Vim condensed *Quick*Reference

Navigation

←,→, middle of line, $\mathcal{N}$th column ..... h l gm |
|◀◀,|◀,▶|,▶▶| ($\mathcal{N}$-1 lines ↓) ..... 0 ^ $
the same for screen line ..... g0 g^ g$
to, till the char *c* →/← ..... f*c* t*c* F*c* T*c*
repeat last fFtT, in opposite direction ..... ; ,
↑↓, for screen lines ..... k j gk gj
$\mathcal{N}$ lines ↑↓ to ①, $\mathcal{N}$-1 lines ↓ ..... - + _
goto line $\mathcal{N}$ with default: ▼, ▲; $\mathcal{N}$% ..... G gg %
matching paren, brace...; goto $\mathcal{N}$th byte .... % go$_{\equiv:\mathcal{R}go\mathcal{N}}$
$\mathcal{N}$th *window* line from ↓↑, middle line ..... H L M
jump list, $\mathcal{N}$th older/newer pos ..... :ju$_{mps}$ ^O ^I
while entering count: delete last char ..... *del*

Text object motions

word →,← / WORD ..... w b W B
‖ of word →,← / WORD ..... e ge E gE
sentence →,← / paragraph .....) (} {
section → at ‖/‖; ← .....]]][[[[]
(, {,), }, /*, */ ..... [([{])]} [*]*
←,→ Java method ‖/‖ ..... [m]m [M]M
←,→ unclosed #if, #else, #endif ..... [#]#

Text object selection

text *O*bject, inner *O*bject ..... a*O* i*O*
O: word, WORD, sentence, paragraph .... w W s p
XML tags; enclosed in .... t (≡) [≡] {≡} <≡> " ' `

Marks

mark current position, list marks ... m*c* :marks [*marks*]
goto local, global / ① ..... `*c* `*C* '*c* '*C*
before jump; last *n* exit, edit, change . `` `*n* `" `.
‖/‖ of yanked or changed text / $\mathcal{V}$. `[$_{\equiv'[}$ `] `< `>
|☜|, |☞ local mark / ① ..... ['] [`] [`]]`
delete, delete local ..... :delm$_{arks}$ *marks* :delm$_{arks}$!

Ex mode

switch to *Ex* mode, with cmd-line editing ..... Q gQ
start entering *Ex* cmd with $\mathcal{N}$ lines ..... $\mathcal{N}$:
exec normal mode commands ..... :norm$^{!}_{all}$ *cmds*
leave mode ..... :vi$^{!}_{sual}$ [++*opt*][+*cmd*][*f*]
read Ex from file, normal cmds ..... :so$_{urce}$ *file* :so!
idem, search in `runtimepath` (!≡all matching) .:ru$^{!}_{ntime}$ *f*

Ex Ranges

line nr separator, absolute nr; +- line nr .. , $\mathcal{N}$ +[*n*] -[*n*]
use instead of , to set cursor to the next line after 1st arg .;
last line, current; all lines ..... $. %$_{\equiv1,\$}$
visual area, pos of mark *t* ..... *$_{\equiv'<,'>}$ '*t*
|☞|, |☜| matching line .... /*pattern*[/] ?*pattern*[?]

Information

redraw screen, show filename and pos ..... ^L ^G
show pos info, ascii, utf-8 ..... g^G ga g8
version info ..... :ve$_{rsion}$
show help, search help .... :h$_{elp}$ [*subj*] :helpg$_{rep}$ *patt*
lookup keyword with `keywordprg` ..... K
generate help tags file for directory ..... :helpt$_{ags}$ *dir*

Files and OS

format for *edit*, *new*, *view*... ..*cmd* [++*opt*] [+*cmd*] [*file*]
reload, edit *f*, edit new ..... :e$^{!}_{dit}$:e$^{!}_{dit}$*f* :ene$^{!}_{w}$
++ *options* ..... ff$_{\equiv file format}$ enc bin nobin
open readonly, find in path ..... :vie$_{w}$ *f* ? fin$^{!}_{d}$*f*
edit $\mathcal{N}$th alternate, goto file under cursor ..... ^^ gf
cd, for curr window ..... :cd *path* :lcd *path*
cd to |☜|, curr file dir, print curr :cd - :cd %:h :pwd
print, set curr filename, save as .. :f$_{ile}$:f$_{ile}$ *f* :sav$^{!}_{eas}$ *f*
show alternate filenames, recover ... :files :rec$^{!}_{over}$*f*
edit *f* making current hidden ..... :hid$_{e}$ e$_{dit}$*f*
write, to *cmd*'s *stdin* . :$\mathcal{R}$w$^{!}_{rite}$ [>>] [*file*] :$\mathcal{R}$w$_{rite}$!*cmd*
write all, write if modified .. :wa$^{!}_{ll}$:$\mathcal{R}$up$^{!}_{date}$ [>>] [*file*]
quit ..... :q$^{!}_{uit}$:qa$^{!}_{ll}$:wq$^{!}$ [*file*] :q!$_{\equiv ZQ}$
write if mofified & quit ..... :x$^{!}_{it}$ [*file*]$_{\equiv ZZ}$:xa$^{!}_{ll}$
read *file* contents, *cmd*'s *stdout* :r [++*opt*] *file* :r! *cmd*
start shell, exec *cmd*, suspend :sh$_{ell}$:!*cmd* :st$_{op\equiv\wedge Z}$
filter lines along μ through external *cmd* (+$\mathcal{V}$) .... !μ*cmd*↲
filter $\mathcal{N}$ lines, range through *cmd* ... !!*cmd*↲ :$\mathcal{R}$! *cmd*↲
filter through `equalprg`+$\mathcal{V}$, $\mathcal{N}$ lines ..... =μ ==

Settings

read/write viminfo ..... :rv$^{!}_{iminfo}$ [*f*] :wv$^{!}$ [*f*]
save maps and options; window :mkv$^{!}_{imrc}$ [*f*] :mkvie$^{!}_{w}$ [*f*]
save session; load window ... :mks$^{!}_{ession}$ [*f*] :lo$_{adview}$ [*f*]

Arguments

print, set arglist ..... :ar$_{gs}$:ar$_{gs}$ [*files*]
edt $\mathcal{N}$th file,open all ..... ? argu$_{ment}\mathcal{N}$:all
write & edit |☞, |☜| ..... :wn$^{!}_{ext}$ [*f*] :wN$_{ext\equiv:wprev}$
move to ..... ? n$_{ext}$? N$_{ext}$? fir$_{st}$? la$_{st}$
set new arglist & edit 1st file ..... ? n$_{ext}$ *files*
exec *cmd* for each file ..... :argdo$^{!}$ *cmd*

Buffers

show all, include unlisted ..... :ls :ls!
edit all $\mathcal{N}$ buffers; all loaded ..... ? ba$_{ll}$? unh$_{ide}$
add, del ($\mathcal{N}$th) ..... :bad$_{d}$ *fname* :bd$^{!}_{elete}$
unload, total delete ..... :bun$^{!}_{load}$:bw$^{!}_{ipeout}$
edit buf by *num*/*name*, $\mathcal{N}$th modified ? b$^{!}_{uffer}$? bm$^{!}_{odified}$
$\mathcal{N}$th |☞, |☜|; |◀, ▶|. ? bn$^{!}_{ext}$:bN$^{!}_{ext}$? bf$^{!}_{irst}$? bl$^{!}_{ast}$
exec *cmd* in each buffer ..... :bufdo$^{!}$ *cmd*

Tabs

new page file ..... :tabe$_{dit\equiv:tabnew}$[++*opt*][+*cmd*][*file*]
find file and open new page . :tabf$_{ind}$[++*opt*][+*cmd*][*file*]
exec *cmd* opening tab #$\mathcal{N}$ instead of window :$\mathcal{N}$tab *cmd*
file under cursor; close, others ^Wgf :tabc$^{!}_{lose}\mathcal{N}$:tabo$^{!}_{nly}$
:tabn$_{ext\equiv gt}\mathcal{N}$:tabN$_{ext\equiv gT\equiv:tabprev}$:tabr$_{ewind}$:tabl$_{ast}$
list,move after $\mathcal{N}$th,iterate :tabs :tabm$_{ove}\mathcal{N}$:tabd$_{o}$*cmd*

Syntax highlighting and filetype detection

start, start and force defaults .. :sy$_{ntax}$ enable :sy on
stop; list syntax items ..... :sy off :sy [list]
filetype detection .. :filet$_{ype}$ [plugin][indent][on|off]
sync .... :sy sync [ccomment|fromstart|$^{max}_{min}$lines=*n*]
clear info ..... :sy$_{ntax}$ clear :hi$_{ghlight}$ clear
highlight pattern ..... :$^{2}_{3}$mat$_{ch}$ [none|*group* /*pattern*/]

Windows

split, quit window ..... ^Ws :sp$_{lit}$ [*file*] ^Wq$_{\equiv:q[uit]}$
split mods :abo$_{veleft}$ *cmd* :bel$_{owright}$:to$_{pleft}$:bo$_{tright}$
vertical split ..... ^Wv :vert$_{ical}$ *cmd* :vs$_{plit}$ [*file*]
split & jump to alternate, file under cursor ..... ^W^ ^Wf
split & start new, vertical ..... ^Wn$_{\equiv:new}$ [*file*] :vne$_{w}$
close, +hide buffer ..... ^Wc$_{\equiv:cl[ose][!]}$:hid$_{e}$ [*cmd*]
make current window only one ..... ^Wo$_{\equiv:on[ly]}$
move cursor ↔↕ ..... ^Wh ^Wl ^Wj ^Wk
move cursor to window ↖, ↘ (wrapping) ^WW ^W^W$_{\equiv\wedge Ww}$
cursor ↖, ↘, |☜| (last active) ..... ^Wt ^Wb ^Wp
goto preview, exchange curr window with |☞ ^WP ^Wx
move window ↔↕ ..... ^WH ^WL ^WJ ^WK
rotate windows in column/row ↘, ↖ ..... ^Wr ^WR
size =, height +, -, $\mathcal{N}$ ..... ^W= ^W+ ^W- ^W_
width +, -, $\mathcal{N}$ ..... ^W> ^W< ^W|
cmd-line history window, search history ... q: q/ q?
exec *cmd* in each window ..... :windo$^{!}$ *cmd*
edit file in preview window .... :ped$^{!}_{it}$ [++*opt*] [+*cmd*] *file*
close preview window ..... ^Wz$_{\equiv:pc[lose]}$

Scrolling

▼^E ▲^Y .....↓^D ↑^U..... ▽^F △^B
curr line in window ⍗÷⍐ ..... zb$_{\equiv z-}$ zz$_{\equiv z.}$ zt$_{\equiv z↲}$
↔, ⇔ ..... zh zl zH zL

Diff mode

start, stop .....:diffs$_{plit}$ *f* :diffpa$_{tch}$ *f* :diffoff$^{!}$
jump ←, → to ‖ of change ..... [c]c
get (obtain), put change, update .. do dp :diffu$_{pdate}$

Quickfix

display $\mathcal{N}$th error, |☞, |☜|: cc$^{!}$ [$\mathcal{N}$] :cn$^{!}$:cN$^{!}_{\equiv:cp}$
1st err in $\mathcal{N}$th |☞ file, last in |☜| :cnf$^{!}_{ile}$:cNf$^{!}_{ile\equiv:cpf}$
|◀, ▶|; |☜|, |☞ errlist :cfir$^{!}_{st}$:cla$^{!}_{st}$:col$^{!}_{der}$:cnew$^{!}$
list, read from *f* ..... :cl [*from*, [*to*]] :cf [*f*]
open, close; toggle :cope[*height*] :ccl :cw [*height*]
jump with split; use *compiler* ... ^W↲ :comp$^{!}_{iler}$*compiler*
make, grep ..... :make [*args*] :gr$_{ep}$[*args*] :grepa$^{!}_{dd}$
built-in; recurse dirs patt .:vim$^{!}_{grep}$/*patt*/[g][j] *f*... **
the same for location list .:ll :lne$_{xt}$:lnf$_{ile}$... :lmak$_{e}$

Folding

fold manually (+$\mathcal{V}$); update&view curr .. zf$\mu_{\equiv:Rfold}$ zx
open, close / all; update ..... zo zc z0 zC zX
delete, switch state / all ..... zd za zD zA
fold more, reduce / all ..... zm zr zM zR
fold none, normal (restore), invert ..... zn zN zi
eliminate folds in window; view cursor line ..... zE zv
‖/‖ of fold, ↕ ..... [z]z zj zk

Messages

redir messages (>> to append) :redi$^{!}_{r}$ > *file* :redi$_{r}$ END
to reg,append to reg ..:redi @*c*> :redi @*C*>$_{\equiv:redi\ @c>>}$
view messages, last msg .....:mes$_{sages}$:echo errmsg
exec silently, verbosely ...:sil$^{!}_{ent}$ *cmd* :$\mathcal{N}$verb$_{ose}$ *cmd*
ask confirmations .....:confirm *cmd*
last page of previous cmd output ..... g<
↕ in the message screen ..... ↕$^{k}_{j}$ ↕$^{u}_{d}$ ↕$^{b}_{f}$ ↕$^{g}_{G}$ q

Edit text

replace $\mathcal{N}$ chars with *c* (+$\mathcal{B}$), keep layout ..... r*c* gr*c*
enter replace mode, keep layout; swap chars R*c* gR*c* xp
char, ▶| $\mathcal{N}$-1 lines (+$\mathcal{V}$); line, char .. cμ C cc$_{\equiv S}$ s
switch case for $\mathcal{N}$ chars ($\mathcal{V}$); $\mathcal{V}$: ↓,↑ ..... ~ u U
switch case, ↓, ↑ ..... g~μ guμ gUμ
rot13 encoding (+$\mathcal{V}$); add/subtract $\mathcal{N}$ ..... g?μ ^A ^X
indent →, ← ..... >μ >> <μ <<
format text, +keep cursor pos ..... gqμ gwμ
realign ..... :$\mathcal{R}$ce$_{nter}\mathcal{N}$:$\mathcal{R}$le$_{ft}\mathcal{N}$:$\mathcal{R}$ri$_{ght}\mathcal{N}$
adjust whitespaces ..... :$\mathcal{R}$ret$^{!}_{ab}$ [*tabstop*]
change list, $\mathcal{N}$th |☜|, |☞ pos .... :changes g; g,
insert ..... ↓i a↓ ‖↓I($\mathcal{B}$) A‖↓($\mathcal{B}$) →o →O
in column 1, last stopped insert (use '^ mark) .... gI gi
record macro into reg, append, stop ..... q*c* q*C* q
exec the contents of reg, repeat last ..... @*c* @@

Delete

$\mathcal{N}$chars ↦, ← (relative to cursor) ..... x$_{\equiv del}$ X
over μ (+$\mathcal{V}$), $\mathcal{N}$ lines, ▶|(and $\mathcal{N}$-1 whole lines) dμ dd D
join lines (+$\mathcal{V}$), w/o inserting spaces(+$\mathcal{V}$) ..... J gJ
$\mathcal{N}$ lines starting from $\mathcal{R}$ into register *c* ..... :$\mathcal{R}$d$_{elete}$ *c* [$\mathcal{N}$]

Copying and moving text

use reg *c* for |☞ del, yank, put; append to reg ... "*c* "*C*
yank into reg (+$\mathcal{V}$), $\mathcal{N}$ lines ..... yμ yy$_{\equiv Y}$
put reg ↓cursor↓/leave cursor after new text P p gP gp
like p P, but adjst indent to curr line .....]p [p
show contents of reg(s) ..... :reg$_{isters}$ [*c*]
special regs: last del or yank, small del ..... " -
filename, alternate, eval expression ..... % # =
clipboard(X11 selection), clipboard ..... * +
last search, cmdline ..... / :
last insert; yank; del/change ..... . 0 1...9

Undo/Redo

undo, undo line, redo; repeat change ... u U ^R $\mathcal{N}$.
jump to after $\mathcal{N}$th change, list changes .:u$_{ndo}\mathcal{N}$:undol$_{ist}$
go to older state, newer .:earlier$\mathcal{N}_{\equiv Ng-}$:later$\mathcal{N}_{\equiv Ng+}$

Insert mode

leave mode; for one cmd, keep cursor *esc*$_{\equiv\wedge C\equiv\wedge[}$ ^O ^\^O
move by words, whole pages ..... Shift⇆ S-PgUp/Dn
scroll ↑, ↓; insert char from ↓↑ .. ^X^E ^X^Y ^E ^Y
insert char literally, by decimal value ..... ^V*c* ^V*n*
repeat recent insert; + stop ..... ^A ^@
|☞, |☜| match; indent →,← ..... ^N ^P ^T ^D
del all indent, +restore in next line ..... 0^D ^^D
special completion: file, whole line, def ... ^X ^F ^L ^D
included file, tag, cmdline ..... ^I ^] ^V
word from dict, thesaurus, spelling .... ^K ^T ^S
keywords,user-defined, omni ... ^N ^P ^U ^O
during completion: |☞, |☜| match ..... ^N ^P
further ^X... will copy words following |☜| expansion
del word before, all entered chars in line ..... ^W ^U
line ↓ to column where insert started, ↑ ..... ^Gj ^Gk
start new undoable edit; toggle lmap ..... ^Gu ^^
reg contents, literally ..... ^R*c* ^R^R*c*
reg literally fixing indent, don't autoindent ^R^P*c* ^R^O*c*

Pattern searches

→,← with offset *o*; interrupt /*patt*[/*o*]↲ ?*patt*[?*o*]↲ ^C
repeat last, in opposite dir, →, ← ..... n N /↲ ?↲
*N*th →,← keyword/part under cursor ... * # g* g#
stop highlighting if `hlsearch` is set ..... :noh*lsearch*
subst *patt* with *str* in *n* lines .. :*R*s*ubst*/*patt*/*str*/ [*flags*] *n*
repeat with new flags ..... :*R*s*≡&&* *flags*
subst last used search pattern with last *subst* .... :*R*~ *flags*
repeat w/o flags on curr line; w/flags everywhere .. & g&
Ex cmd over matching (!≡not matching) :*R*g!*lobal*/*patt*/*cmd*
flags: keep ☜| flags, confirm, skip errors ..... & c e
all occurrences in the line; ignore case, not .... g i I
dry run; for :s w/o args-use last search pattern .... n r

Special chars in search patterns

‖/‖ of line (at ‖/‖ of pattern), any single character ^ $.
‖/‖ of line (at any position in pattern) ..... _^ _$
literal ^, $ in pattern; & in replace string ... \^ \$ \&
‖/‖ of word, whitespace, not ..... \< \> \s \S
char from set, range, not ..... [*abc*] [*a*-*z*] [^...]
digit, hex digit/not ..... \d \x \D \X
alphabetic, word char/not ..... \a \w \A \W
lower/uppercase, ignore; not ... \l \u \c \L \U \C
identifier char, keyword/excl digits ... \i \k \I \K
filename char, printable/excl digits ... \f \p \F \P
end-of-line, last subst; char class *x*+*EOL* .. \n ~ _*x*
esc, *tab*, *cr*, *bs*; in Visual area \e \t \r \b \%V
group into atom, not count as subexpr .. \(...\) \%(...\)
alternatives (or), branch (and) ..... *A*\|*T* *A*\&*T*
optional sequence (*A*, *A*t1, *A*t1t2...) ..... *A*\%[t1t2...]
matched group #1...9, whole match ..... \1...\9 \0≡&
count: 0+, 1+, 0...1, *k*...*n* .\{}≡* \+ \=≡\? \{*k*,*n*}
k, *k*+, 0...*n* matches ..... \{*k*} \{*k*,} \{,*n*}
non-greedy ... \{-} \{-*k*,*n*} \{-*k*} \{-*k*,} \{-,*n*}
atom *A* as a whole w/o backtracking ..... *A*\@>
subst: make upper/lower-case; until \e ... \u \l \U \L
subst: prev replacement, split line, <NL> .... ~ \r \n

Advanced zero width patterns

match at mark, before, after ..... \%'*m* \%<'*m* \%>'*m*
match at atom *A*, not match ..... *A*\@= *A*\@!
match at *T* if *A* matches, not ..... *A*\@<=*T* *A*\@<!*T*
set start/end of returned match ..... \zs \ze

Offsets after search command

n lines ↓,↑ in column 1 ..... *n*≡+*n* -*n*
n chars →,← from ‖/‖ of match .. e+*n* e-*n* s+*n* s-*n*
execute *searchcmd* next ..... ;*searchcmd*

Spell checking

☞|, ☜| misspelled word, stop at bad only]s [s]S [S
add a good/wrong word to spf; temp ..... zg zw zG zW
undo adding ..... zuw zug zuW zuG
correct spelling, repeat everywhere ..... z= :spellr*epall*

Autocommands

create, list ... :au [*grp*] *event patt*|<*buffer*> [nested] *cmd*
remove ... :au!*tocmd* [*grp* [*event*|* [*patt* [nested] *cmd*]]]
define group for next autocmds (!≡del) .. :aug!*roup**name*|END

Command-line mode

‖/‖ of command line, reg contents ..... ^B ^E ^R*c*
insert word under cursor, WORD ..... ^R^W ^R^A
insert filename, expanded with path ..... ^R^F ^R^P
delete word, chars till start of line ..... ^W ^U
insert char literally, by decimal val ..... ^V*c* ^V*n*
↕ history, by entered prefix . *Shift* ↕≡*PageUp*/*PageDn* ↕
cmd-line history, search history ..... :his*tory* :his/
open command-line window ..... ^F
recall search history by prefix; toggle lmap ..... /↕ ^^
copy modeless selection to clipboard ..... ^Y
repeat last command line ..... @:
exec reg as an *Ex* cmd, repeat last ..... :@*c* :@@

Completion

list all matches, insert all matches ..... ^D ^A
longest common, ☞|, ☜| ..... ^L ^N ^P
/&?+incsearch: add char from the end of curr match . ^L

Visual mode

start/stop char-wise, line-wise *V*, *B* ..... v V ^V
exchange cursor pos cross-wise, line-wise ..... o O
restore ☜| selection, *V*: exchange curr & prev ..... gv
extend *B* to *max* ..... $

Tags (tag *t* may be a /*pattern*)

jump to tag *t*, display stack ..... ? ta!*g* *t* :tags
jump to *N*th newer, older ..... :ta!*g* :po!*p*≡∧*T*
select from list, jump (if 1) or select ? ts!*elect* *t* ? tj!*ump* *t*
tag under cursor, select, jump ..... ^] g] g^]
split: tag under cursor, select, jump ^W^] ^Wg] ^Wg^]
→,←,|◀,▶| ..... :tn!*ext* :tp!*rev* :tf!*irst* :tl!*ast*
display unknown/(!≡all) include files ..... :che!*ckpath*

Keywords and macro definitions

display 1st line with keyword, all lines ..... [i [I
idem, but start from curr line .....]i]I
jump to 1st line with keyword, start from curr pos [^I]^I
keyword, macrodef with split .... ^Wi≡∧*W*∧*I* ^Wd≡∧*W*∧*D*
display 1st line in range with kwd :*R*is!*earch*[*N*] [/]*pattern*
split :isearch ..... :*R*isp!*lit* [*N*] [/]*pattern*
display all lines, jump ..... :*R*il!*ist* [/]*pattern* :ij!*ump*
for macrodefs commands, replace (i,I)⟼(d,D)
non-tag: local, global identifier declaration ..... gd gD
non-tag: ignoring matches in sibling blocks ... 1gd 1gD

Preview window

:pta!*g* *t* :pp!*op* :ptn!*ext* :pts!*elect* *t*...
tag under cursor, jump ..... ^W} ^Wg}
1st line in range with keyword .... :*R*ps!*earch*[*N*][/]*pattern*

Cscope

invocation format ..... ? cs*cope* *cmd* [*args*]
commands: connect database .. add *path* [-P*path*] [*flags*]
show connections, help ..... show help
reset, kill connections reset kill *num*|*partialname*
find symbol, definition, callee, caller ..... find s g d c
text, egrep, file, who includes ..... t e f i
search cscope database and *TAGS* files ..... :cst*ag* *tag*

Key mapping

l↦*r*, print ..... :*M*ma!*p* [<*buffer*>] *l r* :*M*ma!*p* [*l*]
disallow remapping of *r* ..... :*M*no!*remap* *l r*
remove, all ..... :*M*unm!*ap* *l* :*M*mapc!*lear*
*M*odes (no *M*≡Normal & *V*, !≡Insert & Command-line)
normal, insert, command line ..... n i c
visual, select, visual & select ..... x s v
operator-pending, lang arg+i+c ..... o l

Abbreviations

*M*odes: insert, command line (default both) ..... i c
create ..... :*M*ab*breviate* *l r* :*M*norea*bbrev* *l r*
print ..... :*M*ab*breviate* [*l*]
remove, all ..... :*M*una*bbreviate* *l* :*M*abc*lear*

Options

show modified, set ..... :se*t* :se*t* *opt*=*val*
non-termcap, termcap ..... :se*t* all :se*t* termcap
toggle option: on, off, invert .. :se*t* *opt* no*opt* inv*opt*
append, prepend, remove /+, ∗, − .:se*t* *opt*+=*val* ^= -=
show val, reset ..... :se*t* *opt*? :se*t* *opt*& :se*t* all&
show where the value was last set ... :verbose set *opt*?
set local, global value ..... :setl*ocal* :setg*lobal*
open options window ..... :opt*ions*

Some useful options

cd to current file directory ..... autochdir acd
autom. read file when changed outside ..... autoread ar
automatically write file if changed ..... autowrite aw
same, works with more cmds ..... autowriteall awa
read/write/edit file in binary mode ..... binary bin
prepend a Byte Order Mark to the file ..... bomb
action for buffer without window ..... bufhidden bh
whether show buffer in the buffer list ..... buflisted bl
the type of a buffer ..... buftype bt
list of directories searched with :cd ..... cdpath cd
number of columns in the display ..... columns co
how keyword completion (^P, ^N) works .. complete cpt
ask about unsaved/read-only files ..... confirm cf
use cscope for tag commands ..... cscopetag cst
:cstag order (0 - cscope 1st; 1...) cscopetagorder csto
highlight the current screen column .. cursorcolumn cuc
highlight the current screen line ..... cursorline cul
diff mode options ..... diffopt dip
encoding used internally ..... encoding enc
use spaces for *Tab* ..... expandtab et
file encoding for multi-byte text .... fileencoding fenc
autom. detected encodings ..... fileencodings fencs
file format used for file I/O ..... fileformat ff
autom. detected fileformats ..... fileformats ffs
type of file ..... filetype ft
close fold when cursor leaves ..... foldclose fcl
width of the folds column ..... foldcolumn fdc
close folds with level > *val* ..... foldlevel fdl
markers for foldmethod=marker ..... foldmarker fmr
kind of folding for the curr window ..... foldmethod fdm
min nr of lines for closed fold ..... foldminlines fml
maximum fold depth ..... foldnestmax fdn
don't unload buffer when it is abandoned .... hidden hid
highlight last search pattern matches ..... hlsearch hls
ignore case in search patterns ..... ignorecase ic
use :lmap or IM in Insert mode ..... iminsert imi
use :lmap or IM for search pattern ..... imsearch ims
incremental search highlight ..... incsearch is
name of a keyboard mapping ..... keymap kmp
chars for other language mode ..... langmap lmap
wrap long lines at a blank ..... linebreak lbr
number of lines in the display ..... lines
lisp indenting mode ..... lisp
words that influences lisp indenting ..... lispwords lw
show unprintable characters ..... list
unprintable chars to display in list mode listchars lcs
maximum amount of memory per buffer ..... maxmem mm
idem for all buffers altogether ..... maxmemtot mmt
changes to the text are not possible ..... modifiable ma
buffer has been modified ..... modified mod
print the line number in front of each line ..... number nu
nr of columns for line number ..... numberwidth nuw
function for omni completion ..... omnifunc ofu
allow pasting text ..... paste
key sequence to toggle paste ..... pastetoggle pt
list of directories searched with "gf" et.al. ..... path pa
disallow writing the buffer ..... readonly ro
scroll in window with others ..... scrollbind scb
round indent to multiple of sw ..... shiftround sr
number of spaces for indent step ..... shiftwidth sw
show full tag when completing ..... showfulltag sft
override ic of pattern contains upper case smartcase scs
use sw when inserting *Tab* ..... smarttab sta
number of spaces for *Tab* key ..... softtabstop sts
check spelling ..... spell
name of the word list file ..... spellfile spf
word list to check spelling ..... spelllang spl
put new splitting window below ..... splitbelow sb
put new splitting window on the right ... splitright spr
window action when switching buffer .... switchbuf swb
syntax to be loaded for current buffer ..... syntax syn
number of spaces for *Tab* ..... tabstop ts
whether push tags onto the tag stack ..... tagstack tgst
encoding used by the terminal ..... termencoding tenc
maximum width of text ..... textwidth tw
give informative messages ..... verbose vbs
log messages in a file ..... verbosefile vfile
modes in which to use virtual editing ... virtualedit ve
long lines wrap and continue on the next line ..... wrap
searches wrap around the end of the file .... wrapscan ws
writing to a file is allowed ..... write
! is not needed for override ..... writeany wa

N - an optional numeric argument (prefix for normal mode commands), *c* - character, *μ* - movement or text object, *n* - number, *f* - file, *V* - Visual mode, *B* - visual block mode, *R* - range, ‖ - start, ‖ - end, ①- first nonblank char, ? - command has variant with splitting the window (i.e. s after :), ☞| / ☜| - next/prev, WORD is a sequence of non-blank chars, separated with white space, w/o - without