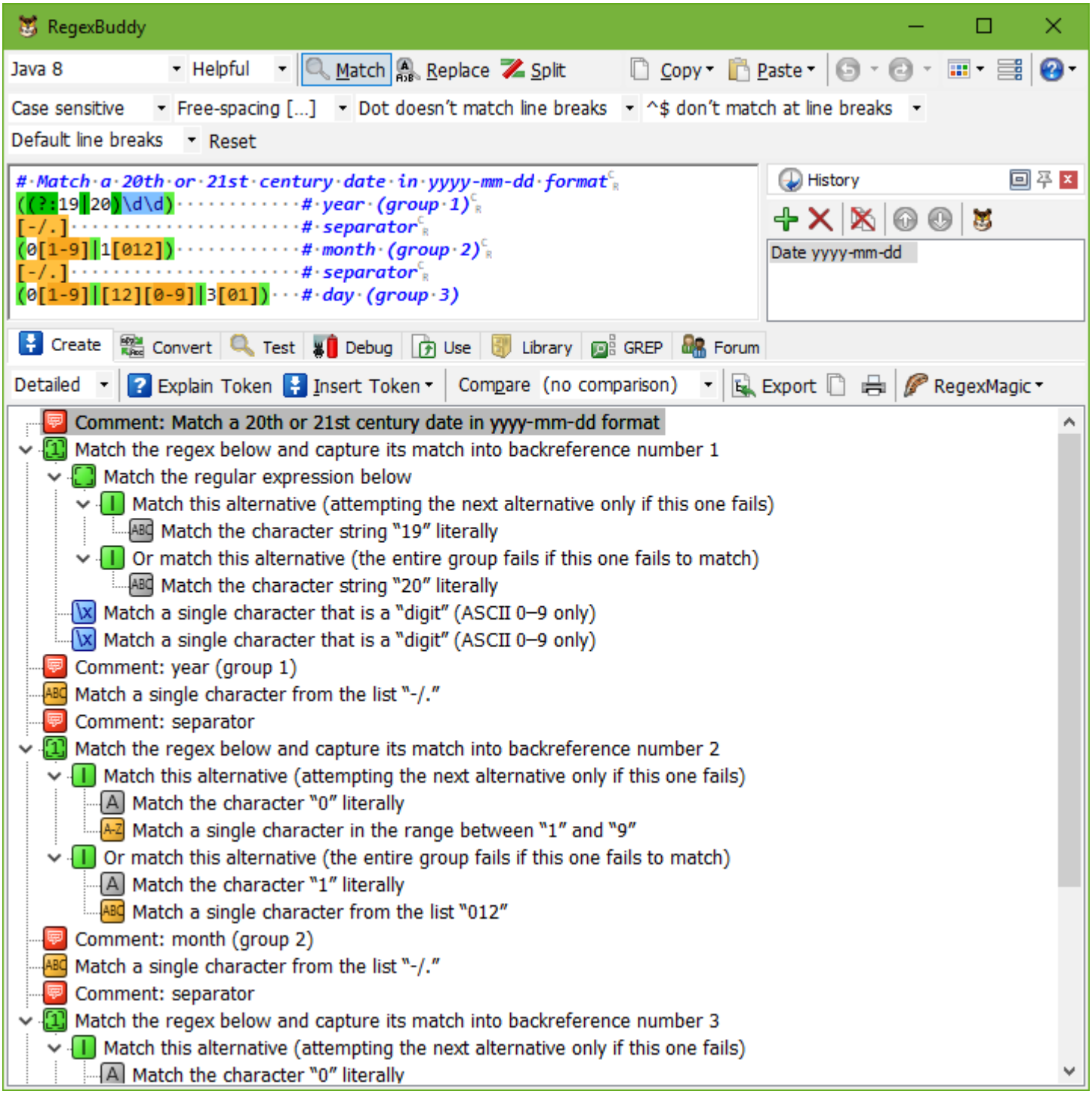
Another feature of free-spacing mode is that the # character starts a comment. The comment runs until the end of the line. Everything from the # until the next newline character is ignored. Most flavors do not recognize any other line break characters as the end of a comment, even if they recognize other line breaks as free whitespace or allow anchors to match at other line breaks. JGsoft V2 is the only flavor that recognizes all Unicode line breaks. Boost misses the vertical tab.

XPath and Oracle do not support comments within the regular expression, even though they have a free-spacing mode. They always treat # as a literal character.

Java is the only flavor that treats # as the start of a comment inside character classes in free-spacing mode. The comment runs until the end of the line, so you can use a] to close a comment. All other flavors treat # as a literal inside character classes. That includes Perl 5.26 in `/xx` mode.

Putting it all together, the regex to match a valid date can be clarified by writing it across multiple lines:

```
# Match a 20th or 21st century date in yyyy-mm-dd format
(19|20)\d\d                 # year (group 1)
[- /.]                      # separator
(0[1-9]|1[012])             # month (group 2)
[- /.]                      # separator
(0[1-9]|[12][0-9]|3[01])    # day (group 3)
```



Comments Without Free-Spacing

Many flavors also allow you to add comments to your regex without using free-spacing mode. The syntax is `(?#comment)` where "comment" can be whatever you want, as long as it does not contain a closing parenthesis. The regex engine ignores everything after the `(?#` until the first closing parenthesis.

Of the flavors discussed in this tutorial, all flavors that support comment in free-spacing mode, except Java and Tcl, also support `(?#comment)`. The flavors that don't support comments in free-spacing mode or don't support free-spacing mode at all don't support `(?#comment)` either.

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/freespacing.html
Page last updated: 22 November 2019
Site last updated: 08 April 2020