Optional Items

The question mark makes the preceding token in the regular expression optional. `colou?r` matches both `colour` and `color`. The question mark is called a quantifier.

You can make several tokens optional by grouping them together using parentheses, and placing the question mark after the closing parenthesis. E.g.: `Nov(ember)?` matches `Nov` and `November`.

You can write a regular expression that matches many alternatives by including more than one question mark. `Feb(ruary)? 23(rd)?` matches `February 23rd`, `February 23`, `Feb 23rd` and `Feb 23`.

You can also use curly braces to make something optional. `colou{0,1}r` is the same as `colou?r`. POSIX BRE and GNU BRE do not support either syntax. These flavors require backslashes to *give* curly braces their special meaning: `colou\{0,1\}r`.

Important Regex Concept: Greediness

The question mark is the first metacharacter introduced by this tutorial that is *greedy*. The question mark gives the regex engine two choices: try to match the part the question mark applies to, or do not try to match it. The engine always tries to match that part. Only if this causes the entire regular expression to fail, will the engine try ignoring the part the question mark applies to.

The effect is that if you apply the regex `Feb 23(rd)?` to the string `Today is Feb 23rd, 2003`, the match is always `Feb 23rd` and not `Feb 23`. You can make the question mark *lazy* (i.e. turn off the greediness) by putting a second question mark after the first.

The discussion about the other repetition operators has more details on greedy and lazy quantifiers.

Looking Inside The Regex Engine

Let's apply the regular expression `colou?r` to the string `The colonel likes the color green`.

The first token in the regex is the literal `c`. The first position where it matches successfully is the `c` in `colonel`. The engine continues, and finds that `o` matches `o`, `l` matches `l` and another `o` matches `o`. Then the engine checks whether `u` matches `n`. This fails. However, the question mark tells the regex engine that failing to match `u` is acceptable. Therefore, the engine skips ahead to the next regex token: `r`. But this fails to match `n` as well. Now, the engine can only conclude that the entire regular expression cannot be matched starting at the `c` in `colonel`. Therefore, the engine starts again trying to match `c` to the first o in `colonel`.

After a series of failures, `c` matches the `c` in `color`, and `o`, `l` and `o` match the following characters. Now the engine checks whether `u` matches `r`. This fails. Again: no problem. The question mark allows the engine to continue with `r`. This matches `r` and the engine reports that the regex successfully matched `color` in our string.

Make a Donation

Site last updated: 08 April 2020