

## 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  
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*

### 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  
^Rx ^R^Rx ... insert content of register *x*, literally  
^R^W ..... content under cursor to 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  
^X^E ^X^Y ... scroll up, down  
<esc> *or* ^] ... abandon edition → command mode

### Copying, Deleting 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  
dgn ..... delete the next search pattern match  
:rd :rdx ..... delete range *r* lines, into register *x*  
u U ..... undo last command, restore last changed line

### 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

. ^R ..... repeat last changes, redo last undo  
:reg :reg *x* show the content of all, specific register *x*  
n. .... repeat last changes with count replaced by *n*  
qc qC ..... record, append typed characters in register *c*  
q ..... stop recording  
qxq ..... empty register *x*  
⇒ delete lines with pattern *c* into Register *x* and copy to clipboard afterwards:  
:g/c/d *x* | let @+ = @x  
@c ..... execute the content of register *c*  
@@ ..... repeat previous @ command  
:@c ..... execute register *c* as an *Ex* command  
q: @: ..... list all, repeat previous *Ex* command(s)

### Visual Mode & Advanced Operations

#### 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

---

Join lines / change cases

**J gJ** ..... join current line with next, without space

**~ g~m**..... switch case and advance cursor, on movement *m*

**gum gUm** ... switch case, lc, uc on movement *m*

**guu gUU**..... lower-/uppercase line

---

Advanced Scrolling

**n^Y n^E**..... Scroll window *n* 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

**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 like **:s/p/q/flag** – for appending flags like *g*, *c*, *l* etc, see subsection 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, 1<sup>st</sup> () 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*

⇒ **You can use the special atom \@<= to assert a match before; use .\* to make it anywhere before in that line:  
: %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.)

---

**Ex Commands (↵)**

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

Reading from & writing to files

`:e f` ..... edit file *f*, reload current file if no *f*  
`:rw f` ..... write range *r* to file *f* (this file if no *f*)  
`:rw >> f` .... append range *r* to file *f*  
`:q :q!` ..... quit and confirm, quit and discard changes  
`:wq` *or* `:x` *or* `ZZ`  
write to current file and exit

Filter Lines

`!mc↔` ..... filter lines of movement *m* through command *c*  
`n!!c↔` ..... filter *n* lines through command *c*  
`:r!c` ..... filter range *r* lines through command *c*

Insert *or* Move Content

`:r f` ..... insert content of file *f* below cursor  
`:r! c` ..... insert output of command *c* below cursor  
`:rco a :rm a` copy, move range *r* below line *a*  
`:rhardcopy rw!lp`  
sending *r* to printer (printout)  
`:rhardcopy > file.ps`  
print range to ps file

Misc :exe Commands

`:mkview [f] :loadview [f]`  
save/load configuration  
`:mks [f] :so [f]`  
save/load session  
`:set ff=dos` convert file to dos eol format  
`:e ++ff=unix` reopen file in unix eol format  
`:sh :!c` .... start shell, execute command *c* in shell

Ex Ranges

both , and ; separate 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)  
*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 number

Compiling

`:compiler c` set/show compiler plugins  
`:make` ..... run `makeprg`, jump to first error  
`:cope` ..... navigate errors from make  
`:cn :cp` .... display the next, previous error  
`:cl :cf` .... list all errors, read errors from file

Standard Mode Formatting/ Filtering

leave out *m* for visual mode commands

Indentation

set indent-foldmethod by `:set fdm=indent`

< *m* > *m* .. shift left, right text of movement *m*  
*n*> *n*< = ... indent/unindent *n* levels, reindent  
*n* << *n* >> .. shift *n* lines left, right  
`G=gg` ..... auto (re)indent entire document

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*

`:rfo` ..... 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 foldcolumn=n`  
show foldcolumn *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* ∈ [*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*^0 ..... go to *n*<sup>th</sup> older position in jump list  
*n*^I ..... go to *n*<sup>th</sup> newer position in jump list  
^] ^T ..... jump to the tag under cursor, return from tag  
`:ts t` ..... list matching tags and select one for jump  
`:tj t` ..... jump to tag or select one if multiple matches

Jumps

print the jump list with `:jumps`

`:ta ^[` ..... jump to tag (under cursor)  
^0 ^I ..... jump to older/newer location btwbuffers  
*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 (↔)

### 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)  
 :badd f.txt load file into new buffer  
 :sbn . . . . . Split window and edit buffer *n* from the bufflist

### Buffer Shortcuts

:h ctrl-w  
 :bd \*.ext ^A Deletes all Buffers with Extension 'ext'  
 ^~ . . . . . toggle between the current and the last window  
 ^Wf gf . . . . . open file under cursor in new/current window  
 ^Ww ^WW . . . . . 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 . . . . . 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  
 :tabc . . . . . close current tab page  
 :wqa :qa . . . . . (save and) quit all tabs  
 :tabo . . . . . 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

#### Completion

^XL . . . . . whole lines  
 ^XN ^XI . . . . . keywords in current file, plus included files  
 ^XK ^XN . . . . . keywords in dictionary, thesaurus  
 ^X] . . . . . tags  
 ^XF . . . . . file names  
 ^XD . . . . . definitions or macros  
 ^XV . . . . . vim command line  
 ^XU . . . . . user defined completion

^X0 . . . . . omni completion

### Vim Start-Options

vimdiff *f*<sub>1</sub> *f*<sub>2</sub>  
                   diff *file*<sub>1</sub> + *file*<sub>2</sub> using synchronized split windows  
 vim -o/-O *f*<sub>1</sub> *f*<sub>2</sub>  
                   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 any other section :-)

K . . . . . run **keywordprg** (manpage) on word under cursor  
 ^A ^X . . . . . increment/decrement number under cursor  
 ^L . . . . . redraw screen  
 ^R = 5\*5 . . . . . insert 25 into text  
 ga . . . . . show ASCII value of character under cursor  
 gf . . . . . open filename under cursor  
 ^Kc<sub>1</sub>c<sub>2</sub> or c<sub>1</sub>←c<sub>2</sub>  
                   enter digraph {*c*<sub>1</sub>, *c*<sub>2</sub>}  
 ⇒       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.