Vim Regular Expressions Cheat Sheet

by truenipple58 via cheatography.com/215225/cs/46860/

Search and Replace

`:range s/patt ern /st rin g/cgiI`	For each line in the **range** replace a match of the **pattern** with the **string**.
`c`	Confirm each substitution
`g`	Replace all occurrences in the line (without `g` only first occurence is replaced)
`i`	Ignore case for the pattern
`I`	Don't ignore case for the pattern

Range of Operation, Line Addressing and Marks

Specifier number	an absolute line number
.	the current line
$	the last line in the file
%	The whole file. Same as 1,$
't	position of mark "t"
/pattern[/]	the next line where text "pattern" matches
?pattern[?]	the previous line where text "pattern" matches
\/	the next line where the previously used search pattern matches
\?	the previous line where the previously used search pattern matches
\&	the next line where the previously used substitute pattern matches

Replacement String Options.

Backre fer-ences	Allows you to utilize patterns grouped using `\(` and `\)` and refer to them inside the replacement pattern by their order.
&	The whole matched pattern
\0	The whole matched pattern
\1	The matched pattern in the first pair of `\(\)`
\n	The matched pattern in the n'th pair of `\(\)`
~	The previous substitute string
Actions	Allow you to "act"
\L	The following characters are made lowercase.

Replacement String Options. (cont)

\U	The following characters are made uppercase.
\E	End of \U and \L
\e	End of \U and \L
\l	Next character is made lowercase.
\u	Next character is made uppercase.
\r	Split line in two at this point

Operator Precedence

Precedence	Description	Regexp
1	Grouping	`\(\)`
2	Quantifiers.	`\=`, `\+`, `*` etc.
3	Characters/Metacharacters not containing grouping or quantifiers.	`abc`, `\w`
4	Alternation/"OR"	|

Search and Execution

`:range g/patt ern/cmd`	Execute the Ex command **cmd** on the lines within **range** where **pattern** matches.
`:range g!/pat ter n/cmd`	Execute the Ex command **cmd** on the lines within **range** where **pattern** *does not* occur.

"Escaped" characters or metacharacters

.	Any character except new line		
\s	Whitespace character	\S	Non-Whitespace character
\w	Word character	\W	Non-Word character
\d	Digit	\D	Non-Digit
\a	Alphabetic character	\A	Non-Alphabetic character
\l	Lowercase character	\L	Non-Lowercase character
\u	Uppercase character	\U	Non-Uppercase character
\h	Head of a word character	\H	Non-head of word character
\p	Printable character	\P	**like \p, but excluding digits**
\x	Hex digit	\X	Non-Hex digit
\o	Octal digit	\O	Non-Octal digit

Quantifiers

Greedy	Greedy quantifiers first tries to repeat the token as many times as possible, and gradually gives up matches as the engine backtracks to find an overall match.
*	matches 0 or more of the preceding characters, ranges or metacharacters. * matches everything including empty line.
\+	matches 1 or more of the preceding characters
\=	matches 0 or more of the preceding characters
\{n,m}	matches from n to m of the preceding characters
\{n}	matches exactly n times of the preceding characters
\{,m}	matches at most m (from 0 to m) of the preceding characters
\{n,}	matches at least n of the preceding characters
Lazy	Lazy quantifier first repeats the token as few times as required, and gradually expands the match as the engine backtracks through the regex to find an overall match.
\{-}	matches 0 or more of the preceding atoms, **as few as possible.**
\{-n,m}	matches from n to m of the preceding characters, **as few as possible.**
\{-n,}	matches at least n of the preceding characters, **as few as possible.**
\{-,m}	matches at most m (from 0 to m) of the preceding characters, **as few as possible.**

n, m > 0

Additional Resources

1. vimregex Archived Link
It is an awesome and pretty comprehensive resource with examples and also provides a good summary, but not an aesthetically printable cheatsheet. Additionally, it never mentions very magic mode.
2. Learn By Example: Vim Regular Expressions Archived Link
Yet another resource, however does mention very magic mode.