awidegreen / **vim_cheatsheet.md**

Last active 12 hours ago • Report gist

Vim shortcuts

vim_cheatsheet.md

# Introduction

- C-a == Ctrl-a
- M-a == Alt-a

# General

```
:q        close
:w        write/saves
:wa[!]    write/save all windows [force]
:wq       write/save and close
:x        save and quit, same as wq
:q!       force close if file has changed and not save changes
```

```
v        Enter visual mode for selection of LINES
C-v      Enter visual mode for selection of BLOCKS
y        Yank/copy selected region
yy       Yank/copy entire line
"<reg>y  Yank/copy marked region into register <reg> (register from a-z)
c        Cut selection
p        Paste yanked content
"<reg>p  Paste yanked content in register <reg> (from a-z)
P        Paste yanked content BEFORE
```

```
u        Undo
C-r      Redo
```

```
:!<cmd>  Execute shell command <cmd>
C-z      send vim to background (fg brings it to front again)
```

### Windows

```
C-ws     Split current window horizontally (alternative :split)
C-wv     Split current window vertically (alternative :vsplit)
C-ww     Jump to the next window
C-wARROW Jump to window left/right/top/bottom (arrow keys) to the current
C-w#<    Shrink/resize current window from the right by # (default 1)
C-w#>    Increase/resize current window to the right by # (default 1)
```

### Entering insert mode

```
a        Append text after the cursor
A        Append text at the end of the line
i        Insert text before the cursor
I        Insert text before the first non-blank in the line
o        Begin a new line BELOW the cursor and insert text
O        Begin a new line ABOVE the cursor and insert text
s        Erase the current letter under the cursor, set insert-mode
S        Erase the whole line, set insert-mode
cc       Delete the current line, set insert-mode
```

```
cw       Delete word, set insert-mode
dd       Delete line under curser
```

### Recording

Vim has 26 registers (a-z), select the one you want to record in, see below. Exit Record mode with ESC

```
q[a-z]   Start recording, everything will be recorded including movement actions.
@[a-z]   Execute the recorded actions.
```

### Spell checking

See vimcast #19 as an introduction: http://vimcasts.org/episodes/spell-checking/

Assuming that you have the following in .vimrc:

```
nnoremap <silent> <leader>s :set spell!<cr>
```

```
<leader>s Toggle Spelling
]s       Next spelling mistake
[s       Previous spelling mistake
z=       Give Suggestions (prepent 1, use first suggestions automatically)
zg       Add misspelled to spellfile
zug      Remove word from spellfile
```

see http://vimdoc.sourceforge.net/htmldoc/spell.html

## Navigation

*essential*

```
h        cursor left
j        cursor down
l        cursor right
k        cursor up
```

```
H        Jump to TOP of screen
M        Jump to MIDDLE of screen
L        Jump to BOTTOM of screen
C-b      Move back one full screen (page up)
C-f      Move forward one full screen (page down)
C-d      Move forward 1/2 screen; half page down
C-u      Move back (up) 1/2 screen; half page up
```

```
w        jump by start of words (punctuation considered words)
e        jump to end of words (punctuation considered words)
b        jump backward by words (punctuation considered words)
0 (zero) start of line
^        first non-blank character of line
$        end of line
G        bottom of file
gg       top of file
```

*good to know*

```
E        jump to end of words (no punctuation)
W        jump by words (spaces separate words)
B        jump backward by words (no punctuation)
#G       goto line #
#gg      goto line #
```

# Search, jump

consider consulting `:help [` and `:help g`

```
*        search for word under cursor (forward) and highlight occurrence (see incsearch, hlsearch below)
%        jump from open/close ( / #if / ( / { to corresponding ) / #endif / }
[{       jump to start of current code block
]}       jump to end of current code block
gd       jump to var declaration (see incsearch, hlsearch below)
f<c>     Find char <c> from current cursor position -- forwards
F<c>     Find char <c> from current cursor position -- backwards
,        Repeat previous f<c> or F<c> in opposite direction
;        Repeat previous f<c> or F<c> in same direction
'.       jump back to last edited line.
g;       jump back to last edited position.
[m       jump to start of funtion body
[i       show first declartion/use of the word under cursor
[I       show all occurrences of word under cursor in current file
[/       cursor to N previous start of a C comment
```

### vimgrep and quickfix list

built-in grep, vimgrep uses vim's quickfix list. see vimcasts#44 for introduction: http://vimcasts.org/episodes/search-multiple-files-with-vimgrep/

```
:vimgrep /<regex>/g %        Search for <regex> with multiple occasions per line (g)
                             in current file (%)
:vimgrep /<C-r>// %          On the command line, <C-r>/ (that is: CTRL-R followed by /)
                             will insert the last search pattern.
:vimgrep /<a>/g <filelist>   Search in the given files (<filelist>)
:vimgrep /<a>/g *.cc         Search in all *.cc files current directory
:vimgrep /<a>/g **/*.cc      Search in all *.cc files in every sub-directory (recursively)
:vimgrep /<a>/g `find . -type f`
                             Search in all files that are returns by the backtick command.

:vim     short for :vimgrep

:cnext   Jump to next record/match in quickfix list
:cprev   Jump to previous record/match in quickfix list
```

Unimpaired plugin (https://github.com/tpope/vim-unimpaired) provides the following mappings:

```
[q       see :cprev
]q       see :cnext
[Q       see :cfirst
]Q       see :clast
```

see also: http://usevim.com/2012/08/24/vim101-quickfix/ and http://vimdoc.sourceforge.net/htmldoc/quickfix.html

## Marks

Mark a position in a buffer and jump back to it. see also http://vim.wikia.com/wiki/Using_marks

```
ma       set mark a at current cursor location
'a       jump to line of mark a (first non-blank character in line)
`a       jump to position (line and column) of mark a
d'a      delete from current line to line of mark a
d`a      delete from current cursor position to position of mark a
c'a      change text from current line to line of mark a
y`a      yank text to unnamed buffer from cursor to position of mark a
:marks   list all the current marks
:marks aB list marks a, B
```

(text is copied from link above)

# Editing

```
x        Delete char UNDER cursor
X        Delete char BEFORE cursor
#x       Delete the next # chars. starting from char under cursor
dw       Delete next word
dW       Delete UP TO the next word
d^       Delete up unto the beginning of the line
d$       Delete until end of the line
D        See d$, delete until end of the line
dd       Delete whole line
dib      Delete contents in parenthesis '(' ')' block (e.g. function args)
diB      Delete inner '{' '}' block
daB      Delete a '{' '}' block
das      Delete a senctence
diw      Delete word under cursor
df<c>    Delete until next occurence of <c> (char) found (including <c>) [in single line]
dt<c>    Delete until next occurence of <c> (char) found (without <c>!!!) [in single line]

ciw      Change word under cursor
ciB      Change inner '{' '}' block
cf<c>    See "df<c>" but change instead of delete
ct<c>    See "dt<c>" but change instead of delete

#J       Merge # number of lines together



gq       (in visual-mode) format selected text according to line-width
gqq      format current line according to line-width
#gqq     format next #-lines



C-n      Keyword completion
Tab      Keyword completion (SuperTab plugin)
r<c>     Replace char <c>
#r<c>    Replace follow # chars with <c>, : csock, cursor on s, 3re ceeek



:s/xxx/yyy/    Replace xxx with yyy at the first occurrence
:s/xxx/yyy/g   Replace xxx with yyy first occurrence, global (whole sentence)
:s/xxx/yyy/gc  Replace xxx with yyy global with confirm
:%s/xxx/yyy/g  Replace xxx with yyy global in the whole file



u        Convert selection (visual mode) to lowercase
U        Convert selection (visual mode) to uppercase



:g/^#/d  Delete all lines that begins with #
:g/^$/d  Delete all lines that are empty
```

## Misc

```
ga       Show ASCII of char under cursor
```

## Key sequences

### Replace a word in a number of occurrences with 'bar'; use word under cursor ( `*` or `/foo` )

`* cw bar ESC n .`

```
*     word under cursor 'foo'
cw    change word (enter insert mode)
bar   typed new word 'bar'
ESC   exit insert mode
n     next occurrence
.     repeat previous command
```

#### Insert 3 times "Help!": `Help! Help! Help!`

```
3i Help!_ ESC
```

#### Insert previously yanked text in line after current

```
oESCp
```

#### Search for selected text

```
<select> y / C-r0
```

```
<select> Select text in VISUAL mode (v)
y        Yank selection
/        Search for
C-r0     Press Ctrl-R and 0 to paste in
```

#### Comment out selection

```
C-v <select> # ESC ESC
```

```
C-v   Enter VISUAL block mode
<sel> Select lines
#     Comment char for programming language (perl, python, bash, etc)
ESC   Exit
ESC   Completes adding comment char for previous selected block
```

## Abbreviations

auto correction of frequently misspelled words.

```
:abbr Lunix Linux
:abbr accross across
:abbr hte the
```

## Configuration

- If you set the **incsearch** option, Vim will show the first match for the pattern, while you are still typing it. This quickly shows a typo in the pattern.
- If you set the **hlsearch** option, Vim will highlight all matches for the pattern with a yellow background. This gives a quick overview of where the search command will take you. In program code it can show where a variable is used. You don't even have to move the cursor to see the matches.

## NERD-tree

https://github.com/scrooloose/nerdtree/blob/master/doc/NERD_tree.txt

```
F3       Toogle NERD-Tree visible
```

## ctrlp.vim

https://github.com/kien/ctrlp.vim

```
C-p      Open ctrlp window (alternative :CtrlP)
:CtrlP d Open CtrlP with specific d = directory
```

```
C-b      Change mode: mru (most recent used) | buffers | files
```

## Formating

Use `gq` (see Editing section) for formating lines according to configured line-width. For C++ formating using clang-format see https://github.com/rhysd/vim-clang-format

## Links

### Cheat sheets

- http://www.worldtimzone.com/res/vi.html
- http://www.fprintf.net/vimCheatSheet.html
- https://wiki.archlinux.org/index.php/Vim
- http://www.fprintf.net/vimCheatSheet.html
- Yet Another Vim Cheat Sheet

### Articles

- Seven habits of effective text editing: http://www.moolenaar.net/habits.html
- Vim After 11 Years: http://statico.github.com/vim.html
- Coming Home to Vim: http://stevelosh.com/blog/2010/09/coming-home-to-vim

### tipps and tricks

- vimcasts.org Video-casts on vim
- usevim.com Plugin introductions and useful tipps
- vimregex.com Infos about vims regex engine
- Productive vim shortcuts http://stackoverflow.com/questions/1218390/what-is-your-most-productive-shortcut-with-vim
- 100 Vim commands every programmer should know http://www.catswhocode.com/blog/100-vim-commands-every-programmer-should-know
- VimGenius Interactive vim lesson, with some muscle learn potential
- Best of VimTips zzapper 15 Years of Vi + 8+ years of Vim and still learning
- http://rayninfo.co.uk/vimtips.html
- Use ag (silver searcher) as an indexer for Ctrl-P; and py-matcher for ctrl-p matching function: http://blog.patspam.com/2014/super-fast-ctrlp
- Command-T authors cheatsheet
- https://takac.github.io/2013/01/30/vim-grammar/

### Plugins

- NERDTree
- NERDCommenter
- Ctrl-P
- easytags
- unimpard
- supertab
- tagbar
- omnicomplete (C++)

### Themes

- zenburn
- tango

## Color column

- activate colorcolumn: http://stackoverflow.com/questions/1919028/how-to-show-vertical-line-to-wrap-the-line-in-vim
- set color: http://choorucode.wordpress.com/2011/07/29/vim-set-color-of-colorcolumn/

```
:set colorcolumn=81
highlight ColorColumn ctermbg=8
```

**omidfi** commented on 14 Aug 2015

This is great, specially the edit section has many new things for me!

**jhsilva** commented on 15 Sep 2016

thank you, \0/

**herlon214** commented on 31 Jan

Thanks!!

**fatagun** commented on 24 Mar

Awesome. Thanks.