VIM QUICK REFERENCE CARD

`:viusage`	Show a summary of all commands

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
`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`	insert text in 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
`c`*m*	change text of movement command *m*
`cc` *or* `S`	change current line
`C`	change to the end of line

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
`J gJ`	join current line with next, without space
`:`*r*`d`	delete range *r* lines
`:`*r*`d`*x*	delete range *r* lines into register *x*

Insert Mode

`^V`c `^V`n	insert char c literally, decimal value n
`^V`n	insert decimal value of character
`^A`	insert previously inserted text
`^@`	same as `^A` and stop insert → command mode
`^R`x `^R^R`x	insert content of register x, literally
`^N` `^P`	text completion before, after cursor
`^W`	delete word before cursor
`^U`	delete all inserted character in current line
`^D` `^T`	shift left, right one shift width
`^K`c_1c_2 *or* c_1←c_2	enter digraph $\{c_1, c_2\}$
`^O`c	execute c in temporary command mode
`^X^E` `^X^Y`	scroll up, down
⟨*esc*⟩ *or* `^[`	abandon edition → command mode

Search & Substitution

`/`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
`:`r`s/`f`/`t`/`x	substitute f by t in range r
	⌊ x : `g`—all occurrences, `c`—confirm changes
`:`r`s` x	repeat substitution with new r & x

Formatting/Filtering

m is movement; leave out for visual mode commands

`gq`m	format m (*formatprg*)
`gqgq` *or* `gqip`	format current paragraph
$=m$	reindent movement m (*equalprg*)
`:`r`ce` w	center lines in range r to width w
`:`r`le` i	left align lines in range r with indent i
`:`r`ri` w	right align lines in range r to width w
`!`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
`~`	switch case and advance cursor
`g~`m `gu`m `gU`m	switch case, lc, uc on movement m
$<m$ $>m$	shift left, right text of movement m
$n\ll$ $n\gg$	shift n lines left, right
`^A` `^X`	increment/decrement number under cursor

Visual Mode & Text Objects

`v V ^V`	highlight characters, lines, block
`o`	exchange cursor pos with start of highlighting
`gv`	start highlighting on previous visual area
`aw as ap`	select a word, a sentence, a paragraph
`ab aB a] a>`	select a block ( ), { }, [ ] < >
`at`	select an html tag
`a" a' a‘`	select a quotet string
`iw is ip ib iB i] i> it i" i' i‘`	select “inner”
n> n< =	indent/unindent n levels, reindent

Undoing, Repeating & Registers

`u U`	undo last command, restore last changed line
`. ^R`	repeat last changes, redo last undo
n`.`	repeat last changes with count replaced by n
`q`c `q`C	record, append typed characters in register c
`q`	stop recording
`@`c	execute the content of register c
`@@`	repeat previous `@` command
`:@`c	execute register c as an *Ex* command

Copying (Yanking)

`"`x	use register x for next delete, yank, put
`:reg [`x`]`	show content registers/x reg
`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

Patterns (differences to Perl) `:help pattern`

`\< \>`	start, end of word
`\i \k \I \K`	an identifier, keyword; excl. digits
`\f \p \F \P`	a file name, printable char.; excl. digits
`\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
`\u \l`	upper, lowercase character
`\c \C`	ignore, match case on next pattern
`\%x`	match hex character

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

Spell Checking

:set spell spelllang=de_20	activate spellcheck
]s	next misspelled word
zg zG	add good word (to internal word list)
zw zW	mark bad word (to internal word list)
z=	suggest corrections

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
‘‘ ‘"	go to position before jump, at last edit
‘[‘]	go to start, end of previously operated text
:marks	print the active marks list
:jumps	print the jump list
n^O	go to n^{th} older position in jump list
n^I	go to n^{th} 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
gf	open file which filename is under cursor
:tags	print tag list

Multiple Files / Buffers (⟷)

:tab ball	show buffer tablist
:buffers	show list of buffers
:buffer n	switch to buffer n
:badd f.txt	load file into new buffer
:bdelete n	delete buffer n (also with filename)
:bnext :bprev :bfirst :blast	buffer movement

Scrolling & Multi-Windowing

^D ^U ^F ^B	scroll half / full page page down, up
zt zz zb	current line to top, center, bottom of win
:[v]split ^Ws ^Wv	split window (horiz., vert.)
:vert help	open help vertically
^Wf ^W]	open file, tag in split
:[v]new ^W^n	new empty window (horiz., vert.)
:on ^Wo	make current win the only one
^Wj ^Wk ^Wh ^Wl	move down, up, left, right to win
^WJ ^WK ^WH ^WL	move win down, up, left, right
^W*n*+ ^W*n*-	increase/decrease win size by *n* lines
^W*n* > ^W*n* <	increase/decrease window width
^W*n*_ ^W*n*|	set height/width (max if no *n*)
^W=	Make win size equal
^Wr ^Wx	Rotate, exchange windows
^Wc	close window

Misc Ex Commands ($\hookleftarrow$) :help holy-grail

:e *f*	edit file *f*, reload current file if no *f*
:*r* w *f*	write range *r* to file *f* (current file if no *f*)
:*r* w>>*f*	append range *r* to file *f*
:*r* g[!]/*p*/*cmd*	ex *cmd* on range *r* [not] matching *p*
:q :q!	quit and confirm, quit and discard changes
:wq *or* :x *or* ZZ	write to current file and exit
:r *f*	insert content of file *f* below cursor
:r! *c*	insert output of command *c* below cursor
:*r*c *a* :*r*m *a*	copy, move range *r* below line *a*

Ex Ranges

, ;	separates two lines numbers, set to first line
n	an absolute line number *n*
. $	the current line, the last line in file
% *	entire file, visual area
'*t*	position of mark *t*
/*p*/ ?*p*?	the next, previous line where *p* matches
+*n* -*n*	+*n*, −*n* to the preceding line number

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^] ^X^F ^X^D	tags, filenames, defs/macros
^X^V	vim command line
^X^U ^X^O	user defined, omni- completion

Folding

`:set fdm=indent`	indent-foldmethod
`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=4`	show foldcolumn

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

Miscellaneous

`:sh :!`*c*	start shell, execute command *c* in shell
`K`	run `keywordprg` (man) on word under cursor
`^L`	redraw screen
`^G`	show cursor column, line, and character position
`:set cuc`	show cursor column visually
`ga`	show ASCII value of character under cursor
`:mkview` [*f*] `:loadview` [*f*]	save/load configuration
`:set ff=dos`	convert file to dos eol format
`:e ++ff=unix`	reopen file in unix eol format
`:write ++enc=utf-8`	Write file in utf-8
`:set hlsearch`	highlight searches
`:`*r*`hardcopy > file.ps`	print range to ps file
`:set list`	show listchar characters (tabs etc.)