

VIM QUICK REFERENCE CARD

Frequently used VIM commands – Version v1.2 August 2014

Vim cheatsheet based on the version by Michael Goerz

Movements

:viusage show a summary of all commands

Basic Movements

refer to helpsection **:h motion.txt**

h l k j character left, right; line up, down
b w word *or* token left, right
ge e end of word *or* token left, right
{ } beginning of previous, next paragraph
() beginning of previous, next sentence
[[]] go to previous, next function
[{]} beginning, end of current block
O _ \$ beginning, first, last character of line
nG ngg line *n*, default the last, first
n| column *n* of current line
% match of next brace, bracket, comment, **#define**
- + line up, down on first non-blank character
B W space-separated word left, right
gE E end of space-separated word left, right
gO gm beginning, middle of *screen* line
g^ g\$ first, last character of *screen* line
fc Fc next, previous occurrence of character *c*
tc Tc before next, previous occurrence of *c*

Insert, Command & Visual Mode

Insertion, Replace → insert mode

cm change text of movement command *m*
cc or S change current line

C change to the end of line
i a insert before, after cursor
o O open a new line below, above the current line
I A insert at beginning, end of line
gi gI insert text on last edited line, first column
rc grc replace character under cursor, without affecting layout
R gR replace characters starting at the cursor, without affecting layout
sc substitute character *c* under cursor → insert mode

Command Mode CTRL-Keys

^Vc ^Vn insert char *c* literally, decimal value *n*
^A ^@ insert previously inserted text, stop insert → command mode
^N ^P text completion before, after cursor
^W ^U delete word before cursor, to start of line
^D ^T shift left, right one shift width
^Oc execute *c* in temporary command mode
<esc> or ^] ... abandon edition → command mode

CTRL-R

^R^A content under cursor to command mode
^R = 5*5 insert 25 into text
^Rx ^R^Rx ... insert content of register *x*, literally

CTRL-X (Command Mode Completions)

^X^L whole lines
^X^N ^X^I keywords in current file, plus included files
^X^K ^X^N keywords in dictionary, thesaurus
^X^] tags
^X^F file names
^X^D definitions or macros
^X^V vim command line
^X^U user defined completion

^X^O omni completion

Visual Mode

refer to **:h object-select**

v V ^V start *or* stop highlighting characters, lines, block
o exchange cursor position with start of highlighting
gv start highlighting on previous visual area
aw as ap ... select a word, a sentence, a paragraph
ab aB select a block (**)**, a block { **}**
g^G Count words, character lines and bytes of selection

Delete, Copy to Registers

Deletion

x X delete character under, before cursor
dm delete text of movement command *m*
dd D delete current line, to the end of line
u U undo last command, restore last changed line
. ^R repeat last *n* changes, redo last undo

Copying

"x use register *x* for next delete, yank, put
ym yank the text of movement command *m*
yy or Y yank current line into register
p P put register after, before cursor position
]p [p like **p**, **P** with indent adjusted
gp gP like **p**, **P** leaving cursor after new text

Registers & Macros

show content of all registers: **:reg**

:@x execute register *x* as an *Ex* command

:register *x* show content of single register *x*

qx qX q..... record, append, stop recording typed characters as macro into register *x*

@x execute macro of register *x*

@@ repeat previous recorded macro

qxq..... empty register *x*

⇒ delete lines with pattern *p* into Register *x* and copy to clipboard afterwards:
:g/p/d *x* | let @+ = @x

q: @: list all, repeat macro

Search & Substitute

substitutions work like **:s/p/q/flag**, you may limit your search to an area between ranges (Ex ranges) – for appending *flags* such as *g*, *c*, *l* etc, see flaglist below!

Forward & Backward Searches

refer to **:h search-commands**

/s↔ ?s↔ ... search forward, backward for *s*

/s/o↔ ?s?o↔
search fwd, bwd for *s* with offset *o*

n or /↔ repeat forward last search

N or ?↔ repeat backward last search

* search backward, forward for word under cursor

g# g* same, but also find partial matches

gd gD local, global definition of symbol under cursor

Substitutions

refer to **:h :sub**

:rs/p/q/g ... substitute all *p* by *q* in range *r*

:rs q repeat substitution with new *r* & *q*

:rg/x/e :rv/x/e
execute *e* on range *r* where *x* matches, not matches

⇒ join any line containing the string *x* with previous line, if it lies between the *a* and *b* marks: **:’a,’bg/x/-1j**

:rg/x/s/p/q/g

for every line in *r* containing *x*, substitute *p* with *q*

:r& or :r~.... repeat last search, substitution on range *r*, but without flags

Ex Ranges

refer to **:h cmdline-ranges**

, cursor position interpreted from current line

; the cursor position will be set to line of last search or substitution

⇒ both , and ; separate line numbers. they differ in interpretation though: , (this line) and ; (that line)

n an absolute line number *n*

. \$ the current, last line in file

% * entire file, visual area

’*t* position of mark *t*

/p/ ?p? the next, previous line where *p* matches

-*n* +*n* preceding, appending line *n*

Patterns (differences to Perl)

refer to **:h pattern** and **:h /zero-width**

\< \> start, end of word

\i \k \I \K an identifier, keyword; excludigits

\f \p \F \P a file name, printable char.; excludigits

\e \t \r \b <esc>, <tab>, <↔>, <←>

\= * \+ match 0..1, 0..∞, 1..∞ of preceding atoms

\{*n,m*> match *n* to *m* occurrences

\{-> non-greedy match

\| separate two branches (≡ or)

\(\) group patterns into an atom

& \1 the whole matched pattern, 1st () group

\& a ‘branch’ matches last concat, but only if all preceding concats also match at the same position

⇒ the following pattern finds all lines that contain both "red" and "blue", in any order: **/. *red\&.*blue**

\u \l upper, lowercase character

\U \L id., whole pattern

\c \C ignore, match case on pattern

\%x match hex character

\@= \@! *char*(?=pattern) *char*(?!pattern)

\@<= \@<! (=pattern)<ichar (?!pattern)*char*

⇒ everything before the comment ‘#’ is excluded from pattern: **/\(#.*\)\@<=pattern**

\@> (?>pattern)

\-^ \-\$ start-of-line, end-of-line, anywhere in pattern

\. any single char, including end-of-line

\zs \ze set start, end of pattern

\%^ \%\$ match start, end of file

\%V match inside visual area

\’m match with position of mark *m*

\%(\) unnamed grouping

\[] collection with end-of-line included

\%[] sequence of optionally matched atoms

\v very magic: patterns almost like perl

Search, Substitute Flags

refer to **:h :s_flags**

c confirm each substitution

e do not issue error messages and continue as if no error occurred

g replace all occurrences in the line

i I ignore, mind case for the pattern (overwrites ‘ignorecase’ and ‘smartcase’ options)

p # l print the line containing the last substitute, like **:list**, prepend line number afterwards

&	must be the first one: keep flags from the previous substitute
n	report the number of matches, do not actually substitute. (the [c] flag is ignored.)

Advanced Operations

Special Text Operations

cg ⁿ dg ⁿ change, delete the next search pattern match (repeat change, deletion with <.>)
J gJ join current line with next, without space
~ g~ <i>m</i> switch case and advance cursor, on movement <i>m</i>
gum gUm	... switch case, lc, uc on movement <i>m</i>
guu gUU lower-/uppercase line

Advanced Scrolling

<i>n</i> ^Y <i>n</i> ^E scroll window <i>n</i> lines up, downwards
^D ^U scroll half a page up, down
^F ^B scroll page up, down
zt zz zb	... current line to top, center, bottom of win.
zh zl scroll one character to the right, left
zH zL scroll half a screen to the right, left

Marks and Tags

:tags	print tag list, :marks print the active marks list
mc mark current position with mark <i>c</i> ∈ [<i>a</i> .. <i>Z</i>]
‘ <i>c</i> ‘ <i>C</i> go to mark <i>c</i> in current, <i>C</i> in any file
‘0..9 go to last exit position
“ “ go to position before jump, at last edit
‘[‘] go to start, end of previously operated text

Jumps

print the jump list with :jumps

<i>n</i> ^O <i>n</i> ^I go to <i>n</i> th older, newer position in jump list
^] ^T jump to the tag under cursor, return from tag
^O ^I jump to older, newer location btw buffers
ng; ng, go to <i>n</i> older, newer position in change list
’. jump back on last edited line
’’ toggle back, forward to previous, next position

Ex Commands (↔)

refer to :help holy-grail for list of all commands

Tags

:tselect <i>t</i>	.. list matching tags and select one for jump
:tjump <i>t</i> jump to tag or select one if multiple matches
:tag ^[..... jump to tag (under cursor)

Reading from & writing to files

:edit <i>f</i> edit file <i>f</i> , reload current file if no <i>f</i>
:args <i>f</i> ₁ ... <i>f</i> _{<i>n</i>}	load files to buffer in background
:rwrite <i>f</i>	.. write range <i>r</i> to file <i>f</i> (this file if no <i>f</i>)
:rwrite >> <i>f</i>	append range <i>r</i> to file <i>f</i>
:quit :quit!	quit and confirm, discard changes
:wq or :x or ZZ	write to current file and exit
:rdelete :rdelete <i>x</i>	delete range <i>r</i> lines, into register <i>x</i>

Filter Lines

!mc↔ filter lines of movement <i>m</i> through command <i>c</i>
<i>n</i> !c↔ filter <i>n</i> lines through command <i>c</i>
:r!c filter range <i>r</i> lines through command <i>c</i>

Insert, Send Content

:r <i>f</i> insert content of file <i>f</i> below cursor
:r! <i>c</i> insert output of command <i>c</i> below cursor
:rcopy <i>a</i> :rmov ^e <i>a</i>	copy, move range <i>r</i> below line <i>a</i>
:rhardcopy > file.ps	print range to ps file
:rha rw!lp	.. sending <i>r</i> to printer (printout)

Compile

:clist :cfile	list all errors, read errors from file
:cnext :cp ^{revious}	display the next, previous error
:compiler <i>c</i>	set, show compiler plugins
:copen navigate errors from make
:make run makeprg, jump to first error

Standard Mode Formatting, Filtering

leave out *m* for visual mode commands

Indentation

set indent-foldmethod by :set fdm=indent

< <i>m</i> > <i>m</i>	.. shift left, right text of movement <i>m</i>
<i>n</i> > <i>n</i> < =	... indent, unindent <i>n</i> levels, reindent
<i>n</i> << <i>n</i> >>	.. shift <i>n</i> lines left, right

Alignment

gqm gqgq	... format movement <i>m</i> , current line
:rce <i>w</i> center lines in range <i>r</i> to width <i>w</i>
:rri <i>w</i> rightalign lines in range <i>r</i> to width <i>w</i>
:rle <i>i</i> left align lines in range with indent <i>i</i>

Folds

zfm create fold of movement <i>m</i>
:rfold create fold for range <i>r</i>
zd zE delete fold at cursor, all in window
zo zc zO zC	open, close one fold; recursively

[z]z move to start, end of current open fold
 zj zk move down, up to start, end of next fold
 zm zM fold more, close all folds
 zr zR fold less, open all folds
 zn zN zi ... fold non, fold normal, invert folding
 :set fdc=*n* . show foldcolumn to level *n*

Multiple Files, Buffers, Tabs (↔)

Generic Buffer Commands

:tab ball .. show buffers as tablist
 :buffers ... show list of buffers
 :on make current window one on screen
 :new :vnew . create new empty window (vert.)
 :bn switch to buffer *n*
 :bn :bp :bf :bl
 buffer movement next, prev, first, last
 :bdn delete buffer *n* (also with filename)
 ⇒ Delete all Buffers with Extension ‘ext’:
 :bd *.ext ^A
 :badd f.txt load file into new buffer
 :sbn Split window and edit buffer *n* from the
 bufflist

Buffer Shortcuts

refer to :h ctrl-w

^^ toggle between the current and the last window
 ^Wf gf open file under cursor in new, current window
 ^Ww ^W^W move to window below, above (wrap)
 ^Wj ^Wk move to window below, above
 ^Wt ^Wb move to top, bottom window
 ^Wc ^Wo close current, all other window(s)
 ^Ws ^Wv split window in two (vert.)
 ^Wx *or* ^W^R... swap open buffer windows
 ^Wn+ ^Wn- ... increase, decrease window size by *n* lines

^Wn > ^Wn < increase, decrease window width
 ^W = Make all windows equally high and wide
 ^W n_ set window height to *n* (default: very high)
 ^W n| set current window width to *n*

Tab Management

:tabs list all tabs including their displayed windows
 :tabfirst .. go to first tab
 :tablast ... go to last tab
 :tabnew open a new empty tab page
 :tabclose .. close current tab page
 :qall :wqall quit, and save all tabs
 :tabonly ... close all other tabs
 gt gT go to next, previous Tab
 ngt goto tab in position *n*

Miscellaneous

Spell Check

activate spellcheck: :set spell spelllang=en.us

]s [s next, previous misspelled word
 zg zG add good word (to internal word list)
 zug zuG..... undo the addition of a word to the dictionary
 zw zW mark bad word (to internal word list)
 z= suggest corrections

Invocation

vimdiff *f*₁ *f*₂
 diff *file*₁ + *file*₂ using synchronized split windows
 vim -o/-O *f*₁ *f*₂...*f*_{*n*}
 open *files* in horiz, vert split mode
 vim +*n file* . open *file* at *n*th line (eof if *n* omitted)
 vim +/*s file* open *file* and search for *string*

vim -S *name* reload vim-session *name*

Special operations

Usefull (and not so usefull) operations which don't fit to any other section :-)

K run **keywordprg** (manpage) on word under cursor
 ^A ^X..... increment, decrement number under cursor
 ^L redraw screen
 ga show ASCII value of character under cursor
 gf open filename under cursor
 ^Kc₁c₂ *or* c₁←c₂
 enter digraph {*c*₁,*c*₂}
 ⇒ for a complete list of all digraphs enter: :digraphs *or* :h digraph-table

Helpsections

:h /zero-width
 matches with ‘zero-width’ <@!> patterns

github.com/emzap79/QRCS

emzap79@gmail.com

This TeXfile is based on Gabriel B. Burcas © git-qrc.tex and has then been modified to my own requirements, with permission!