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

`:qa`	
`:qa!`	
`:w`	
`:wq / :x`	
`:q`	
`:q!`	
`ZZ`	
`ZQ`	

Exiting insert mode

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

Visual mode

`v`	Enter visual mode
`V`	Enter visual line mode
`<C-V>`	Enter visual block mode
In visual mode	
`d / x`	Delete selection
`s`	Replace selection
`y`	Yank selection (Copy)

See Operators for other things you can do.

Navigating

`h j k l`	
`<C-U> / <C-D>`	
Words	
Shortcut	Description
ff	—
`b / w`	Previous/next word
`ge / e`	Previous/next end of word
Line	
`0` (zero)	
`^`	
`$`	
	Go forward to character
`Fc`	Go backward to character
Document	
`gg`	
	Center this lin
	Top this lin
	Move to top of scree
	Move to middle of scree
`L`	Move to bottom of scree
Tab pages	
`:tabedit [file]`	Edit file in a new ta
`:tabfind [file]`	Open file if exists in new ta
`:tabclose`	Close current ta
`:tabs`	List all tab
`:tabfirst`	Go to first ta
`:tablast`	Go to last ta
`:tabn`	Go to next ta
`:tabp`	Go to previous ta

Editing

`a`	
`i`	
`o`	
`O`	
`s`	
`S`	
`C`	
`r`	
`R`	
`u`	
`<C-R>`	

Clipboard

`x`	
`dd`	
`yy`	
`p`	
`P`	

Operators

Usage

Operators let you operate in a range of text (defined by motion). T

d	w
Operator	Motion

Operators list

d
y
c
>
<
g~
gU
gu
!

See `:help operator`

Examples

Combine operators with motions to us

dd
dw
db
2dd
dip
(in visual mode) d

See: `:help motion.txt`

Text objects

Usage

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

v	i
Operator	[i]nside or [a]round

Diff

`gvimdiff file1 file2 [file3]`	See differencies between files, in HMI

Text objects

p	
w	
s	
[({ <	
' " `	
b	
t	A XML tag bloc

Examples

vip
vipipipip
yip
yap
dip
cip

See Operators for other things you ca

Misc

Folds

zo / zO	
zc / zC	
za / zA	
zv	Open folds for t
zM	Close all
zR	Open all
zm	Fold more (foldlev
zr	Fold less (foldle

Navigation

[([{ [<
])
[m
[M

Jumping

<C-O>
<C-I>

`zx`	Upda

Uppercase ones are recursive (eg, `zO` is open recursively).

Windows

`z{height}<Cr>`	Resize pane to `{height}` lines tall

Case

`~`	Toggle case (Case => cASE)
`gU`	Uppercase
`gu`	Lowercase
`gUU`	Uppercase current line (also `gUgU`)
`guu`	Lowercase current line (also `gugu`)

Do these in visual or normal mode.

Misc

`.`	Repeat last command
`]p`	Paste under the current indentation level
`:ff=unix`	Convert Windows line endings to Unix line endings

Command line

`<C-R><C-W>`	Insert current word into the command line
`<C-R>"`	Paste from " register
`<C-X><C-F>`	Auto-completion of path in insert mode

Calculator

`<C-R>=128/2`	Shows the result of the division : '64'

Do this in insert mode.

Spell 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`

`gf`	

Counters

`<C-A>`	
`<C-X>`	

Tags

`:tag Classname`	Jump
`<C-I>`	
`ect Classname`	
`› Classname`	Find definitions

Marks

	Last pc
`ma`	
`` `a ``	
	Jump to the begi

Text alignment

```
:center [width]
:right [width]
:left
```

```
formatting
```

Exiting with an error

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

Also see

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