:h operator  :h navigation

# [operator][count][motion]

**d** delete/cut

**y** yank/copy

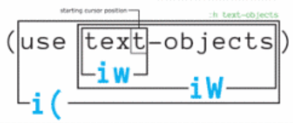
**c** change

| gU | ~ |
|---|---|
| make uppercase | swap case |
| < | = |
| shift left | indent |

Any motion can follow an operator. Marks and searches count as motions, too! `d/foo` will delete from the cursor to the next instance of "foo". `y3fi` will yank from the cursor to the 3rd "i" on the line after it. Counts can also come before operators: `5dd` will delete five lines.

| | |
|---|---|
| w | word |
| W | WORD |
| s | sentence |
| [, ] | [ ] block |
| (, ) | ( ) block |
| <, > | < > block |
| t | XML/HTML tag |
| {, } | { } block |
| ", ' | quoted string |

:h text-objects



:h up-down-motions

**gg** first line · **^b** up 1 page · **^u** up ½ page · **k** up 1 line

| | ts | sw | sts | et |
|---|---|---|---|---|
| use spaces only | *n* | *n* | *n* | on |
| use tabs only | *n* | *n* | 0 | off |

Set *n* to desired tab width (default 8)

| | | |
|---|---|---|
| tabstop | ts | Columns per tabstop |
| shiftwidth | sw | Columns per `<<` |
| softtabstop | sts | Spaces per tab |
| expandtab | et | `<Tab>` inserts spaces |

**MIXING TABS AND SPACES IS RIGHT OUT.**
(that means don't do it.)

**:retab** Replace all tabs with spaces according to current tabstop setting

| | | |
|---|---|---|
| fileformat | ff | Try changing this if your line-endings are messed up |
| list | | Display whitespace visibly according to listchars |

:h left-right-motions

beginning of line **0** · first non-blank character **^** · previous WORD **B** · previous word **b** · previous character **h**

next character **l** · end of word **e** · beginning of next word **w** · end of WORD **E** · beginning of next WORD **W** · end of line **$**

**j** down 1 line · **^d** down ½ page · **^f** down 1 page · **G** last line

:h word-motions

7 words

http://www.vimcheatsheet.com

1 WORD

:h pattern-searches

## SEARCHING

| Prev | Next | Forward | Backward | Matches |
|---|---|---|---|---|
| N | n | /*foo* | ?*foo* | *foo* |
| | | * | # | word under cursor |
| ; | , | t*x* | T*x* | upto *x* |
| | | f*x* | F*x* | find *x* |

:h mark-motions

| | | | | | |
|---|---|---|---|---|---|
| m*m* | set mark *m* (a-z) in file | m*M* | set mark *M* (A-Z) across files | '[ | jump to first char of just-changed text |
| '*m* | jump to first char of line containing *m* | \`*m* | jump to exact character of *m* | '' | jump back to last jump |

Pass a directory to the `:edit` command to open a directory explorer. Instructions for usage are at the top of the screen.

## ENTERING INSERT MODE

beginning of line **I** · before cursor **i** · after cursor **a** · end of line **A**

previous line **O** · next line **o** · substitute character **s** · substitute line **S** · line from cursor **C**

## ENTERING VISUAL (SELECT) MODE

**v** The most basic type. Use Visual mode to select characters within a line.

**V** Useful for moving chunks of a program around the file. Use Visual Line mode to select one or more lines.

**^v** Great for working with tables made of text, or anything that happens to be conveniently aligned. Visual Block mode can be used to select boxes across lines.

| | | | |
|---|---|---|---|
| switch cursor to start/end **o** :h v_o | re-select previous area **gv** :h gv | prepend to each Visual block line **I** :h v_b_I | jump to start of prior area **'<** :h '< |

| | | | |
|---|---|---|---|
| **ZZ** | Write current file, if modified, and quit | **ZQ** | Quit without checking for changes (like `:q!`) |

| | |
|---|---|
| **:write** | Write current file |
| **:wq** | Write current file and quit |

Use `:scriptnames` to list all files sourced during initialization.

**:syntax** Enable and configure syntax highlighting
Use `:sy sync fromstart` to redraw broken highlights

:h quickfix

**:make** Run a compiler and enter quickfix mode

| | | | |
|---|---|---|---|
| **:!** | Execute external shell command | **!** | Filter motion with shell command |

Use `:earlier` and `:later` to quickly jump backward and forward in a file's history.

**:read** Read external program output into current file

| | | | | | |
|---|---|---|---|---|---|
| p | paste after cursor | P | paste before cursor | ^[ | return to Normal mode |
| u | undo | ^r | redo | . | repeat |
| gf | find file under cursor in path and jump to it | dd | delete current line | yy | yank current line |
| x | delete character after cursor | % | jump to matching paren | r | replace char under cursor |
| *n*G | jump to line *n* | ^o | jump back | ^i | jump forward |
| zz | center screen on cursor | zt | align top of screen with cursor | zb | align bottom of screen with cursor |
| == | auto-indent current line | << | shift current line left by shiftwidth | >> | shift current line right by shiftwidth |

file-searching · jump-motions

Using `^[` to return to Normal mode lets you keep your fingers on the home row. It's even easier if you map Caps Lock to Control!

:h insert.txt

## COOL INSERT MODE STUFF

| | | | |
|---|---|---|---|
| ^w | delete word before cursor | ^u | delete line before cursor |
| ^r*r* | insert the contents of register *r* | ^r= | use the expression register (try `^r=5+10`) |
| ^t | increase line indent by shiftwidth | ^d | decrease line indent by shiftwidth |
| ^x^l | line completion | ^n | find next completion suggestion according to complete |

:h cmdline.txt

## COMMAND-LINE MODE ONLY

| | | |
|---|---|---|
| edit using Normal mode **^f** cmdwin | insert word under cursor **^r^w** cmdline-editing | completion suggestions **^d** cmdline-completion |

Put `cnoremap %% <C-R>=expand('%:h').'/'<CR>` in your `.vimrc` so you can type `%%` in Command-line mode to refer to the directory of the current file, regardless of `pwd`.

Supply `%` as a range to the `:substitute` command to run it on every line in the file.

```
:%s/Scribbl/Design/                "Scribbled" -> "Designed"
```

Specify the "`g`" flag to apply the substitution to *every* match on a line.

```
:s/[dla]//g                        "badly" -> "by"                 :h s_flags, :h /[]
```

Vim supports many regular expression features.

```
:s/..k/ax/                         "Mook" -> "Max"                 :h usr_27, :h /.
```

Use `\_.` instead of `.` if you want to search across multiple lines.

```
:%s/heat\_.*Bungle/anto/           "Cheatsheet\nBungler" -> "Cantor"   :h /\_.
```

Special escapes can be used to change the case of substitutions.

```
:s_\(f..\)_\U\1\E_                 "foobar" -> "FOObar"            :h sub-replace-special
```

Use `:global` to perform a command on matching lines.

```
:g/foobar/delete                   Delete all lines containing "foobar"
```

If your pattern contains slashes, just use a different character as your delimiter.

```
:s_Data/Lore_Brent Spiner_
```

Use `\=` to evaluate expressions with replacement groups.

```
:s_\d_\=submatch(0) + 1_g          "10 25" -> "21 36"              :h sub-replace-\=
```

:help

| | |
|---|---|
| :h *cmd* | Normal mode *cmd* help |
| :h i_*cmd* | Insert mode *cmd* help |
| :h v_*cmd* | Visual mode *cmd* help |
| :h c_*cmd* | Command-line editing *cmd* help |
| :h :*cmd* | Command-line *cmd* help |
| :h '*option*' | *Option* help |
| :helpgrep | Search through all help docs! |

# vim

:h tags-and-searches

| | |
|---|---|
| ^] | Jump to tag under cursor, including [tags] in help files |
| ^t | Jump back up the tag-list |
| g^] | Jump to tag if it's the only match; else list matching tags |

:h keycodes

| | | | |
|---|---|---|---|
| <CR> | ^m | \r | Enter |
| <Tab> | ^i | \t | Tab |
| <C-*n*> | ^*n* | | Ctrl-*n* |
| <M-*n*> | | | Alt-*n* |
| <Esc> | ^[ | | Escape |
| <BS> | ^h | \b | Backspace |
| <Del> | | | Delete |

:h options

| | |
|---|---|
| :set *opt*? | View current value of *opt* |
| :set no*opt* | Turn off flag *opt* |
| :set *opt* | Turn on flag *opt* |
| :set *opt*=*val* | Overwrite value of *opt* |
| :set *opt*+=*val* | Append to value of *opt* |
| :echo &*opt* | Access *opt* as a variable |

| | | |
|---|---|---|
| hidden | hid | Lets you switch buffers without saving |
| laststatus | ls | Show status line never (0), always (2) or with 2+ windows (1) |
| hlsearch | hls | Highlight search matches. Also see 'highlight' |
| number | nu | Show line numbers |
| showcmd | sc | Show commands as you type them |
| ruler | ru | Show line and column number of the cursor |
| backspace | bs | Set to '2' to make backspace work like sane editors |
| wrap | | Control line wrapping |
| background | bg | Set to 'dark' if you have a dark color scheme |

Use `a` instead of `i` when beginning text-object motions to include delimiters or surrounding whitespace. For example, `di(` will change "(foo)" into "()", but `da(` will delete the parentheses as well.

:h buffers

| | |
|---|---|
| :ls | List all open files |
| :b *path* | Jump to unique file matching *path*. Use **<Tab>** to scroll through available completions! |
| :b*n* | Jump to file *n*, number from first column of :ls |
| :bnext | Jump to next file |
| :bprev | Jump to previous file |
| :bdelete | Remove file from the buffer list |
| :edit | Open a file for editing |
| :enew | Open a blank new file for editing |

:h windows

| | |
|---|---|
| :split | Split current window horizontally |
| :vsplit | Split current window vertically |
| ^w hjkl | Move cursor to window left, below, above or to the right of the current window |
| ^w HJKL | Move current window to left, bottom, top, or right of screen |
| ^w r | Rotate windows clockwise |
| ^w +-<> | Increase/decrease current window height/width |
| ^w T | Move current window to a new tab |
| :only | Close all windows except current window |
| :bufdo | Execute a command in each open file |

## REGISTERS are CLIPBOARDS

All commands that delete, copy, or paste text use registers. To change which register is used by a command, type the register before the command. The default register is called "the unnamed register", and it is invoked with a pair of double-quotes (`""`). Typing `dd` or `yy` is the same as typing `""dd` or `""yy`. Think of the first `"` as a short way of saying "register", so `""` is pronounced "register `"`", and `"a`, "register `a`".

:h registers

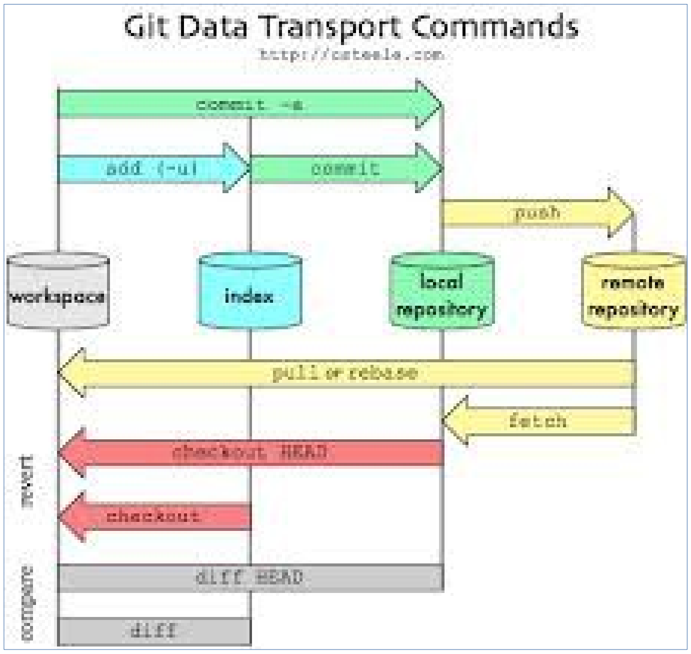
| | | |
|---|---|---|
| :registers | | View all current registers |
| :echo @*r* | | Access register *r* as a variable |
| "/ | Last search pattern register | Contains the last pattern you searched for |
| "_ | The black hole register | Use this to delete without clobbering any register (`"_dd`) |
| "0 | Last yank register | Contains the last text you yanked |
| "1 | Last big delete register | Contains the last line(s) you deleted |
| "2-"9 | Big delete register stack | Every time `"1` is written to, its content is pushed to `"2`, then `"2` to `"3`, and so on |
| "- | Small delete register | Contains the last text you deleted within a single line |
| "+ | System clipboard | If the OS integration gods smile upon you, this register reads and writes to your system clipboard. |
| "a-"z | Named registers | 26 registers for you to play with |
| "A-"Z | Append registers | Using upper-case to refer to a register will append to it rather than overwrite it |
| q*r* | Record | Record into register *r*. Stop recording by hitting `q` again |
| @*r* | Playback | Execute the contents of register *r* |
| @@ | Repeat last playback | Repeat the last @*r*, this is particularly useful with a count |

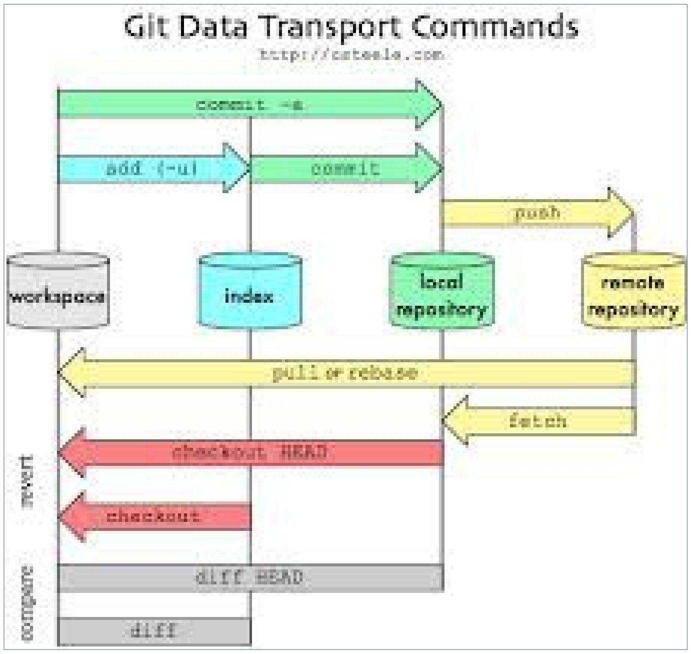
Use `:map` to view all current custom key mappings. Read :h map-which-keys for a guide on which keys are best for your own custom mappings. Get used to Vim's help system - it's a fantastic resource!

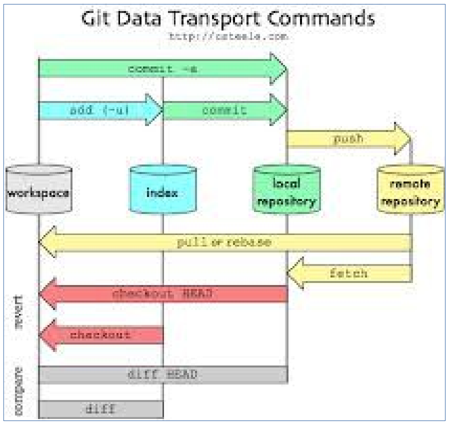
vim one-liner used to sort the list of names by length:

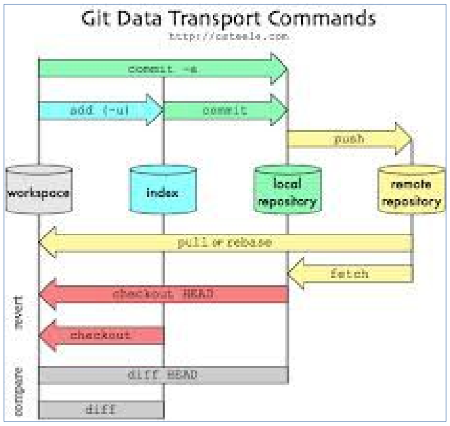
```
:exe 'g/^/let @x = len(getline(".")) | normal "xPa ' | sort n | :g/normal dw
```

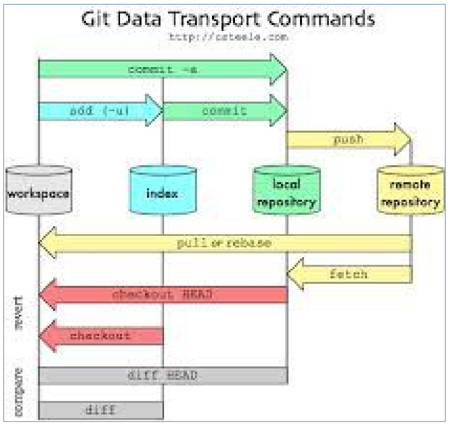
1st Edition
funded by...
RML
Adam
jdiems
Lin Qiu
#webjoe
grantok
storzgie
David O.
Digininja
Dave Remy
Ralph Bean
Arve Laken
Rajat Dhal
Paul-Kenji
Mark Dixon
Neil Munro
Kyle Verma
Guy Bensky
Ross Timson
Abdul Qabiz
Daniel Beck
Hazel Smith
Kelly Ralla
Charles Chen
Pivotal Labs
Eric Sporkin
M. Adam Price
Daniel Hahler
Rauli Silvola
Lawrence Kemp
Bill LaPierre
Brandon Tyree
Laurence Emms
Infonautic.de
#wandersosthus
Damon Jablons
Travis McHenry
Ganesh Suguran
Rouven Herrier
Digital Gnomes
Sam Napolitano
Jens O. Meiert
Thomas Nitsche
Philip Ratzsch
Hurricane Labs
Michael Schmidt
Matthew Datcher
Arya Reais-Parsi
David MacFarlane
Christer Edvartsen
St. Louis Vim Geeks
Haraldur Tristan Gunnarsson
and others!

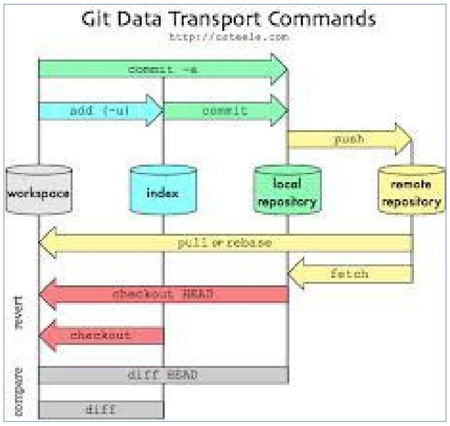

Git Data Transport Commands
http://osteele.com
commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
checkout
revert
diff HEAD
diff
compare

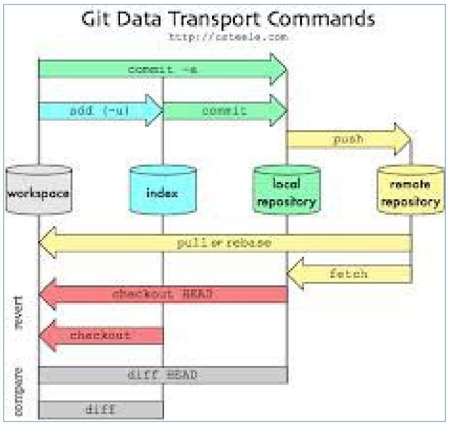


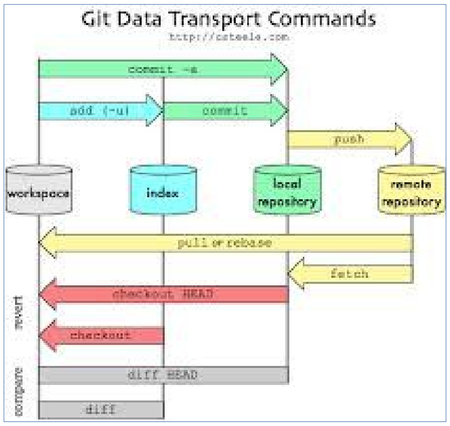


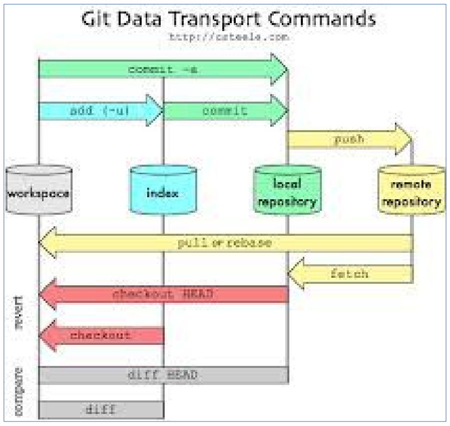


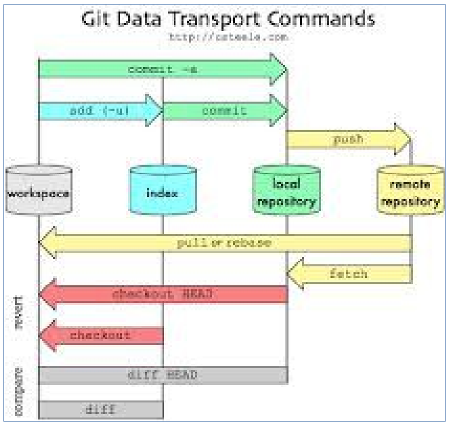


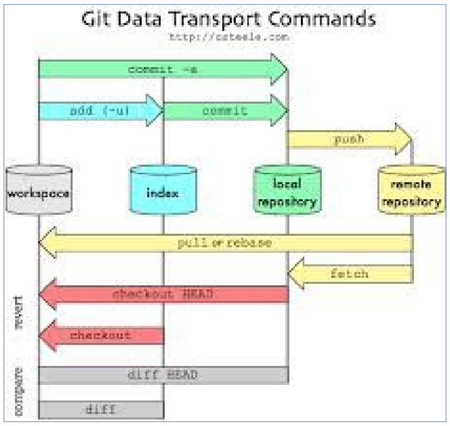




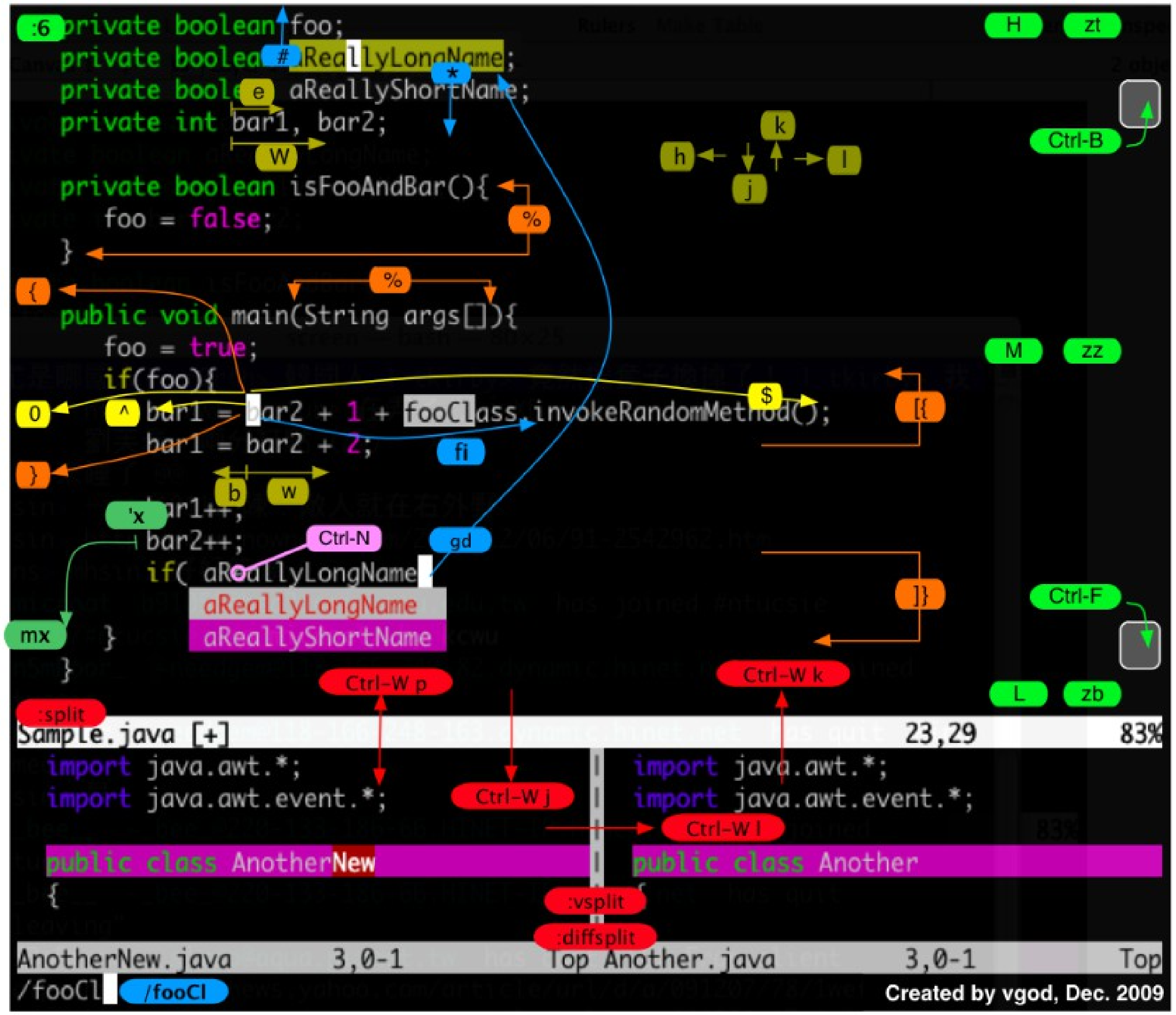








# Vim Visual Cheat Sheet



```
private boolean foo;
private boolean aReallyLongName;
private boolean aReallyShortName;
private int bar1, bar2;

private boolean isFooAndBar(){
    foo = false;
}

public void main(String args[]){
    foo = true;
    if(foo){
        bar1 = bar2 + 1 + fooClass.invokeRandomMethod();
        bar1 = bar2 + 2;
        bar1++;
        bar2++;
        if( aReallyLongName
            aReallyLongName
            aReallyShortName
    }
}
Sample.java [+]                                  23,29              83%
import java.awt.*;                      | import java.awt.*;
import java.awt.event.*;                | import java.awt.event.*;

public class AnotherNew                 | public class Another
{                                       |
AnotherNew.java        3,0-1        Top Another.java        3,0-1        Top
/fooCl
```

Created by vgod, Dec. 2009

## Movement/Range

### Character

| | | | |
|---|---|---|---|
| h | j | k | l |
| ← | ↓ | ↑ | → |

### word, WORD(all non-blank ch)

| | | |
|---|---|---|
| w | b | next/prev word |
| W | B | next/prev WORD |
| e | E | end of word/WORD |

### Line

| | | |
|---|---|---|
| 0 | $ | begin/end of line |
| ^ | | begin (non-blank) of line |

### Paragraph, Block

| | | |
|---|---|---|
| { | } | prev/ next paragraph |
| [[ | ]] | begin/end of block |
| % | | matching parenthesis |

### Window, File

| | | | |
|---|---|---|---|
| H | top of win | zt | scroll to top |
| M | mid of win | zz | scroll to middle |
| L | btm of win | zb | scroll to bottom |
| C-B | C-F | prev/next page | |
| gg | G | begin/end of file | |
| mx | 'x | mark/jump to x | |

### Search

| | | |
|---|---|---|
| * | # | find current word backward/forward |
| fx | | to character x to right |
| gd | | to definition of current word |
| /xxx | | search xxx |
| n | N | next/prev search result |

## Mode Commands

| | |
|---|---|
| ESC C-[ | enter normal mode |
| v | enter visual mode |
| V | enter visual line mode |
| C-v | enter visual block mode |
| i | enter insert mode |
| R | enter replace mode |
| a | append |
| A | append at end of line |

## General Commands

| | |
|---|---|
| y | yank/copy (range) |
| d | delete/cut (range) |
| c | modify (range) |
| x | delete/cut (character) |
| D | delete to end of line |
| C | modify to end of line |
| p | paste after cursor |
| J | join lines |
| r | replace (character) |
| > | indent |
| < | indent leftward |
| . | redo |
| u | undo |

## EX Commands

| | |
|---|---|
| :w | save(:wq save and quit) |
| :q | quit(:q! quit anyway) |
| :e x | edit file x |
| :n | new window |
| :h | vim help |
| :xx | jump to line #xx |

## Auto-completion [insert mode]

| | |
|---|---|
| C-N C-P | auto-complete next/prev keyword |
| C-X C-F | auto-complete file name |

## Split window

| | |
|---|---|
| :vsp :sp | vertically/horizontally split |
| :diffs | split and diff |
| C-W p | to last accessed window |
| C-W w | to next window |

**Esc** Normal

*Revision 2.0*
*Sept. 11, 2011*

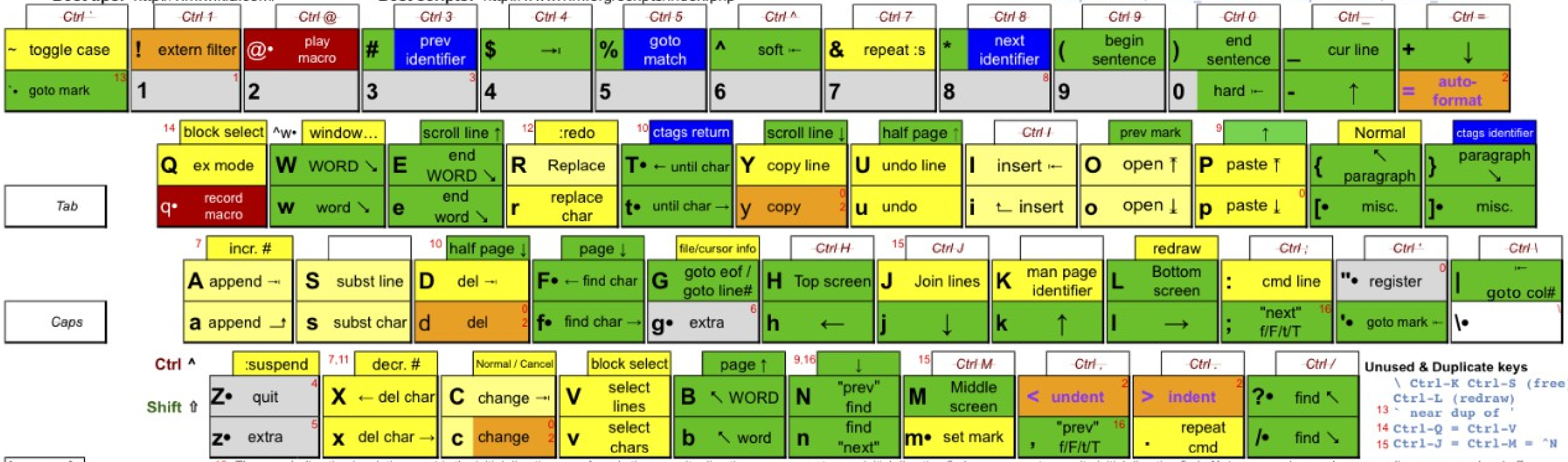
Vim 7.3+
`:version`

# *Vim Cheat Sheet for Programmers*

*http://michael.PeopleOfHonorOnly.com/vim/*

***HOW-TO** make Vim not suck Out of the Box:* ***:help statusline*** :set nocompatible ruler laststatus=2 showcmd showmode number ***Search*** :set incsearch ignorecase smartcase hlsearch ***Remove useless splash screen*** :set shortmess+=I

**Best tips:** http://vim.wikia.com/ **Best scripts:** http://www.vim.org/scripts/index.php :map <F9> :e $HOME/_vimrc<CR> :map <F6> :so $HOME/_vimrc<CR>



~~Ctrl `~~ ~~Ctrl 1~~ ~~Ctrl @~~ ~~Ctrl 3~~ ~~Ctrl 4~~ ~~Ctrl 5~~ ~~Ctrl ^~~ ~~Ctrl 7~~ ~~Ctrl 8~~ ~~Ctrl 9~~ ~~Ctrl 0~~ ~~Ctrl _~~ ~~Ctrl =~~

~ toggle case | ` • goto mark 13
! extern filter | 1 1
@• play macro | 2
# prev identifier | 3 3
$ →| | 4
% goto match | 5
^ soft |← | 6
& repeat :s | 7
* next identifier | 8 8
( begin sentence | 9
) end sentence | 0 hard |←
_ cur line | - ↑
+ ↓ | = auto-format 2

Tab

14 block select ^w• window... scroll line ↑ 12 :redo 10 ctags return scroll line ↓ half page ↑ ~~Ctrl I~~ prev mark 9 ↑ Normal ctags identifier

Q ex mode | q• record macro
W WORD ↘ | w word ↘
E end WORD ↘ | e end word ↘
R Replace | r replace char
T• ← until char | t• until char →
Y copy line | y copy 0 2
U undo line | u undo
I insert |← | i ⌞ insert
O open ↥ | o open ↧
P paste ↥ | p paste ↧ 0
{ ↖ paragraph | [• misc.
} paragraph ↘ | ]• misc.

Caps

7 incr. # 10 half page ↓ page ↓ file/cursor info ~~Ctrl H~~ 15 ~~Ctrl J~~ redraw ~~Ctrl ;~~ ~~Ctrl '~~ ~~Ctrl \~~

A append → | a append ⌟
S subst line | s subst char
D del → | d del 0 2
F• ← find char | f• find char →
G goto eof / goto line# | g• extra 6
H Top screen | h ←
J Join lines | j ↓
K man page identifier | k ↑
L Bottom screen | l →
: cmd line | ; "next" f/F/t/T 16
"• register | '• goto mark ← 0
| goto col# | \•

Ctrl ^
Shift ⇧

:suspend 7,11 decr. # Normal / Cancel block select page ↑ 9,16 ↓ 15 ~~Ctrl M~~ ~~Ctrl ,~~ ~~Ctrl .~~ ~~Ctrl /~~

Z• quit 4 | z• extra 5
X ← del char | x del char →
C change → | c change 0 2
V select lines | v select chars
B ↖ WORD | b ↖ word
N "prev" find | n find "next"
M Middle screen | m• set mark
< undent 2 | , "prev" f/F/t/T 16
> indent 2 | . repeat cmd
?• find ↖ | /• find ↘

**Unused & Duplicate keys**
\ Ctrl-K Ctrl-S (free)
Ctrl-L (redraw)
13 ` near dup of '
14 Ctrl-Q = Ctrl-V
15 Ctrl-J = Ctrl-M = ^N

16 The search direction is relative; **next** is the initial direction, **previous** is the opposite direction. n ; repeat same initial direction find. N , repeat opposite initial direction find. **Note**: ; , only searches cursor line, n N searches buffer.

## Legend:

| | |
|---|---|
| Macro | Register name (0-9a-zA-Z) required |
| Op | Motion req.; act between cursor & dst |
| Cmd | Command |
| Ins | Command and enter insert mode |
| Move | Moves cursor or defines range for op |
| Find | Search (↖ = reverse, ↘ = forward) |
| tag | ctags / diffs / folding |
| Code | Code formatting, whitespace, etc. |
| Extra | Extended functionality; req. extra chars |
| • | Char arg req. g z Z ^w ' " ` ... |

## Modes

`:help modes`

| | | |
|---|---|---|
| n | Normal | Esc ^[ ^c |
| i | Insert | a i r s |
| v | Visual | v V ^v ^q |
| o | Op pending | c d y < > |
| c | Command Line | : / ? ! |

**word** Foo ( src , ␢ dst , ␢ len );
**Note**: There is no whitespace in-between 'Foo(src,' but before/after 'dst,'
**WORD** Foo ( src , ␢ dst , ␢ len );

## Startup

| | |
|---|---|
| `vim <filename> +123` | **goto line 123** |
| `vim <file> .. -t Foo` | **edit at tag 'Foo'** |
| `vim <file> .. -c "/Foo"` | **cmd: find 'Foo' & edit** |
| GUI `vim -g` or `gvim` | start GUI ver. |
| GUI Linux | `:set guifont=ProggyTinyTT\ 12` |
| GUI OSX | `:set guifont=ProggyTiny\:h11` |
| diff | `gvimdiff <file1> <file2> [<file3>]` |

bug **Broken Keys** ~~Ctrl-I = Tab, Ctrl-[ = ESC~~
Vim is *still* **unable** to map certain keys for your own use...
§ **Caps, Ctrl-1, Ctrl-Shift-1, Ctrl-I, Ctrl-\, etc.**

0 ***See***: `src/ops.c -c "/valid_yank_reg"` *for "• reg. names*
6 ***See***: `src/normal.c -c "/nv_cmds"` *for g• extra cmds*
11 ***See***: `src/edit.c -c "/ctrl_x_msgs"` *for ^x• insert cmds*

## :help cmdline

`:r file` insert file

| | | | |
|---|---|---|---|
| `:w` | save | `:gui` | switch to GUI |
| `:q` | quit | `:q!` | quit w/o save |
| `:e <file>` | edit file | | |
| `:source %` | **exec cmds in cur file** | | |
| `:exec '...'` | **do cmd** | | |

## :help movement

| | | | | |
|---|---|---|---|---|
| soft | `^` | ⇤ **Start of Line** | 1st non-whitespace | |
| hard | `0` | ⇤ **Start of Line** | column 0 | |
| | `$` | ⇥ **End of Line** | | |
| | `\|` | move col 0 | `#\|` | **move col #** |
| | `^b` | page ↑ | `^f` | page ↓ |
| | `^u` | ½ page ↑ | `^d` | ½ page ↓ |
| | `^e` | scroll line ↑ | `^y` | scroll line ↓ |
| | `1g` | start of file | `0g` | end of file |
| | `#g` | **goto line #** | `G` | **end of file** |
| | `[[` | **begin this func {** | | |
| | `]]` | **begin next func {** | | |

:set matchpairs=(:),{:},[:],<:>,?:\:
`%` **goto matching { } < > [ ]**

## :help range

| | |
|---|---|
| `:s/Foo/Bar` | find Foo replace w/ Bar |
| `:s/Foo/Bar/g` | ...all instances on line |
| `:%s/Foo/Bar` | **apply to whole file** |
| `.,.+#` | **cur line, cur line + # lines** |
| `$` | **last line** |
| `'<` | start of select |
| `'>` | end of select |

## :help tags

| | |
|---|---|
| `:ts` | list active tags |
| `^]` | **jump to tag under cursor** |
| `^t` | **restore cursor before tag jump** |
| `^p` | **complete word** |
| `:ta Foo` | manual jump to tag 'Foo' |

## :help diff

| | | | |
|---|---|---|---|
| `[c` | prev diff | `:hi DiffAdd` | `guifg=#rrggbb` |
| `]c` | next diff | `:hi DiffChange` | `guibg=#rrggbb` |
| `:diffupdate` | | `:hi DiffText` | `gui=none` |
| | resync | `:hi DiffDelete` | |

## :help folding

| | |
|---|---|
| `zR` | fold remove |
| `zo` | fold open |
| `zc` | fold close |
| `zi` | invert all |
| `zr` | fold reduce |
| `zm` | fold more |

## :help changes

`:changes`
`g;` older change
`g,` newer change

## :help syntax

`:syntax enable`
`:set filetype=`
*c cpp sh make perl python*
Note: chose only ONE type!

## :help recording

| | |
|---|---|
| `q•` | start recording |
| `@•` | playback |
| `q` | stop recording |
| `@@` | repeat |

## convert <eol>

`:set fileformat=`
*unix* or *dos* or *mac*
then :w to convert

`\ :map \ :Explore<CR>` manually type <,C,R,>

| | | | | |
|---|---|---|---|---|
| §0 | `"•` | before del/copy/paste to use register | | |
| | `"+x` | cut to system clipboard reg. '+' | | |
| | `"+gP` | paste from system clipboard | | |
| 1 | | Number before any action repeats it | | |
| | `2p` | paste twice | `3.` | repeat thrice |
| 2 | | Repeat op to act on current line | | |
| | `yy` | copy line | `dd` | del line |
| | `<<` | undent line | `>>` | indent line |
| 3 | `#` | highlight words under cursor | | |
| 4 | `ZZ` | save & quit | `ZQ` | quit w/o save |
| 5 | `zz` | **center cursor line in window** | | |
| | `zh` | **scroll left** | `zl` | **scroll right** |
| | `zt` | scroll top | `zb` | scroll bottom |
| §6 | `gg` | top of file | | |
| | `gf` | **open file under cursor** | | |
| 7 | `^a` | incr # under cursor (Dec / Hex) | | |
| | `^x` | decr # under cursor (Dec / Hex) | | |
| 8 | `*` | start a "new" search | | |

## Insert mode

| | | | | |
|---|---|---|---|---|
| 9 | `^p` | prev auto-complete | `^n` | next |
| 10 | `^d` | undent | `^t` | indent |
| §11 | `^x•` `^f` | filename completion | | |
| | `^s` | spelling | `:set spell!` | |
| | `^k` | dictionary | `]s` | next bad |
| | `^t` | thesaurus | `:help spell` | |
| 12 | `^r•` | paste register 0-9a-zA-Z or ... | | |
| | `+` | **clipboard** (or '*') | `:help c_CTRL-R` | |
| | `"` | last del/copy | `%` | filename |

## :buffer #

| | |
|---|---|
| `:buffers` | list |
| `:new` | blank file/buffer |
| `:bn` | **next file** |
| `:bp` | prev file |
| `:bd` | **close file** |
| `:bd!` | force close |
| `:set lines=#` | |
| `:set columns=#` | |
| `:winpos # #` | GUI |

## Windows

`:help windows`
`^w•` or `:wincmd •`

| | | |
|---|---|---|
| `w` | **:switch to next** | |
| `c` | **:close!** | |
| `n` | :new | |
| `s` | **:split** | horz. |
| `v` | :vsplit | vertical |
| `o` | :only | maximize |
| `=` | **all same size** | |
| `h` | move to win ← | |
| `j` | move to win ↓ | |
| `k` | move to win ↑ | |
| `l` | move to win → | |

`:sp [<filename>]`
*edit in split window*

## Cursor Bookmarks

`:marks` ` ' ^o

| | |
|---|---|
| `ma` | mark local 'a' |
| `'A` | goto global 'A' |
| `''` | **prev location** |

## File / Directory

`:Explore` or `:e .`
`:set browsedir=..`
one of *buffer last*

## Code

`= < > << >>`

| | |
|---|---|
| `:set backspace=indent,eol,start` | **allow backspace join lines** |
| `:set shiftwidth=#` | **indent width for ai** |
| `:set autoindent!` | **toggle auto-indent** |
| `:set lisp` | **lisp indent mode** |
| `:set tabstop=#` | **set tab stop every #th col** |
| `:set expandtab!` | **toggle hard/soft tabs** |
| `:set listchars=..` | *tab:>-,trail:-,nbsp:%,eol:$* |
| `:set list!` | **toggle whitespace** |
| `:set colorcolumn=80` | **visible right margin indicator** |
| `noremap + :s/^/\/\//<CR>` | **block comment** |
| `:set numbers!` | **toggle line numbers** |
| `:set wrap!` | **toggle linewrap display** |
| `:set showmatch` | **highlite matching ()** |
| `noremap - :s/^\/\///<CR>` | **uncomment** |

:h operator                :h navigation

# [operator][count][motion]

**d** delete/cut

**y** yank/copy

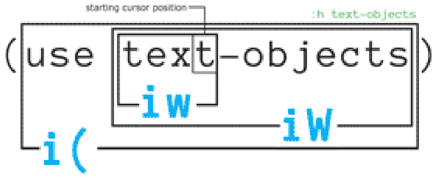
**c** change

| | |
|---|---|
| gU<br>make uppercase | ~<br>swap case |
| <<br>shift left | =<br>indent |

Any motion can follow an operator. Marks and searches count as motions, too! d/foo will delete from the cursor to the next instance of "foo". y3fi will yank from the cursor to the 3rd "i" on the line after it. Counts can also come before operators: 5dd will delete five lines.

| | |
|---|---|
| w | word |
| W | WORD |
| s | sentence |
| [ , ] | [ ] block |
| ( , ) | ( ) block |
| < , > | < > block |
| t | XML/HTML tag |
| { , } | { } block |
| " , ' | quoted string |

:h text-objects



**gg** first line

**^b** up 1 page

**^u** up ½ page

**k** up 1 line

:h up-down-motions

| | ts | sw | sts | et |
|---|---|---|---|---|
| use spaces only | n | n | n | on |
| use tabs only | n | n | 0 | off |

Set n to desired tab width (default 8)

| | | |
|---|---|---|
| tabstop | ts | Columns per tabstop |
| shiftwidth | sw | Columns per << |
| softtabstop | sts | Spaces per tab |
| expandtab | et | <Tab> inserts spaces |

**MIXING TABS AND SPACES IS RIGHT OUT.**
(that means don't do it.)

| | |
|---|---|
| :retab | Replace all tabs with spaces according to current tabstop setting |

| | | |
|---|---|---|
| fileformat | ff | Try changing this if your line-endings are messed up |
| list | | Display whitespace visibly according to listchars |

:h left-right-motions

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| beginning of line | 0 | first non-blank character | ^ | previous WORD | B | previous word | b | previous character | h | | |
| next character | l | end of word | e | beginning of next word | w | end of WORD | E | beginning of next WORD | W | end of line | $ |

:h pattern-searches

## SEARCHING

| Prev | Next | Forward | Backward | Matches |
|---|---|---|---|---|
| N | n | /*foo* | ?*foo* | *foo* |
| | | * | # | word under cursor |
| ; | , | t*x* | T*x* | upto *x* |
| | | f*x* | F*x* | find *x* |

:h mark-motions

| | | | | | |
|---|---|---|---|---|---|
| m*m* | set mark *m* (a-z) in file | m*M* | set mark *M* (A-Z) across files | '[ | jump to first char of just-changed text |
| '*m* | jump to first char of line containing *m* | `*m* | jump to exact character of *m* | '' | jump back to last jump |

**j** down 1 line

**^d** down ½ page

**^f** down 1 page

**G** last line

| | | | | | |
|---|---|---|---|---|---|
| p | paste after cursor | P | paste before cursor | ^[ | return to Normal mode |
| u | undo | ^r | redo | . | repeat |
| gf<br>file-searching | find file under cursor in path and jump to it | dd | delete current line | yy | yank current line |
| x | delete character after cursor | % | jump to matching paren | r | replace char under cursor |
| *n*G | jump to line *n* | ^o<br>jump-motions | jump back | ^i<br>jump-motions | jump forward |
| zz | center screen on cursor | zt | align top of screen with cursor | zb | align bottom of screen with cursor |
| == | auto-indent current line | << | shift current line left by shiftwidth | >> | shift current line right by shiftwidth |

Pass a directory to the :edit command to open a directory explorer. Instructions for usage are at the top of the screen.

Using ^[ to return to Normal mode lets you keep your fingers on the home row. It's even easier if you map Caps Lock to Control!

## ENTERING INSERT MODE

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| beginning of line | I | before cursor | i | after cursor | a | end of line | A | | |
| previous line | O | next line | o | substitute character | s | substitute line | S | line from cursor | C |

:h insert.txt

## COOL INSERT MODE STUFF

| | | | |
|---|---|---|---|
| ^w | delete word before cursor | ^u | delete line before cursor |
| ^r*r* | insert the contents of register *r* | ^r= | use the expression register (try ^r=5+10) |
| ^t | increase line indent by shiftwidth | ^d | decrease line indent by shiftwidth |
| ^x^l | line completion | ^n | find next completion suggestion according to complete |

## ENTERING VISUAL (SELECT) MODE

| | |
|---|---|
| v | The most basic type. Use Visual mode to select characters within a line. |
| V | Useful for moving chunks of a program around the file. Use Visual Line mode to select one or more lines. |
| ^v | Great for working with tables made of text, or anything that happens to be conveniently aligned. Visual Block mode can be used to select boxes across lines. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| switch cursor to start/end<br>:h v_o | o | re-select previous area<br>:h gv | gv | prepend to each Visual block line<br>:h v_b_I | I | jump to start of prior area<br>:h '< | '< |

| | | | |
|---|---|---|---|
| ZZ | Write current file, if modified, and quit | ZQ | Quit without checking for changes (like :q!) |

| | |
|---|---|
| :write | Write current file |
| :wq | Write current file and quit |

Use :scriptnames to list all files sourced during initialization.

| | |
|---|---|
| :syntax | Enable and configure syntax highlighting<br>Use :sy sync fromstart to redraw broken highlights |

:h quickfix

| | |
|---|---|
| :make | Run a compiler and enter quickfix mode |

| | | | |
|---|---|---|---|
| :! | Execute external shell command | ! | Filter motion with shell command |

Use :earlier and :later to quickly jump backward and forward in a file's history.

| | |
|---|---|
| :read | Read external program output into current file |

:h cmdline.txt

## COMMAND-LINE MODE ONLY

| | | | | | |
|---|---|---|---|---|---|
| edit using Normal mode<br>cmdwin | ^f | insert word under cursor<br>cmdline-editing | ^r^w | completion suggestions<br>cmdline-completion | ^d |

Put cnoremap %% <C-R>=expand('%:h').'/'<CR> in your .vimrc so you can type %% in Command-line mode to refer to the directory of the current file, regardless of pwd.

Supply % as a range to the :substitute command to run it on every line in the file.

| | | |
|---|---|---|
| :%s/Scribbl/Design/ | "Scribbled" -> "Designed" | |

Specify the "g" flag to apply the substitution to *every* match on a line.

| | | |
|---|---|---|
| :s/[dla]//g | "badly" -> "by" | :h s_flags, :h /[] |

Vim supports many regular expression features.

| | | |
|---|---|---|
| :s/..k/ax/ | "Mook" -> "Max" | :h usr_27, :h /. |

Use _. instead of . if you want to search across multiple lines.

| | | |
|---|---|---|
| :%s/heat_.*Bungle/anto/ | "Cheatsheet\nBungler" -> "Cantor" | :h /_. |

Special escapes can be used to change the case of substitutions.

| | | |
|---|---|---|
| :s_\(f..\)_\U\1\E_ | "foobar" -> "FOObar" | :h sub-replace-special |

Use :global to perform a command on matching lines.

| | |
|---|---|
| :g/foobar/delete | Delete all lines containing "foobar" |

If your pattern contains slashes, just use a different character as your delimiter.

:s_Data/Lore_Brent Spiner_

Use \= to evaluate expressions with replacement groups.

| | | |
|---|---|---|
| :s_\d_\=submatch(0) + 1_g | "10 25" -> "21 36" | :h sub-replace-\= |

:help

| | |
|---|---|
| :h cmd | Normal mode cmd help |
| :h i_cmd | Insert mode cmd help |
| :h v_cmd | Visual mode cmd help |
| :h c_cmd | Command-line editing cmd help |
| :h :cmd | Command-line cmd help |
| :h 'option' | Option help |
| :helpgrep | Search through all help docs! |

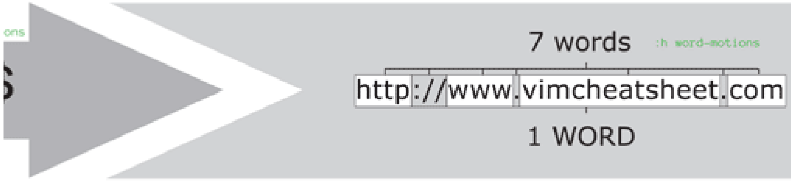
vim

:h tags-and-searches

| | |
|---|---|
| ^] | Jump to tag under cursor, including [tags] in help files |
| ^t | Jump back up the tag-list |
| g^] | Jump to tag if it's the only match; else list matching tags |

:h keycodes

| | | | |
|---|---|---|---|
| <CR> | ^m | \r | Enter |
| <Tab> | ^i | \t | Tab |
| <C-n> | ^n | | Ctrl-n |
| <M-n> | | | Alt-n |
| <Esc> | ^[ | | Escape |
| <BS> | ^h | \b | Backspace |
| <Del> | | | Delete |



:h options

| | |
|---|---|
| :set opt? | View current value of opt |
| :set noopt | Turn off flag opt |
| :set opt | Turn on flag opt |
| :set opt=val | Overwrite value of opt |
| :set opt+=val | Append to value of opt |
| :echo &opt | Access opt as a variable |

| | | |
|---|---|---|
| hidden | hid | Lets you switch buffers without saving |
| laststatus | ls | Show status line never (0), always (2) or with 2+ windows (1) |
| hlsearch | hls | Highlight search matches. Also see 'highlight' |
| number | nu | Show line numbers |
| showcmd | sc | Show commands as you type them |
| ruler | ru | Show line and column number of the cursor |
| backspace | bs | Set to '2' to make backspace work like sane editors |
| wrap | | Control line wrapping |
| background | bg | Set to 'dark' if you have a dark color scheme |

Use a instead of i when beginning text-object motions to include delimiters or surrounding whitespace. For example, di( will change "(foo)" into "()", but da( will delete the parentheses as well.

:h buffers

| | |
|---|---|
| :ls | List all open files |
| :b path | Jump to unique file matching path. Use **<Tab>** to scroll through available completions! |
| :bn | Jump to file n, number from first column of :ls |
| :bnext | Jump to next file |
| :bprev | Jump to previous file |
| :bdelete | Remove file from the buffer list |
| :edit | Open a file for editing |
| :enew | Open a blank new file for editing |

:h windows

| | |
|---|---|
| :split | Split current window horizontally |
| :vsplit | Split current window vertically |
| ^w hjkl | Move cursor to window left, below, above or to the right of the current window |
| ^w HJKL | Move current window to left, bottom, top, or right of screen |
| ^w r | Rotate windows clockwise |
| ^w +-<> | Increase/decrease current window height/width |
| ^w T | Move current window to a new tab |
| :only | Close all windows except current window |
| :bufdo | Execute a command in each open file |

## REGISTERS are CLIPBOARDS

All commands that delete, copy, or paste text use registers. To change which register is used by a command, type the register before the command. The default register is called "the unnamed register", and it is invoked with a pair of double-quotes (""). Typing dd or yy is the same as typing ""dd or ""yy. Think of the first " as a short way of saying "register", so "" is pronounced "register "", and "a, "register a".

:h registers

| | | |
|---|---|---|
| :registers | | View all current registers |
| :echo @r | | Access register r as a variable |
| "/ | Last search pattern register | Contains the last pattern you searched for |
| "_ | The black hole register | Use this to delete without clobbering any register ("_dd) |
| "0 | Last yank register | Contains the last text you yanked |
| "1 | Last big delete register | Contains the last line(s) you deleted |
| "2-"9 | Big delete register stack | Every time "1 is written to, its content is pushed to "2, then "2 to "3, and so on |
| "- | Small delete register | Contains the last text you deleted within a single line |
| "+ | System clipboard | If the OS integration gods smile upon you, this register reads and writes to your system clipboard. |
| "a-"z | Named registers | 26 registers for you to play with |
| "A-"Z | Append registers | Using upper-case to refer to a register will append to it rather than overwrite it |
| qr | Record | Record into register r. Stop recording by hitting q again |
| @r | Playback | Execute the contents of register r |
| @@ | Repeat last playback | Repeat the last @r, this is particularly useful with a count |

vim one-liner used to sort the list of names by length:

```
:exe 'g/^/let @x = len(getline("."))| normal "xPa ' | sort n | :g//normal dw
```

Use :map to view all current custom key mappings. Read :h map-which-keys for a guide on which keys are best for your own custom mappings. Get used to Vim's help system - it's a fantastic resource!

1st Edition
funded by...
MML
Adam.
jdiems
Lin Qiu
@webjoe
grantok
storrgie
David G.
Digininja
Dave Remy
Ralph Bean
Arve Leken
Rajat Ohai
Paul-Kenji
Mark Dixon
Neil Munro
Kyle Verma
Guy Bensky
Ross Timson
Abdul Qabiz
Daniel Beck
Hazel Smith
Kelly Raila
Charles Chen
Pivotal Labs
Eric Sporkin
M. Adam Price
Daniel Hahler
Rauli Silvola
Lawrence Kemp
Bill LaPierre
Brandon Tyree
Laurence Emms
Infonautic.de
@andorsosthus
Damon Jablons
Travis McHenry
Ganesh Sugunan
Rouven Hernier
Digital Gnomes
Sam Napolitano
Jens O. Meiert
Thomas Nitsche
Philip Ratzsch
Hurricane Labs
Michael Schmidt
Matthew Datcher
Arya Reais-Parsi
David Macfarlane
Christer Edvartsen
St. Louis Vim Geeks
Haraldur Tristan Gunnarsson
and others!