Regex cheatsheet

Metacharacters defined

`^` Start of a string

`^abc` would match abc, abcdefg, abc123, ...

`$` End of a string

`abc$` would match abc, endsinabc, 123abc, ...

`.` Any character except `\n` (newline)

`a.c` would match abc, aac, acc, adc, aec, ...

`{...}` Explicit quantifier notation

`ab{2}c` would match abbc

`ab{2,4}c` would match abbc, abbbc, or abbbbc

`{,4}` indicates 4 or less repeats, `{2,}` indicates 2 or more repeats

`[...]` Explicit definition of a character class

`a[bB]c` would match abc or aBc

`(...)` Logical grouping of a part of an expression; can also be used for back referencing

`(abc){2}` would match abcabc

`*` 0 or more of the previous expression

`ab*c` would match ac, abc, abbc, abbbc, ...

`+` 1 or more of the previous expression

`ab+c` would match abc, abbc, abbbc, ...

`?` 0 or 1 of the previous expression

`ab?c` would match ac or abc

`|` alternation; this can be used to allow matching multiple, multi-character strings

`(bill|ted)` would match bill or ted

`\` Preceding one of the above characters, makes it a literal instead of a special character.

`ab\{2,4\}c` would match ab{2,4}c

Character classes

`.` Matches any character except `\n`

`[aeiou]` Matches any single character included in the specified character set

`[^aeiou]` Matches any single character not in the specified character set

`[0-9a-fA-f]` A hyphen specifies a contiguous character range (based on ASCII ordering)

Back referencing

If a regular expression or a portion of a regular expression is written within parentheses `()`, this portion of the match can be referenced later using `\` followed by a number. If there is only one set of parentheses used then `\1` would be used. If multiple sets of parentheses are used, you'll have to count `(` from left to right to determine what number to use behind the `\`.