

VIM QUICK REFERENCE CARD

Frequently used VIM commands – Version v1.5 July 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/token left, right
`ge e` end of word/token left, right
`{ }` beginning of previous, next paragraph
`()` beginning of previous, next sentence
`[[]]` go to previous, next function
`[{ }]` beginning, end of current block
`0 _ $` beginning, first, last character of line
`nG ngg` line *n*, default the last, first
`n|` column *n* of current line
`%` match of next brace, bracket, comment, **#define**
`- +` line up, down on first non-blank character
`B W` space-separated word left, right
`gE E` end of space-separated word left, right
`g0 gm` beginning, middle of *screen* line
`g^ g$` 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

`i a` insert before, after cursor
`I A` insert at beginning, end of line

`gi gI` insert text on last edited line / first column
`o O` open a new line below, above the current line
`rc` replace character under cursor with *c*
`grc` like `r`, but without affecting layout
`R` replace characters starting at the cursor
`gR` like `R`, but without affecting layout
`sc` substitute char. *c* under cursor
`cm` change text of movement command *m*
`cc or S` change current line
`C` change to the end of line

Insert, Command Mode

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

Insert, Command Mode Completion

CTRL-R

`^R^A` content under cursor to command mode
`^Rx ^R^Rx` ... insert content of register *x*, literally

CTRL-X

`^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

`:h` object-select

`v V ^V` start/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`

```

:rdelete :rdelete x
           delete range r lines, into register x

:@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 in register x

@x ..... execute macro of register x

@@ ..... repeat previous recorded macro

qxq ..... empty register x

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

q: @: ..... list all, repeat previous Ex command(s)

```

Search & Substitute

refer to :h search-commands and :h :sub, respectively

Forward & Backward Searches

```

/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

substitutions work as :s/p/q/flag – for appending flags like *g*, *c*, *l* etc, see flaglist below!

```

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

:rs x ..... repeat substitution with new r & x

:rg/p/c :rv/p/c
           execute Ex c on range r where p matches,
           not matches

```

```

:rg/p/-1j .. join all lines in r containing pattern p
              with prevline

:rg/o/s/q/y/g
              for every line in r containing o, substitute
              p with q

:'a,'bg/o/j . join any line containing the string o to
               its subsequent line, if it lies between
               the a and b marks

```

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

```

⇒ the following pattern finds all lines that contain both "red" and "blue", in any order:

```

/. *red\&. *blue

```

```

\u \l       upper, lowercase character

\c \C       ignore, match case on next pattern

\%x         match hex character

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

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

```

⇒ everything which comes before the comment sign '#' gets ignored (special atom \@<=, to assert a match 'before'):

```

:%s/\(.*\)\@<=pattern/saturn/g

\@>         (>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

```

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, don't ignore case for the pattern (overwrite 'ignorecase' and 'smartcase' options)

p #         Print the line containing the last substitute, prepend line number afterwards

l           Like [p] but print the text like :list

&           must be the first one: keep flags from the previous substitute

⇒ Repeat last search or substitution with same substitute string: :&r (same as :s//c or :~)

n           Report the number of matches, do not actually substitute. (The [c] flag is ignored.)

```

Advanced Operations

Special Text Operations

cg**n** **dgn** change, delete the next search pattern match (repeat change, deletion with $\langle . \rangle$)
J **gJ** join current line with next, without space
 \sim **g** \sim **m** switch case and advance cursor, on movement *m*
gu**m** **gUm** ... switch case, lc, uc on movement *m*
gu**u** **gUU** lower-/uppercase line

Advanced Scrolling

n^{\wedge} **Y** n^{\wedge} **E** Scroll window *n* lines up-/downwards
 \mathcal{D} \mathcal{U} scroll half a page up, down
 \mathcal{F} \mathcal{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

Ex Commands (\leftrightarrow)

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

Reading from & writing to files

:edit *f* edit file *f*, reload current file if no *f*
:args $f_1 \dots f_n$ load files to buffer in background
:rwrite *f* .. write range *r* to file *f* (this file if no *f*)
:rwrite \gg *f* append range *r* to file *f*
:quit **:quit!** quit and confirm, quit and discard changes
:wq *or* **:x** *or* **ZZ**
write to current file and exit

Filter Lines

!mc \leftrightarrow filter lines of movement *m* through command *c*
 n^{\wedge} **!c** \leftrightarrow filter *n* lines through command *c*
:r!c filter range *r* lines through command *c*

Insert, Move Content

:r *f* insert content of file *f* below cursor
:r! *c* insert output of command *c* below cursor
:rcopy *a* **:rmove** *a*
copy, move range *r* below line *a*
:rhardcopy \gg **file.ps**
print range to ps file
:rha **rw!lp** . sending *r* to printer (printout)

Ex Ranges

both , and ; seperate line numbers. They differ in interpretation though (see also **:h cmdline-ranges**).

,	cursor position interpreted from current line (this line)
;	the cursor position will be set to that line before interpreting the next line specifier (that line)
<i>n</i>	an absolute line number <i>n</i>
. \$	the current, last line in file
% *	entire file, visual area
' <i>t</i>	position of mark <i>t</i>
/p/ ?p?	the next, previous line where <i>p</i> matches
- <i>n</i> + <i>n</i>	preceding/appending line number

Compile

:clist **:cfile**
list all errors, read errors from file
:cnext **:cprevious**
display the next, previous error
:compiler *c* 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**

$\langle m \rangle m$.. shift left, right text of movement *m*
 $n^{\wedge} n^{\wedge} =$... indent/unindent *n* levels, reindent
 $n^{\wedge} \ll n^{\wedge} \gg$.. shift *n* lines left, right

Alignment

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

Folds

zfm create fold of movement *m*
:rfold create fold for range *r*
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*

Tags, marks & jumps

:tags print tag list, **:marks** print the active marks list

Marks, Motions, and Tags

mc mark current position with mark *c* $\in [a..Z]$
'*c* '*C* go to mark *c* in current, *C* 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
 n^{\wedge} 0 go to n^{th} older position in jump list

n^{th} go to n^{th} newer position in jump list
 $\wedge]$ $\wedge\text{T}$ jump to the tag under cursor, return from tag
:tselect t . list matching tags and select one for jump
:tjump t jump to tag or select one if multiple matches

Jumps

print the jump list with **:jumps**

:tag $\wedge[$ jump to tag (under cursor)
 $\wedge\text{O}$ $\wedge\text{I}$ jump to older/newer location btw buffers
 $ng;$ $ng,$ go to n older/newer position in change list
 $'.$ jump back on last edited line
 $''$ toggle back/forward to previous/next position

Multiple Files, Buffers, Tabs (\leftrightarrow)

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)
 \Rightarrow Delete all Buffers with Extension 'ext':
:bd *.ext $\wedge\text{A}$
:badd $f.txt$ load file into new buffer
:sbn Split window and edit buffer n from the buflist

Buffer Shortcuts

:h ctrl-w

$\wedge\wedge$ toggle between the current and the last window

$\wedge\text{Wf}$ **gf** open file under cursor in new/current window
 $\wedge\text{Ww}$ $\wedge\text{W}\text{W}$ move to window below, above (wrap)
 $\wedge\text{Wj}$ $\wedge\text{Wk}$ move to window below, above
 $\wedge\text{Wt}$ $\wedge\text{Wb}$ move to top/bottom window
 $\wedge\text{Wc}$ $\wedge\text{Wo}$ close current/all other window(s)
 $\wedge\text{Ws}$ $\wedge\text{Wv}$ split window in two (vert.)
 $\wedge\text{Wx}$ or $\wedge\text{W}\text{R}$... swap open buffer windows
 $\wedge\text{Wn}+$ $\wedge\text{Wn}-$... increase/decrease window size by n lines
 $\wedge\text{Wn}$ $>$ $\wedge\text{Wn}$ $<$ increase/decrease window width
 $\wedge\text{W} =$ Make all windows equally high and wide
 $\wedge\text{W}$ n_+ set window height to n (default: very high)
 $\wedge\text{W}$ $n|$ 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
:wqa **:qa** ... (save and) quit all tabs
:tabonly ... close all other tabs
gt **gT** go to next/previous Tab
ngt goto tab in position n

Miscellaneous

Spell Check

:set **spell** **spelllang=de_20**
 activate spellcheck
 $]s$ $[s$ 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

Vim Start-Options

vimdiff f_1 f_2
 diff $file_1 + file_2$ using synchronized split windows
vim $-o/-O$ f_1 $f_2 \dots f_n$
 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
 $\wedge\text{A}$ $\wedge\text{X}$ increment/decrement number under cursor
 $\wedge\text{L}$ redraw screen
 $\wedge\text{R} = 5*5$ insert 25 into text
ga show ASCII value of character under cursor
gf open filename under cursor
 $\wedge\text{K}c_1c_2$ or $c_1 \leftarrow c_2$
 enter digraph $\{c_1, c_2\}$
 \Rightarrow for a complete list of all digraphs enter:
:digraphs or **:h digraph-table**

Helpsections

:h /zero-width
 matches with 'zero-width' $\langle @! \rangle$ 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!