



Vim cheatsheet

Vim is a very efficient text editor. This reference was made for Vim 8.0.

For shortcut notation, see :help key-notation.

Exiting

<code>:qa</code>	Close all files
<code>:qa!</code>	Close all files, abandon changes
<code>:w</code>	Save
<code>:wq</code> / <code>:x</code>	Save and close file
<code>:q</code>	Close file
<code>:q!</code>	Close file, abandon changes
<code>ZZ</code>	Save and quit
<code>ZQ</code>	Quit without checking changes

Exiting insert mode

<code>Esc</code> / <code><C-[></code>	Exit insert mode
<code><C-C></code>	Exit insert mode, and abort current command

Visual mode

<code>v</code>	Enter visual mode
<code>V</code>	Enter visual line mode
<code><C-V></code>	Enter visual block mode

Navigating

<code>h</code> <code>j</code> <code>k</code> <code>l</code>	Arrow keys
<code><C-U></code> / <code><C-D></code>	Half-page up/down
<code><C-B></code> / <code><C-F></code>	Page up/down
Words	
<code>b</code> / <code>w</code>	Previous/next word
<code>ge</code> / <code>e</code>	Previous/next end of word
Line	
<code>0</code> (zero)	Start of line
<code>^</code>	Start of line (after whitespace)
<code>\$</code>	End of line
Character	
<code>f c</code>	Go forward to character c
<code>F c</code>	Go backward to character c
Document	
<code>gg</code>	First line
<code>G</code>	Last line
<code>: n</code>	Go to line n
<code>nG</code>	Go to line n

Editing

<code>a</code>	Append
<code>A</code>	Append from end of line
<code>i</code>	Insert
<code>o</code>	Next line
<code>O</code>	Previous line
<code>s</code>	Delete char and insert
<code>S</code>	Delete line and insert
<code>C</code>	Delete until end of line and insert
<code>r</code>	Replace one character
<code>R</code>	Enter Replace mode
<code>u</code>	Undo changes
<code><C-R></code>	Redo changes

Clipboard

<code>x</code>	Delete character
<code>dd</code>	Delete line (Cut)
<code>yy</code>	Yank line (Copy)

Adobe Creative Cloud for Teams starting at \$33.99 per month.

ads via Carbon

In visual mode	
<code>d</code> / <code>x</code>	Delete selection
<code>s</code>	Replace selection
<code>y</code>	Yank selection (Copy)
See Operators for other things you can do.	

Window	
<code>zz</code>	Center this line
<code>zt</code>	Top this line
<code>zb</code>	Bottom this line
<code>H</code>	Move to top of screen
<code>M</code>	Move to middle of screen
<code>L</code>	Move to bottom of screen
Search	
<code>n</code>	Next matching search pattern
<code>N</code>	Previous match
<code>*</code>	Next whole word under cursor
<code>#</code>	Previous whole word under cursor
Tab pages	
<code>:tabedit [file]</code>	Edit file in a new tab
<code>:tabfind [file]</code>	Open file if exists in new tab
<code>:tabclose</code>	Close current tab
<code>:tabs</code>	List all tabs
<code>:tabfirst</code>	Go to first tab
<code>:tablast</code>	Go to last tab
<code>:tabn</code>	Go to next tab
<code>:tabp</code>	Go to previous tab

<code>p</code>	Paste
<code>P</code>	Paste before
<code>"*p</code> / <code>" + p</code>	Paste from system clipboard
<code>"*y</code> / <code>" + y</code>	Paste to system clipboard

Operators

Usage

Operators list

Examples

Operators let you operate in a range of text (defined by motion). These are performed in normal mode.

d	w
---	---

Operator	Motion
----------	--------

d	Delete
---	--------

y	Yank (copy)
---	-------------

c	Change (delete then insert)
---	-----------------------------

>	Indent right
---	--------------

<	Indent left
---	-------------

=	Autoindent
---	------------

g~	Swap case
----	-----------

gU	Uppercase
----	-----------

gu	Lowercase
----	-----------

!	Filter through external program
---	---------------------------------

See :help operator

Combine operators with motions to use them.

dd	(repeat the letter) Delete current line
----	---

dw	Delete to next word
----	---------------------

db	Delete to beginning of word
----	-----------------------------

2dd	Delete 2 lines
-----	----------------

dip	Delete a text object (inside paragraph)
-----	---

(in visual mode) d	Delete selection
--------------------	------------------

See: :help motion.txt

Text objects

Usage

Text objects let you operate (with an operator) in or around text blocks (objects).

v	i	p
---	---	---

Operator	[i]nside or [a]round	Text object
----------	----------------------	-------------

Diff

gvimdiff file1 file2 [file3]	See differences between files, in HMI
------------------------------	---------------------------------------

Text objects

p	Paragraph
---	-----------

w	Word
---	------

s	Sentence
---	----------

[({ <	A [], (), or {} block
---------	-----------------------

' " `	A quoted string
-------	-----------------

b	A block [(
---	------------

B	A block in [{
---	---------------

t	A XML tag block
---	-----------------

Examples

vip	Select paragraph
-----	------------------

vipipipip	Select more
-----------	-------------

yip	Yank inner paragraph
-----	----------------------

yap	Yank paragraph (including newline)
-----	------------------------------------

dip	Delete inner paragraph
-----	------------------------

cip	Change inner paragraph
-----	------------------------

See Operators for other things you can do.

Misc

Folds

<code>zo</code> / <code>zO</code>	Open
<code>zc</code> / <code>zC</code>	Close
<code>za</code> / <code>zA</code>	Toggle
<code>zv</code>	Open folds for this line
<code>zM</code>	Close all
<code>zR</code>	Open all
<code>zm</code>	Fold more (foldlevel += 1)
<code>zr</code>	Fold less (foldlevel -= 1)
<code>zx</code>	Update folds
Uppercase ones are recursive (eg, zO is open recursively).	

Windows

<code>z{height}<Cr></code>	Resize pane to {height} lines tall
----------------------------------	------------------------------------

Tags

<code>:tag Classname</code>	Jump to first definition of Classname
<code><C -]></code>	Jump to definition
<code>g]</code>	See all definitions
<code><C - T></code>	Go back to last tag
<code><C - O></code> <code><C - I></code>	Back/forward
<code>:tselect Classname</code>	Find definitions of Classname
<code>:tjump Classname</code>	Find definitions of Classname (auto-select 1st)

Navigation

<code>%</code>	Nearest/matching <code>{[()]}</code>
<code>[(</code> <code>[{</code> <code>[<</code>	Previous (<code>or { or <</code>
<code>])</code>	Next
<code>[m</code>	Previous method start
<code>[M</code>	Previous method end

Jumping

<code><C - O></code>	Go back to previous location
<code><C - I></code>	Go forward
<code>gf</code>	Go to file in cursor

Counters

<code><C - A></code>	Increment number
<code><C - X></code>	Decrement

Case

<code>~</code>	Toggle case (Case => cASE)
<code>gU</code>	Uppercase
<code>gu</code>	Lowercase
<code>gUU</code>	Uppercase current line (also gUgU)
<code>guu</code>	Lowercase current line (also gugU)
Do these in visual or normal mode.	

Editing with error

```
:cq  
:cquit
```

Works like `:qa`, but throws an error. Great for aborting Git commands.

Spelling checking

`:set spell spelllang=en_us` Turn on US English spell checking

`]s` Move to next misspelled word after the cursor

`[s` Move to previous misspelled word before the cursor

`z=` Suggest spellings for the word under/after the cursor

`zg` Add word to spell list

`zw` Mark word as bad/mispelling

`zu / C-X (Insert Mode)` Suggest words for bad word under cursor from spellfile

See `:help spell`

`ma` Mark this cursor position as a

``a` Jump to the cursor position a

`'a` Jump to the beginning of the line with position a

`d'a` Delete from current line to line of mark a

`d`a` Delete from current position to position of mark a

`c'a` Change text from current line to line of a

`y`a` Yank text from current position to position of a

`:marks` List all current marks

`:delm a` Delete mark a


`:delm a-d` Delete marks a, b, c, d

`:delm abc` Delete marks a, b, c

Also see

- [Vim cheatsheet](#) (vim.rotrr.com)
- [Vim documentation](#) (vimdoc.sourceforge.net)
- [Interactive Vim tutorial](#) (openvim.com)



►  **13 Comments** for this cheatsheet. [Write yours!](#)

devhints.io / Search 352+ cheatsheets

