VIM QUICK REFERENCE CARD

Frequently used VIM commands – Version v1.3 July 2014

`:viusage`	Show a summary of all commands

Movements

see `:h motion.txt`

Basic Movements

`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 _ $`	beginning, first, last character of line
n`G` *n*`gg`	line *n*, default the last, first
n`|`	column *n* of current line
`%`	match of next brace, bracket, comment, `#define`
`- +`	line up, down on first non-blank character
`B W`	space-separated word left, right
`gE E`	end of space-separated word left, right
`g0 gm`	beginning, middle of *screen* line
`g^ g$`	first, last character of *screen* line
`f`*c* `F`*c*	next, previous occurence of character *c*
`t`*c* `T`*c*	before next, previous occurence 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
`r`*c*	replace character under cursor with *c*
`gr`*c*	like `r`, but without affecting layout
`R`	replace characters starting at the cursor
`gR`	like `R`, but without affecting layout
`s`*c*	substitute char. *c* under cursor
`c`*m*	change text of movement command *m*
`cc` *or* `S`	change current line
`C`	change to the end of line

Insert/ Command Mode

`^V`*c* `^V`*n*	insert char *c* literally, decimal value *n*
`^A`	insert previously inserted text
`^@`	same as `^A` and stop insert → command mode
`^R`*x* `^R^R`*x*	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
`^O`*c*	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
`d`*m*	delete text of movement command *m*
`dd D`	delete current line, to the end of line
`dgn`	delete the next search pattern match
`:`*r*`d` `:`*r*`d`*x*	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
`y`*m*	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*
`q`*c* `q`*C*	record, append typed characters in register *c*
`q`	stop recording
`q`*x*`q`	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*
`gu`*m* `gU`*m*	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 *or* Substitute

see `: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!

`:`*r*`s/`*p*`/`*q*`/g`	substitute all *p* by *q* in range *r*
`:`*r*`s` *x*	repeat substitution with new *r* & *x*
`:`*r*`g/`*p*`/`*c* `:`*r*`v/`*p*`/`*c*	execute *Ex c* on range *r* where *p* matches, not matches
`:`*r*`g/`*p*`/-1j`	join all lines in *r* containing pattern *p* with prevline
`:`*r*`g/`*o*`/s/`*q*`/`*y*`/g`	for every line in *r* containing *o*, substitute *p* with *q*
`:'`*a*`,'`*b*`g/`*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)

see `:h pattern` and `:h zero-width`

`\< \>`	start, end of word
`\i \k \I \K`	an identifier, keyword; excldigits
`\f \p \F \P`	a file name, printable char.; excldigits
`\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*
`\@>`	`(?>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

see `: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 (↩)

see `: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*
`:`*r*`w` *f*	write range *r* to file *f* (this file if no *f*)
`:`*r*`w` ≫ *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
`:`*r*`co` *a* `:`*r*`m` *a*	copy, move range *r* below line *a*
`:`*r*`hardcopy` *r*`w!lp`	sending *r* to printer (printout)
`:`*r*`hardcopy > 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 `;` 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)
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

`gq`*m* `gqgq`	format movement *m*/current paragraph
`:`*r*`ce` *w*	center lines in range *r* to width *w*
`:`*r*`ri` *w*	rightalign lines in range *r* to width *w*
`:`*r*`le` *i*	left align lines in range with indent *i*

Folds

`zf`*m*	create fold of movement *m*
`:`*r*`fo`	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

`m`*c*	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`ˆO`	go to n^{th} older position in jump list
n`ˆI`	go to n^{th} newer position in jump list
`ˆ] ˆT`	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 ˆ[`	jump to tag (under cursor)
`ˆO ˆI`	jump to older/newer location btwbuffers
n`g;` *n*`g,`	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.)
`:b`*n*	switch to buffer *n*
`:bn :bp :bf :bl`	buffer movement next, prev, first, last
`:bd`*n*	delete buffer *n* (also with filename)
`:badd f.txt`	load file into new buffer
`:sb`*n*	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 ^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`	swap open buffer windows
`^W`*n*`+ ^W`*n*`-`	increase/decrease window size by *n* lines
`^W`*n* > `^W`*n* <	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
n`gt`	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

`^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

Vim Start-Options

`vimdiff` f_1 f_2	diff $file_1$ + $file_2$ using synchronized split windows
`vim -o/-O` f_1 f_2	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
`^K`c_1c_2 *or* $c_1 \leftarrow c_2$	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