

## Best of Vim Tips

zzapper 16 Years of Vi + 8+ years of Vim and still learning

20Nov15 : Last Update (Now in VIM Help Format :h helptags)

### These Tips are now being maintained at [zzapper.co.uk/vimtips.html](http://zzapper.co.uk/vimtips.html)

```
-----
BEGIN
*vimtips.txt*   For Vim version 7.3.
-----
Tips
Home
Vim Tips
Blog
(NEW)
ZSH
Cygwin
VimTools
Buy Vim
Book
Support
VIM
Submit to
Social
Websites

" new items marked [N] , corrected items marked [C]
" *best-searching*
/joe/e          : cursor set to End of match
3/joe/e+1       : find 3rd joe cursor set to End of match plus 1 [C]
/joe/s-2        : cursor set to Start of match minus 2
/joe/+3         : find joe move cursor 3 lines down
/^joe.*fred.*bill/ : find joe AND fred AND Bill (Joe at start of line)
/^ [A-J]/       : search for lines beginning with one or more A-J
/begin\_.end    : search over possible multiple lines
/fred\_s*joe/    : any whitespace including newline [C]
/fred\|joe      : Search for FRED OR JOE
/.*fred\&.*joe  : Search for FRED AND JOE in any ORDER!
/\<fred>/       : search for fred but not alfred or frederick [C]
/\<\d\d\d\d>    : Search for exactly 4 digit numbers
/\D\d\d\d\D    : Search for exactly 4 digit numbers
/\<\d{4}>\>     : same thing
/\([0-9]\|^\)\%.*% : Search for absence of a digit or beginning of line
" finding empty lines
/^\\n{3}        : find 3 empty lines
/^str.\nstr     : find 2 successive lines starting with str
/\(^str.\n\)\{2} : find 2 successive lines starting with str
" using regexp memory in a search find fred.*joe.*joe.*fred *C*
/\(fred\).*\(joe\).*\2.*\1
" Repeating the Regexp (rather than what the Regexp finds)
/^\([^\,]*\)\{8}
" visual searching
:vmap // y/<C-R>"<CR>      : search for visually highlighted text
:vmap <silent> // y/<C-R>=escape(,"',\.\$^~[]')<CR><CR> : with spec chars
" \zs and \ze regex delimiters :h /\zs
/<\zs[^\>]*\ze>          : search for tag contents, ignoring chevrons
" zero-width :h /\@=
/<\@<=[^>]*\@=          : search for tag contents, ignoring chevrons
/<\@<=[^>]*\@=          : search for tags across possible multiple lines
" searching over multiple lines \_ means including newline
/<!--\_p{-}-->          : search for multiple line comments
/fred\_s*joe/           : any whitespace including newline *C*
/bugs\(\_\.)*bunny     : bugs followed by bunny anywhere in file
:h \_                  : help
" search for declaration of subroutine/function under cursor
:nmap gx yiw/^^(sub<bar>function\)\s\+<C-R>"<CR>
" multiple file search
:bufdo /searchstr/      : use :rewind to recommence search
" multiple file search better but cheating
:bufdo %s/searchstr/&/gic : say n and then a to stop
" How to search for a URL without backslashing
?http://www.vim.org/    : (first) search BACKWARDS!!! clever huh!
" Specify what you are NOT searching for (vowels)
/\c\([^\aeiou]\&\a){4}  : search for 4 consecutive consonants
/\%>20l\%<30lgoat       : Search for goat between lines 20 and 30 [N]
/^.\{-}home.\{-}\zs/home/e : match only the 2nd occurrence in a line of "home" [N]
:%s/home.\{-}\zs/home/alone : Substitute only the 2nd occurrence of home in any line [U]
" find str but not on lines containing tongue
^\(. *tongue.*)\@!.*nose.*$
\v^((tongue)@!.) *nose((tongue)@!.) *$
.*nose.*\&^\%(tongue\)\@!.*$
:v/tongue/s/nose/&/gic
'a,'bs/extrascost//gc    : trick: restrict search to between markers (answer n) [N]
-----
" *best-substitution*
:%s/fred/joe/igc        : general substitute command
:%s//joe/igc            : Substitute what you last searched for [N]
:%s/~ /sue/igc          : Substitute your last replacement string [N]
:%s/\r//g              : Delete DOS returns ^M
" Is your Text File jumbled onto one line? use following
:%s/\r/\r/g            : Turn DOS returns ^M into real returns
:%s= *$==              : delete end of line blanks
:%s= \+$==             : Same thing
:%s#\s*\r\?$##        : Clean both trailing spaces AND DOS returns
:%s#\s*\r*$##          : same thing
" deleting empty lines
:%s/^\n{3}///          : delete blocks of 3 empty lines
:%s/^\n+/\r/           : compressing empty lines
:%s#<[>]\>##g          : delete html tags, leave text (non-greedy)
:%s#<\_.\{-1,\}>##g     : delete html tags possibly multi-line (non-greedy)
:%s#.*\(\d\+hours\).*#\l# : Delete all but memorised string (\l) [N]
" parse xml/soap
:%s#><([/])\#>\r<\l#g   : split jumbled up XML file into one tag per line [N]
:%s/</r&/g             : simple split of html/xml/soap [N]
```

```

:%s#<[^\r]#\r&#gic      : simple split of html/xml/soap but not closing tag [N]
:%s#<[^\r]#\r&#gi       : parse on open xml tag [N]
:%s#\[\d\+\]\#\r&#g     : parse on numbered array elements [1] [N]
ggVGgJ                  : rejoin XML without extra spaces (gJ) [N]
%s=\n#\d=\r&=g         : parse PHP error stack [N]
" VIM Power Substitute
:'a,'bg/fred/s/dick/joe/igc : VERY USEFUL
" duplicating columns
:%s= [^ ]\+$=&&=         : duplicate end column
:%s= \f\+$=&&=          : Duplicate filename
:%s= \S\+$=&&           : usually the same
" memory
:%s#example#& = &#gic    : duplicate entire matched string [N]
:%s#.*\{tbl_\w\+\}.*#\1# : extract list of all strings tbl_* from text [NC]
:s/\(.*\):\{1,2\}/2 : \1/ : reverse fields separated by :
:%s/^\{.*\}\n\1$/\1/     : delete duplicate lines
:%s/^\{.*\}\n\1\}\n\1$/\1/ : delete multiple duplicate lines [N]
" non-greedy matching \{-}
:%s/^\{-}\.pdf/new.pdf/   : delete to 1st occurrence of pdf only (non-greedy)
%s#^\{-}\{0-9\}\{3,4\}serial\}\#\1#gic : delete up to 123serial or 1234serial [N]
" use of optional atom \?
:%s#<[zy]\?tbl_[a-z_]\+\>#\L&#gc : lowercase with optional leading characters
" over possibly many lines
:%s/<!--\_\{-}-->//       : delete possibly multi-line comments
:help /\{-}              : help non-greedy
" substitute using a register
:s/fred/<c-r>a/g           : sub "fred" with contents of register "a"
:s/fred/<c-r>asome_text<c-r>s/g
:s/fred/=a/g             : better alternative as register not displayed (not *) [C]
" multiple commands on one line
:%s/\f\.\gif\>\/\r&\r/g | v/\.\gif$/d | %s/gif/jpg/
:%s/a/but/gie|update|next : then use @: to repeat
" ORing
:%s/goat\|cow/sheep/gc   : ORing (must break pipe)
:'a,'bs#\[\|\]\##g       : remove [] from lines between markers a and b [N]
:%s/\v(.*\n){5}/&\r      : insert a blank line every 5 lines [N]
" Calling a VIM function
:s/_date_\/=strftime("%c")/ : insert datestring
:inoremap \zd <C-R>=strftime("%d%b%y")<CR> : insert date eg 31Jan11 [N]
" Working with Columns sub any str1 in col3
:%s:\(\(w\+s\+)\)\{2\}str1:lstr2:
" Swapping first & last column (4 columns)
:%s:\(w\+\)\(.*s\+)\(w\+)\$:\3\2\1:
" format a mysql query
:%s#<from>\|<where>\|<left join>\|<inner join>\>#\r&#g
" filter all form elements into paste register
:redir @*|sil exec 'g#<(\input|select|textarea|/\=form\)\>#p'|redir END
:nmap ,z :redir @*<Bar>sil exec 'g@<(\input<Bar>select\<Bar>textarea\<Bar>/\=form\)\>#p'<Bar>redir END<CR>
" substitute string in column 30 [N]
:%s/^\{30\}\)xx\lyy/
" decrement numbers by 3
:%s/\d\+/\=(submatch(0)-3)/
" increment numbers by 6 on certain lines only
:g/loc\|function/s/\d\+=submatch(0)+6/
" better
:%s#txtdev\zs\d#\=submatch(0)+1#g
:h /\zs
" increment only numbers gg\d\d by 6 (another way)
:%s/\(gg\)\@<=\d\+/\=submatch(0)+6/
:h zero-width
" rename a string with an incrementing number
:let i=10 | 'a,'bg/Abc/s/yy/\=i/ |let i=i+1 # convert yy to 10,11,12 etc
" as above but more precise
:let i=10 | 'a,'bg/Abc/s/xx\zsyy\ze/\=i/ |let i=i+1 # convert xxyy to xx11,xx12,xx13
" find replacement text, put in memory, then use \zs to simplify substitute
:%s/"\([^\]]+\).*\zsxx\1/
" Pull word under cursor into LHS of a substitute
:nmap <leader>z :%s#<<c-r>=expand("<cword>")<cr>\>#
" Pull Visually Highlighted text into LHS of a substitute
:vmap <leader>z :<C-U>%s/\<<c-r>*>/
" substitute singular or plural
:'a,'bs/bucket\(\s\)*\bowl\1/gic [N]
-----
" all following performing similar task, substitute within substitution
" Multiple single character substitution in a portion of line only
:%s,\(all\.*\)\@<=/,_g : replace all / with _ AFTER "all/"
" Same thing
:s#all/\zs.*#\=substitute(submatch(0), '/', '_', 'g')#
" Substitute by splitting line, then re-joining
:s#all/#&^M#|s#/#_#g|-j!
" Substitute inside substitute
:%s/.*/\='cp '.submatch(0).' all/'.'substitute(submatch(0), '/', '_', 'g')/
-----
" *best-global* command
:g/gladiolli/#          : display with line numbers (YOU WANT THIS!)
:g/fred.*joe.*dick/     : display all lines fred,joe & dick
:g/\<fred>/             : display all lines fred but not freddy
:g/^\s*$/d              : delete all blank lines
:g!/^dd/d               : delete lines not containing string

```

```

:v/^dd/d          : delete lines not containing string
:g/joe/,/fred/d   : not line based (very powerfull)
:g/fred/,/joe/j   : Join Lines [N]
:g/-----/.-10,.d : Delete string & 10 previous lines
:g/{/ ,/}- s/\n\+/r/g : Delete empty lines but only between {...}
:v/\S/d          : Delete empty lines (and blank lines ie whitespace)
:v/./,./.-j      : compress empty lines
:g/^$/./.-j      : compress empty lines
:g/<input\|<form/p : ORing
:g/^/put_        : double space file (pu = put)
:g/^/m0          : Reverse file (m = move)
:g/^/m$          : No effect! [N]
:'a,'bg/^/m'b    : Reverse a section a to b
:g/^/t.          : duplicate every line
:g/fred/t$       : copy (transfer) lines matching fred to EOF
:g/stage/t'a      : copy (transfer) lines matching stage to marker a (cannot use .) [C]
:g/^Chapter/t.|s/./-/g : Automatically underline selecting headings [N]
:g/\(^I[^^I]*\)\{80}/d : delete all lines containing at least 80 tabs
" perform a substitute on every other line
:g/^/ if line('.')%2|s/^/zz /
" match all lines containing "somestr" between markers a & b
" copy after line containing "otherstr"
:'a,'bg/somestr/co/otherstr/ : co(py) or mo(ve)
" as above but also do a substitution
:'a,'bg/str1/s/str1/&&&/|mo/str2/
:%norm jdd        : delete every other line
" incrementing numbers (type <c-a> as 5 characters)
:.,$g/^d/exe "norm! \<c-a>": increment numbers
:'a,'bg/d\+/\norm! ^A : increment numbers
" storing glob results (note must use APPEND) you need to empty reg a first with qaq.
" save results to a register/paste buffer
:g/fred/y A       : append all lines fred to register a
:g/fred/y A | :let @*=@a : put into paste buffer
:g//y A | :let @*=@a : put last glob into paste buffer [N]
:let @a=''|g/Barratt/y A |:let @*=@a
" filter lines to a file (file must already exist)
:'a,'bg/^Error/ . w >> errors.txt
" duplicate every line in a file wrap a print ' ' around each duplicate
:g/./yank|put|-ls/'"/g|s./Print '&'/
" replace string with contents of a file, -d deletes the "mark"
:g/^MARK$/r tmp.txt | -d
" display prettily
:g/<pattern>/z#.5 : display with context
:g/<pattern>/z#.5|echo "=====" : display beautifully
" Combining g// with normal mode commands
:g/|/norm 2f|r* : replace 2nd | with a star
" send output of previous global command to a new window
:nmap <F3> :redir @a<CR>:g//<CR>:redir END<CR>:new<CR>:put! a<CR><CR>
"
" *Best-Global-combined-with-substitute* (*power-editing*)
:'a,'bg/fred/s/joe/susan/gic : can use memory to extend matching
:/fred/,/joe/s/fred/joe/gic : non-line based (ultra)
:/biz/,/any/g/article/s/wheel/bucket/gic: non-line based [N]
" Find fred before beginning search for joe
:/fred/;/joe/-2,/sid/+3s/sally/alley/gIC
"
" create a new file for each line of file eg 1.txt,2.txt,3.txt etc
:g/^/exe ".w ".line(".").".txt"
"
" chain an external command
:.g/^/ exe ".!sed 's/N/X/'" | s/I/Q/ [N]
"
" Operate until string found [N]
d/fred/          :delete until fred
y/fred/          :yank until fred
c/fred/e         :change until fred end
v12|            : visualise/change/delete to column 12 [N]
"
" Summary of editing repeats [N]
.      last edit (magic dot)
&      last substitute
:%&    last substitute every line
:%&gic last substitute every line confirm
g%     normal mode repeat last substitute
g&     last substitute on all lines
@@     last recording
@:     last command-mode command
:!!    last :! command
:~     last substitute
:help repeating
"
" Summary of repeated searches
;      last f, t, F or T
,      last f, t, F or T in opposite direction
n      last / or ? search
N      last / or ? search in opposite direction
"
" *Absolutely-essential*

```

```

-----
* # g* g#      : find word under cursor (<word>) (forwards/backwards)
%             : match brackets {}[]()
.             : repeat last modification
@:            : repeat last : command (then @@)
matchit.vim   : % now matches tags <tr><td><script> <?php etc
<C-N><C-P>    : word completion in insert mode
<C-X><C-L>    : Line complete SUPER USEFUL
/<C-R><C-W>    : Pull <word> onto search/command line
/<C-R><C-A>    : Pull <WORD> onto search/command line
:set ignorecase : you nearly always want this
:set smartcase  : overrides ignorecase if uppercase used in search string (cool)
:syntax on      : colour syntax in Perl,HTML,PHP etc
:set syntax=perl : force syntax (usually taken from file extension)
:h regexp<C-D> : type control-D and get a list all help topics containing
                regexp (plus use TAB to Step thru list)
-----
" MAKE IT EASY TO UPDATE/RELOAD _vimrc
:nmap ,s :source $VIM/_vimrc
:nmap ,v :e $VIM/_vimrc
:e $MYVIMRC : edits your _vimrc wherever it might be [N]
" How to have a variant in your .vimrc for different PCs [N]
if $COMPUTERNAME == "NEWPC"
ab mypc vista
else
ab mypc dell25
endif
-----
" splitting windows
:vsplit other.php # vertically split current file with other.php [N]
-----
"VISUAL MODE (easy to add other HTML Tags)
:vmap sb "zdi<b><C-R>z</b><ESC> : wrap <b></b> around VISUALLY selected Text
:vmap st "zdi<?<C-R>z ?><ESC> : wrap <?> ?> around VISUALLY selected Text
-----
"vim 7 tabs
vim -p fred.php joe.php : open files in tabs
:tabe fred.php          : open fred.php in a new tab
:tab ball               : tab open files
:close                  : close a tab but leave the buffer *N*
" vim 7 forcing use of tabs from .vimrc
:noremap gf <C-W>gf
:cab e tabe
:tab sball              : retab all files in buffer (repair) [N]
-----
" Exploring
:e .                    : file explorer
:Exp(lore)             : file explorer note capital Ex
:Sex(plore)            : file explorer in split window
:browse e              : windows style browser
:ls                    : list of buffers
:cd ..                 : move to parent directory
:args                  : list of files
:pwd                   : Print Working Directory (current directory) [N]
:args *.php            : open list of files (you need this!)
:lcd %:p:h             : change to directory of current file
:cd :h                 : change to directory of current file [N]
:autocmd BufEnter * lcd %:p:h : change to directory of current file automatically (put in _vimrc)
-----
" Changing Case
guu                    : lowercase line
gUU                    : uppercase line
Vu                     : lowercase line
VU                     : uppercase line
g~~                    : flip case line
vEU                    : Upper Case Word
vE~                    : Flip Case Word
ggguG                  : lowercase entire file
" Titlise Visually Selected Text (map for .vimrc)
vmap ,c :s/\<(\.)(\k*)\>/\u\1\L\2/g<CR>
" Title Case A Line Or Selection (better)
vnoremap <F6> :s/\%V\<(\w\)\(\w*\)\>/\u\1\L\2/ge<cr> [N]
" titlise a line
nmap ,t :s/.*\&/<bar>:s/\<.\u&/g<cr> [N]
" Uppercase first letter of sentences
:s/[.!?]\_s\+<a/U&\E/g
-----
gf                    : open file name under cursor (SUPER)
:noremap gf :view <cfile><cr> : open file under cursor, create if necessary
ga                    : display hex,ascii value of char under cursor
ggVGg?               : rot13 whole file
ggg?G                : rot13 whole file (quicker for large file)
:8 | normal VGg?     : rot13 from line 8
:normal l0GVGg?      : rot13 from line 8
<C-A>,<C-X>          : increment,decrement number under cursor
win32 users must remap CNTRL-A
<C-R>=5*5            : insert 25 into text (mini-calculator)
-----
" Make all other tips superfluous

```

```

:h 42          : also http://www.google.com/search?q=42
:h holy-grail
:h!
-----
" disguise text (watch out) [N]
ggVGg?        : rot13 whole file (toggles)
:set rl!      : reverse lines right to left (toggles)
:g/^/m0       : reverse lines top to bottom (toggles)
:%s/\(\<.\{-\}>\)/\=join(reverse(split(submatch(1), '\zs')), '')/g : reverse all text *N*
-----
" History, Markers & moving about (what Vim Remembers) [C]
'.           : jump to last modification line (SUPER)
`.           : jump to exact spot in last modification line
g;          : cycle thru recent changes (oldest first)
g,          : reverse direction
:changes
:h changelist : help for above
<C-O>       : retrace your movements in file (starting from most recent)
<C-I>       : retrace your movements in file (reverse direction)
:ju(mps)    : list of your movements
:help jump-motions
:history     : list of all your commands
:his c      : commandline history
:his s      : search history
q/          : Search history Window (puts you in full edit mode) (exit CTRL-C)
q:          : commandline history Window (puts you in full edit mode) (exit CTRL-C)
:<C-F>      : history Window (exit CTRL-C)
-----
" Abbreviations & Maps
" Maps are commands put onto keys, abbreviations expand typed text [N]
" Following 4 maps enable text transfer between VIM sessions
:map <f7> : 'a,'bw! c:/aaa/x      : save text to file x
:map <f8> : r c:/aaa/x            : retrieve text
:map <f11> : .w! c:/aaa/xr<CR>    : store current line
:map <f12> : r c:/aaa/xr<CR>      : retrieve current line
:ab php    : list of abbreviations beginning php
:map ,     : list of maps beginning ,
" allow use of F10 for mapping (win32)
set wak=no : :h winaltkeys
" For use in Maps
<CR>       : carriage Return for maps
<ESC>      : Escape
<LEADER>   : normally \
<BAR>      : | pipe
<BACKSPACE> : backspace
<SILENT>   : No hanging shell window
"display RGB colour under the cursor eg #445588
:nmap <leader>c :hi Normal guibg=#<c-r>=expand("<cword>")<cr><cr>
:map <f2> /price only\\|versus/ :in a map need to backslash the \
" type table,,, to get <table></table>      ### Cool ###
imap ,,, <esc>bdwa<<esc>pa><cr></<esc>pa><esc>kA
" list current mappings of all your function keys
:for i in range(1, 12) | execute("map <F".i.">") | endfor [N]
" for your .vimrc
:cab ,f :for i in range(1, 12) \ | execute("map <F".i.">") \ | endfor
"chain commands in abbreviation
cabbrev vrep tabe class.inc \ | tabe report.php ## chain commands [N]
-----
" Simple PHP debugging display all variables yanked into register a
iab phpdb exit("<hr>Debug <C-R>a ");
-----
" Using a register as a map (preload registers in .vimrc)
:let @m=":'a,'bs/"
:let @s=":%!sort -u"
-----
" Useful tricks
" ayy@a      : execute "Vim command" in a text file
yy@          : same thing using unnamed register
u@.          : execute command JUST typed in
"ddw        : store what you delete in register d [N]
"ccaw       : store what you change in register c [N]
-----
" Get output from other commands (requires external programs)
:r!ls -R     : reads in output of ls
:put=glob('**') : same as above [N]
:r !grep "^ebay" file.txt : grepping in content [N]
:20,25 !rot13 : rot13 lines 20 to 25 [N]
!!date      : same thing (but replaces/filters current line)
" Sorting with external sort
:%!sort -u   : use an external program to filter content
:'a,'b!sort -u : use an external program to filter content
!l} sort -u  : sorts paragraph (note normal mode!!)
:g/^$/;/^$/-1!sort : Sort each block (note the crucial ;)
" Sorting with internal sort
:sort /*.*%2v/ : sort all lines on second column [N]
" number lines (linux or cygwin only)
:new | r!nl # [N]
-----
" Multiple Files Management (Essential)

```

```

:bn          : goto next buffer
:bp          : goto previous buffer
:wn          : save file and move to next (super)
:wp          : save file and move to previous
:bd          : remove file from buffer list (super)
:bun         : Buffer unload (remove window but not from list)
:badd file.c : file from buffer list
:b3          : go to buffer 3 [C]
:b main      : go to buffer with main in name eg main.c (ultra)
:sav php.html : Save current file as php.html and "move" to php.html
:sav! %<.bak  : Save Current file to alternative extension (old way)
:sav! %:r.cfm : Save Current file to alternative extension
:sav %:s/fred/joe/ : do a substitute on file name
:sav %:s/fred/joe/:r.bak2 : do a substitute on file name & ext.
:!mv % %:r.bak : rename current file (DOS use Rename or DEL)
:help filename-modifiers
:e!          : return to unmodified file
:w c:/aaa/%  : save file elsewhere
:e #         : edit alternative file (also cntrl-^)
:rew        : return to beginning of edited files list (:args)
:brew       : buffer rewind
:sp fred.txt : open fred.txt into a split
:sball,;sb   : Split all buffers (super)
:scrollbind  : in each split window
:map <F5> :ls<CR>;e # : Pressing F5 lists all buffer, just type number
:set hidden  : Allows to change buffer w/o saving current buffer
-----
" Quick jumping between splits
map <C-J> <C-W>j<C-W>_
map <C-K> <C-W>k<C-W>_
-----
" Recording (BEST Feature of ALL)
qq # record to q
your complex series of commands
q # end recording
@q to execute
@@ to Repeat
5@@ to Repeat 5 times
qQ@qQ          : Make an existing recording q recursive [N]
" editing a register/recording
"qp            :display contents of register q (normal mode)
<ctrl-R>q      :display contents of register q (insert mode)
" you can now see recording contents, edit as required
"qdd          :put changed contacts back into q
@q            :execute recording/register q
" Operating a Recording on a Visual BLOCK (blockwise)
1) define recording/register
qq:s/ to/ from/g^Mq
2) Define Visual BLOCK
V}
3) hit : and the following appears
: '<,'>
4) Complete as follows
: '<,'>norm @q
-----
"combining a recording with a map (to end up in command mode)
"here we operate on a file with a recording, then move to the next file [N]
:nnoremap ] @q:update<bar>bd
-----
" Visual is the newest and usually the most intuitive editing mode
" Visual basics
v          : enter visual mode
V          : visual mode whole line
<C-V>      : enter VISUAL BLOCKWISE mode (remap on Windows to say C-Q *C*
gv         : reselect last visual area (ultra)
o          : navigate visual area
"*y or "+y : yank visual area into paste buffer [C]
V%         : visualise what you match
V}J        : Join Visual block (great)
V}gJ       : Join Visual block w/o adding spaces
`[v`       : Highlight last insert
:%s/\%Vold/new/g : Do a substitute on last visual area [N]
-----
" Delete 8th and 9th characters of 10 successive lines [C]
08l<c-v>10j2ld (use Control Q on win32) [C]
-----
" how to copy a set of columns using VISUAL BLOCK
" visual block (AKA columnwise selection) (NOT BY ordinary v command)
<C-V> then select "column(s)" with motion commands (win32 <C-Q>)
then c,d,y,r etc
-----
" how to overwrite a visual-block of text with another such block [C]
" move with hjkl etc
Pick the first block: ctrl-v move y
Pick the second block: ctrl-v move P <esc>
-----
" text objects :h text-objects [C]
daW          : delete contiguous non whitespace
di< yi< ci<  : Delete/Yank/Change HTML tag contents

```



```

da< ya< ca< : Delete/Yank/Change whole HTML tag
dat dit : Delete HTML tag pair
diB daB : Empty a function {}
das : delete a sentence
-----
" _vimrc essentials
:imap <TAB> <C-N> : set tab to complete [N]
:set incsearch : jumps to search word as you type (annoying but excellent)
:set wildignore=*.o,*.obj,*.bak,*.exe : tab complete now ignores these
:set shiftwidth=3 : for shift/tabbing
:set vb t_vb=" : set silent (no beep)
:set browsedir=buffer : Maki GUI File Open use current directory
-----
set iminsert=0 (or toggle CTRL-^ ) : reset "stuck" caps lock key [N]
-----
" launching Win IE
:nmap ,f :update<CR>:silent !start c:\progra~1\intern~1\iexplore.exe file://%:p<CR>
:nmap ,i :update<CR>: !start c:\progra~1\intern~1\iexplore.exe <cWORD><CR>
-----
" FTPing from VIM
cmap ,r :Nread ftp://209.51.134.122/public_html/index.html
cmap ,w :Nwrite ftp://209.51.134.122/public_html/index.html
gvim ftp://www.somedomain.com/index.html # uses netrw.vim
-----
" appending to registers (use CAPITAL)
" yank 5 lines into "a" then add a further 5
"a5yy
10j
"A5yy
-----
[I : show lines matching word under cursor <cword> (super)
-----
" Conventional Shifting/Indenting
:'a,'b>>
" visual shifting (builtin-repeat)
:vnoremap < <gv
:vnoremap > >gv
" Block shifting (magic)
>i{
>a{
" also
>% and <%
== : index current line same as line above [N]
-----
" Redirection & Paste register *
:redir @* : redirect commands to paste buffer
:redir END : end redirect
:redir >> out.txt : redirect to a file
" Working with Paste buffer
"*yy : yank current line to paste
"*p : insert from paste buffer
" yank to paste buffer (ex mode)
:'a,'by* : Yank range into paste
:%y* : Yank whole buffer into paste
:.y* : Yank Current line to paster
" filter non-printable characters from the paste buffer
" useful when pasting from some gui application
:nmap <leader>p :let @* = substitute(@*,'[^\[:print:]]',' ','g')<cr>"*p
:set paste : prevent vim from formatting pasted in text *N*
-----
" Re-Formatting text
gq} : Format a paragraph
gqap : Format a paragraph
ggVGgq : Reformat entire file
Vgq : current line
" break lines at 70 chars, if possible after a ;
:s/.\{,69\};\s*\|.\{,69\}\s\+/\&\r/g
-----
" Operate command over multiple files
:argdo %s/foo/bar/e : operate on all files in :args
:bufdo %s/foo/bar/e
:windo %s/foo/bar/e
:argdo exe '!sort'|w! : include an external command
:bufdo exe "normal @q" | w : perform a recording on open files
:silent bufdo !zip proj.zip %p : zip all current files
-----
" Command line tricks
gvim -h : help
ls | gvim - : edit a stream!!
cat xx | gvim - -c "v/^d\d\|^[3-9]/d " : filter a stream
gvim -o file1 file2 : open into a horizontal split (file1 on top, file2 on bottom) [C]
gvim -O file1 file2 : open into a vertical split (side by side,for comparing code) [N]
" execute one command after opening file
gvim.exe -c "/main" joe.c : Open joe.c & jump to "main"
" execute multiple command on a single file
vim -c "%s/ABC/DEF/ge | update" file1.c
" execute multiple command on a group of files
vim -c "argdo %s/ABC/DEF/ge | update" *.c
" remove blocks of text from a series of files

```

```

vim -c "argdo /begin/+1,/end/-1g/^/d | update" *.c
" Automate editing of a file (Ex commands in convert.vim)
vim -s "convert.vim" file.c
"load VIM without .vimrc and plugins (clean VIM) e.g. for HUGE files
gvim -u NONE -U NONE -N
" Access paste buffer contents (put in a script/batch file)
gvim -c 'normal ggdG"*p' c:/aaa/xp
" print paste contents to default printer
gvim -c 's/^/\=@*/|hardcopy!|q!'
" gvim's use of external grep (win32 or *nix)
:!grep somestring *.php      : creates a list of all matching files [C]
" use :cn(ext) :cp(rev) to navigate list
:h grep
" Using vimgrep with copen
vimgrep /keywords/ *.php
:copen
-----
" GVIM Difference Function (Brilliant)
gvim -d file1 file2      : vimdiff (compare differences)
dp                        : "put" difference under cursor to other file
do                        : "get" difference under cursor from other file
[c                        : jump backwards to change
]c                        : jump forwards to change
:set diffopt+=icase      : switch case sensitivity on and off
" complex diff parts of same file [N]
:1,2yank a | 7,8yank b
:tabedit | put a | vnew | put b
:windo diffthis
-----
" Vim traps
In regular expressions you must backslash + (match 1 or more)
In regular expressions you must backslash | (or)
In regular expressions you must backslash ( (group)
In regular expressions you must backslash { (count)
/fred\+/                 : matches fred/freddy but not free
/\(fred\) \{2,3}/        : note what you have to break
-----
" \v or very magic (usually) reduces backslashing
/codes\(\n\|s\) *where   : normal regexp
/\v codes\(\n\|s\) *where : very magic
-----
" pulling objects onto command/search line (SUPER)
<C-R><C-W> : pull word under the cursor into a command line or search
<C-R><C-A> : pull WORD under the cursor into a command line or search
<C-R>-      : pull small register (also insert mode)
<C-R>[0-9a-z] : pull named registers (also insert mode)
<C-R>%      : pull file name (also #) (also insert mode)
<C-R>=somevar : pull contents of a variable (eg :let sray="ray[0-9]")
-----
" List your Registers
:reg          : display contents of all registers
:reg a        : display content of register a
:reg 12a      : display content of registers 1,2 & a [N]
"5p          : retrieve 5th "ring"
"1p...       : retrieve numeric registers one by one
:let @y='yy@' : pre-loading registers (put in .vimrc)
qqq          : empty register "q"
qaq          : empty register "a"
:reg ./%:*"   : the seven special registers [N]
:reg 0        : what you last yanked, not affected by a delete [N]
"_dd         : Delete to blackhole register "_ , don't affect any register [N]
-----
" manipulating registers
:let @a=@_    : clear register a
:let @a=""    : clear register a
:let @a=@     : Save unnamed register [N]
:let @*=@a    : copy register a to paste buffer
:let @*=@     : copy last command to paste buffer
:let @*=@/    : copy last search to paste buffer
:let @*=@%    : copy current filename to paste buffer
-----
" help for help (USE TAB)
:h quickref    : VIM Quick Reference Sheet (ultra)
:h tips        : Vim's own Tips Help
:h visual<C-D><tab> : obtain list of all visual help topics
                  : Then use tab to step thru them
:h ctrl<C-D>    : list help of all control keys
:helpg uganda  : grep HELP Files use :cn, :cp to find next
:helpgrep edit.*director: grep help using regexp
:h :r          : help for :ex command
:h CTRL-R      : normal mode
:h /\r         : what's \r in a regexp (matches a <CR>)
:h \\zs        : double up backslash to find \zs in help
:h i_CTRL-R    : help for say <C-R> in insert mode
:h c_CTRL-R    : help for say <C-R> in command mode
:h v_CTRL-V    : visual mode
:h tutor       : VIM Tutor
<C-J>          : jump to {keyword} under cursor in help file [C]
<C-[>, <C-T>  : Move back & Forth in HELP History

```



```

gvim -h          : VIM Command Line Help
:cabbrev h tab help : open help in a tab [N]
-----
" where was an option set
:scriptnames      : list all plugins, _vimrcs loaded (super)
:verbose set history? : reveals value of history and where set
:function         : list functions
:func SearchCompl : List particular function
-----
" making your own VIM help
:helptags /vim/vim64/doc : rebuild all *.txt help files in /doc
:help add-local-help
" save this page as a VIM Help File [N]
:sav! $VIMRUNTIME/doc/vimtips.txt|:1,/ ^__BEGIN__/_d|:/^__END__/_,$d|:w!|:helptags $VIMRUNTIME/doc
-----
" running file thru an external program (eg php)
map <f9> :update<CR>:!c:/php/php.exe %<CR>
map <f2> :update<CR>:!perl -c %<CR>
-----
" capturing output of current script in a separate buffer
:new | r!perl # : opens new buffer,read other buffer
:new! x.out | r!perl # : same with named file
:new+read!ls
-----
" create a new buffer, paste a register "q" into it, then sort new buffer
:new +put q|%!sort
-----
" Inserting DOS Carriage Returns
:%s/$/\<C-V><C-M>&/g : that's what you type
:%s/$/\<C-Q><C-M>&/g : for Win32
:%s/$/\^M&/g : what you'll see where ^M is ONE character
-----
" automatically delete trailing Dos-returns,whitespace
autocmd BufRead * silent! %s/[ \r \t]\+$//
autocmd BufEnter *.php :%s/[ \t\r]\+$//e
-----
" perform an action on a particular file or file type
autocmd VimEnter c:/intranet/note011.txt normal! ggVGg?
autocmd FileType *.pl exec('set fileformats=unix')
-----
" Retrieving last command line command for copy & pasting into text
i<c-r>:
" Retrieving last Search Command for copy & pasting into text
i<c-r>/
-----
" more completions
<C-X><C-F> :insert name of a file in current directory
-----
" Substituting a Visual area
" select visual area as usual (:h visual) then type :s/Emacs/Vim/ etc
: '<,'>s/Emacs/Vim/g : REMEMBER you dont type the '<.'>
gv : Re-select the previous visual area (ULTRA)
-----
" inserting line number into file
:g/^/exec "s/^/".strpart(line(".")." " , 0, 4)
:%s/^/\=strpart(line(".")." " , 0, 5)
:%s/^/\=line('.'). ' '
-----
" *numbering lines VIM way*
:set number : show line numbers
:map <F12> :set number!<CR> : Show linenumbers flip-flop
:%s/^/\=strpart(line('.')." " ,0,&ts)
" numbering lines (need Perl on PC) starting from arbitrary number
:'a,'b!perl -pne 'BEGIN{$a=223} substr($_,2,0)=$a++'
" Produce a list of numbers
" Type in number on line say 223 in an empty file
qqmnYP`n^Aq : in recording q repeat with @q
" increment existing numbers to end of file (type <c-a> as 5 characters)
:.,$g/^/d/exe "normal! \<c-a>"
" advanced incrementing
http://vim.sourceforge.net/tip\_view.php?tip\_id=150
-----
" *advanced incrementing* (really useful)
" put following in _vimrc
let g:I=0
function! INC(increment)
let g:I =g:I + a:increment
return g:I
endfunction
" eg create list starting from 223 incrementing by 5 between markers a,b
:let I=223
:'a,'bs/^/\=INC(5)/
" create a map for INC
cab viminc :let I=223 \ | 'a,'bs/$/\=INC(5)/
-----
" *generate a list of numbers* 23-64
o23<ESC>qqYp<C-A>q40@q
unmap = : get the original vim command on a mapped key [N]
-----

```

```

" editing/moving within current insert (Really useful)
<C-U>                : delete all entered
<C-W>                : delete last word
<HOME><END>           : beginning/end of line
<C-LEFTARROW><C-RIGHTARROW> : jump one word backwards/forwards
<C-X><C-E>, <C-X><C-Y> : scroll while staying put in insert
-----
#encryption (use with care: DON'T FORGET your KEY)
:X                  : you will be prompted for a key
:h :X
-----
" modeline (make a file readonly etc) must be in first/last 5 lines
// vim:noai:ts=2:sw=4:readonly:
" vim:ft=html:      : says use HTML Syntax highlighting
:h modeline
-----
" Creating your own GUI Toolbar entry
amenu Modeline.Insert\ a\ VIM\ modeline <Esc><Esc>ggOvim:ff=unix ts=4 ss=4<CR>vim60:fdm=marker<esc>gg
-----
" A function to save word under cursor to a file
function! SaveWord()
    normal yiw
    exe '!echo '.@0.' >> word.txt'
endfunction
map ,p :call SaveWord()
-----
" function to delete duplicate lines
function! Del()
    if getline(".") == getline(line(".") - 1)
        norm dd
    endif
endfunction

:g/^/ call Del()
-----
" Digraphs (non alpha-numeric)
:digraphs          : display table
:h dig             : help
i<C-K>e'           : enters é
i<C-V>233          : enters é (Unix)
i<C-Q>233          : enters é (Win32)
ga                 : View hex value of any character
#Deleting non-ascii characters (some invisible)
:%s/[\x00-\x1f\x80-\xff]/ /g : type this as you see it
:%s/[<C-V>128-<C-V>255]/ /gi : where you have to type the Control-V
:%s/[\x{00000000}-\x{000000ff}]/ /gi : Should see a black square & a dotted y
:%s/[<C-V>128-<C-V>255<C-V>01-<C-V>31]/ /gi : All pesky non-asciis
:exec "norm /[\x00-\x1f\x80-\xff]/" : same thing
#Pull a non-ascii character onto search bar
yl/<C-R>"
/[^a-zA-Z0-9_[:space:][:punct:]] : search for all non-ascii
-----
" All file completions grouped (for example main.c)
:e main_<tab>          : tab completes
gf                    : open file under cursor (normal)
main_<C-X><C-F>        : include NAME of file in text (insert mode)
-----
" Complex Vim
" swap two words
:%s/\<(on|off)\>/\=strpart("offon", 3 * ("off" == submatch(0)), 3)/g
" swap two words
:vnoremap <C-X> <Esc>`.``gvP``P
" Swap word with next word
nmap <silent> gw      "_yw:s/(\%#\w\+)\(\_W\+)\(\w\+)\(/3\2\1/<cr><c-o><c-l> [N]
-----
" Convert Text File to HTML
:runtime! syntax/2html.vim : convert txt to html
:h 2html
-----
" VIM has internal grep
:grep some_keyword *.c    : get list of all c-files containing keyword
:cn                        : go to next occurrence
-----
" Force Syntax coloring for a file that has no extension .pl
:set syntax=perl
" Remove syntax coloring (useful for all sorts of reasons)
:set syntax off
" change coloring scheme (any file in ~vim/vim?/colors)
:colorscheme blue
:colorscheme morning      : good fallback colorscheme *N*
" Force HTML Syntax highlighting by using a modeline
# vim:ft=html:
" Force syntax automatically (for a file with non-standard extension)
au BufRead,BufNewFile */Content.IE?/* setfiletype html
" make cursor standout against highlighted text for vim terminal (linux) [N]
:highlight search term=reverse ctermbg=5
:highlight Normal guibg=white      : restore white background [N]
-----
:set noma (non modifiable)      : Prevents modifications

```

```

:set ro (Read Only) : Protect a file from unintentional writes
-----
" Sessions (Open a set of files)
gvim file1.c file2.c lib/lib.h lib/lib2.h : load files for "session"
:mksession : Make a Session file (default Session.vim)
:mksession MySession.vim : Make a Session file named file [C]
:q
gvim -S : Reload all files (loads Session.vim) [C]
gvim -S MySession.vim : Reload all files from named session [C]
-----
#tags (jumping to subroutines/functions)
taglist.vim : popular plugin
:Tlist : display Tags (list of functions)
<C-J> : jump to function under cursor
-----
" columnise a csv file for display only as may crop wide columns
:let width = 20
:let fill=' ' | while strlen(fill) < width | let fill=fill.fill | endwhile
:%s/\([^\;]*\);\\=\\=strpart(submatch(1).fill, 0, width)/ge
:%s/\\s\\+$/\\ge
" Highlight a particular csv column (put in .vimrc)
function! CSVH(x)
    execute 'match Keyword /^\\([^\;]*\\)\\{'.a:x.'\\}zs[^,]*/'
    execute 'normal ^'.a:x.'.f,'
endfunction
command! -nargs=1 Csv :call CSVH(<args>)
" call with
:Csv 5 : highlight fifth column
-----
zf1G : fold everything before this line [N]
" folding : hide sections to allow easier comparisons
zf} : fold paragraph using motion
v}zf : fold paragraph using visual
zf'a : fold to mark
zo : open fold
zc : re-close fold
" also visualise a section of code then type zf [N]
:help folding
zfG : fold everything after this line [N]
-----
" displaying "non-asciis"
:set list
:h listchars
-----
" How to paste "normal vim commands" w/o entering insert mode
:norm qqy$jq
-----
" manipulating file names
:h filename-modifiers : help
:w % : write to current file name
:w %:r.cfm : change file extension to .cfm
:!echo %:p : full path & file name
:!echo %:p:h : full path only
:!echo %:t : filename only
:reg % : display filename
<C-R>% : insert filename (insert mode)
"%p : insert filename (normal mode)
/<C-R>% : Search for file name in text
-----
" delete without destroying default buffer contents
"_d : what you've ALWAYS wanted
"_dw : eg delete word (use blackhole)
-----
" pull full path name into paste buffer for attachment to email etc
nnoremap <F2> :let @*=expand("%:p")<cr> :unix
nnoremap <F2> :let @*=substitute(expand("%:p"), "/", "\\ ", "g")<cr> :win32
-----
" Simple Shell script to rename files w/o leaving vim
$ vim
:r! ls *.c
:%s/\\(.*\\).c/mv & \\1.bla
:w !sh
:q!
-----
" count words/lines in a text file
g<C-G> # counts words
:echo line("'b")-line("'a") # count lines between markers a and b [N]
:'a,'bs/^//n # count lines between markers a and b
:'a,'bs/somestring//gn # count occurrences of a string
:s/,/,/gn # count commas in a line (csv/parameters)
-----
" example of setting your own highlighting
:syn match DoubleSpace " "
:hi def DoubleSpace guibg=#e0e0e0
-----
" reproduce previous line word by word
imap ] @@@<ESC>hhkyWjl?@@@<CR>P/@@@<CR>3s
nmap ] i@@@<ESC>hhkyWjl?@@@<CR>P/@@@<CR>3s
" Programming keys depending on file type

```

```

:autocmd bufenter *.tex map <F1> :!latex %<CR>
:autocmd bufenter *.tex map <F2> :!xdvi -hush %<.dvi&<CR>
" allow yanking of php variables with their dollar [N]
:autocmd bufenter *.php :set iskeyword+=\$
-----
" reading Ms-Word documents, requires antiword (not docx)
:autocmd BufReadPre *.doc set ro
:autocmd BufReadPre *.doc set hlsearch!
:autocmd BufReadPost *.doc %!antiword "%"
-----
" a folding method
vim: filetype=help foldmethod=marker foldmarker=<<<,>>>
A really big section closed with a tag <<<
--- remember folds can be nested ---
Closing tag >>>
-----
" Return to last edit position (You want this!) [N]
autocmd BufReadPost *
    \ if line("'\"") > 0 && line("'\"") <= line("$") |
    \   exe "normal! g`\"" |
    \ endif
-----
" store text that is to be changed or deleted in register a
"act<                               : Change Till < [N]
-----
"installing/getting latest version of vim on Linux (replace tiny-vim) [N]
yum install vim-common vim-enhanced vim-minimal
-----
# using gVIM with Cygwin on a Windows PC
if has('win32')
source $VIMRUNTIME/mswin.vim
behave mswin
set shell=c:\\cygwin\\bin\\bash.exe shellcmdflag=-c shellxquote=\"
endif
-----
" *Just Another Vim Hacker JAVH*
vim -c ":%s%*%Cyrnfr)fcbafbe[Oenz(Zbbyranne%|:%s)[[()])~)Ig|norm Vg?"
-----
vim:tw=78:ts=8:ft=help:norl:
__END__
-----
"Read Vimtips into a new vim buffer
:silent r ! lynx -dump "http://zzapper.co.uk/vimtips.html" [N]
" read webpage source html into vim
gvim http://www.zzapper.co.uk/vimtips.html &
-----
updated version at http://www.zzapper.co.uk/vimtips.html
-----
Please email any errors, tips etc to
vim@rayninfo.co.uk
" Information Sources
-----
www.vim.org
Vim Wiki *** VERY GOOD *** [N]
Vim Use VIM newsgroup [N]
comp.editors
groups.yahoo.com/group/vim "VIM" specific newsgroup
VIM Webring
VimTips PDF Version (PRINTABLE!)
Vimtips in Belarusian
-----
" : commands to neutralise < for HTML display and publish
" use yy@ to execute following commands
:w!|sav! vimtips.html|:/^__BEGIN__/,/^__END__/_s#<#<#g|:w!|:!vimtipsftp
-----

```