Regular Expression Quick Reference

Match pattern

Special characters

`. ? + * ^ $ \ ( ) [ ] { } |` Need to be escaped with a backslash (`\`) to match the actual character

- Any other character matches itself

`.`	Matches one of any character
`(…)`	Groups elements into a single element (also captures contents)
`(?:…)`	Groups elements into a single element (doesn't captures contents)
`(…|…|…)`	Matches one of the alternatives

Character classes

`[abc]`	Matches any character (same as `(a|b|c)`)
`[^abc]`	Matches any other character

- Only `^ - \ ]` need to be escaped inside a character class
- May include simple ranges (eg, `[a-z123A-F]`)

`\d`	Matches digits (same as `[0-9]`)
`\D`	Matches non-digits (same as `[^0-9]`)
`\w`	Matches alphanumeric (same as `[a-zA-Z0-9_]`)
`\W`	Matches non-alphanumeric (same as `[^a-zA-Z0-9_]`)
`\s`	Matches whitespace (same as `[ ]`)*
`\S`	Matches non-whitespace (same as `[^ ]`)*

* In *RegexRenamer* the only relevant whitespace character is the space character

Anchors

- Anchors match the position between characters, not the characters themselves

`^`	Matches the position at the beginning of the line
`$`	Matches the position at the end of the line
`\b`	Matches the position between a `\w\W` or `\W\w` (word boundary)*
`\B`	Matches the position between a `\w\w` or `\W\W` (non-word boundary)

* `\b` also matches at the beginning and end of a line

Quantifiers

- Quantifiers are normally greedy (match as much as possible)
- When followed by `?` they become lazy (match as little as possible)

`?`	Match the previous element zero or one times (one if possible)
`??`	Match the previous element zero or one times (zero if possible)
`+`	Match the previous element one or more times (as many as possible)
`+?`	Match the previous element one or more times (as few as possible)
`*`	Match the previous element zero or more times (as many as possible)
`*?`	Match the previous element zero or more times (as few as possible)
`{n}`	Match the previous element exactly *n* times
`{n,}`	Match the previous element at least *n* times (as many as possible)
`{n,}?`	Match the previous element at least *n* times (as few as possible)
`{n,m}`	Match the previous element between *n* - *m* times (as many as possible)
`{n,m}?`	Match the previous element between *n* - *m* times (as few as possible)

Unnamed captures

`(…)`	Capture text matched between parentheses to an unnamed capture
`\n`	Match the text in capture #*n*, captured earlier in the match pattern

- The order of unnamed captures are defined by the order of the opening parentheses:
  `(reg)ex((re)(name)r)` — #1 = *reg*, #2 = *renamer*, #3 = *re*, #4 = *name*
- *n* > 9 is only available if you have more than 9 captures

Named captures

`(?<foo>…)`	Capture text matched between parentheses to a capture named "*foo*"
`\<foo>`	Match the text in capture "*foo*", captured earlier in the match pattern

Lookaround

`(?=…)`	Positive lookahead (match the position before the specified regex)
`(?!…)`	Negative lookahead (don't match, as above)
`(?<=…)`	Positive lookbehind (match the position after the specified regex)
`(?<!…)`	Negative lookbehind (don't match, as above)

Alternation

`(?(test)true)`	If positive lookahead *test* matches, match *true* regex
`(?(test)true|false)`	As above, otherwise match *false* regex
`(?(capture)true)`	If *capture* (name or number) contains text, match *true* regex
`(?(capture)true|false)`	As above, otherwise match *false* regex

Inline modifiers

`(?x)`	Turn on modifier *x* until the end of the containing group
`(?-x)`	Turn off modifier *x* until the end of the containing group
`(?x:…)`	Turn on modifier *x* for the section
`(?-x:…)`	Turn off modifier *x* for the section

- Relevent modifiers are `i` (ignore case) and `x` (extended regex).
- You may group more than one modifier together

Replace pattern

Any text other than the variables below will be replaced as-is.

Special variables

`$n`	Insert the contents of unnamed capture #*n*
`${foo}`	Insert the contents of named capture "*foo*"
`$0`	Insert all text matched in the regex (automatic unnamed capture)
`$` ` (backtick)	Insert text before `$0`
`$' (single-quote)`	Insert text after `$0`
`$_`	Insert the entire original filename (same as `$`0'`)
`$#`	Insert a number sequence (see Numbering)

`$$`	Insert an actual $ character (therefore, `$$#` to insert actual $#)

- For unnamed captures, use `${`*n*`}` if the following character is an actual digit
- *n* > 9 is only available if you have more than 9 captures