```
Vim Cheat Sheet
---------------

.Quitting
  ZZ - exit, saving changes
  ZQ - exit, ignore changes

.Help
  :help' or ':h - Keyword
  :viusage - A full Cheet Sheet

.Files
  :w file - "Save" write to file
  :sav file2 - "Save As" & edit file2
  :w file.bak - save a backup copy
  :r file - read file in after line
  :Explore - open vim file explorer
  vim -O file1 file2 - open 2 files split
  vim -y - open in click and type mode
  vim file1 file2 - open 2 arg files
  :args - see list of open args
  :argadd t.txt - add to args list
  :argdelete t.txt - delete arg(s)
  :argedit t.txt - add to args list
  :fir - edit <-first args list file
  :la - edit last-> args list file
  :n - goto-> next file
  :1n - go one file <-right
  :N - goto <-previous file
  :1N - go one file <-left
  :e file.txt - open file in vim

.File Tabs
  vim -p file1 file2 - open 2 tabs
  :Texplore - open file explorer tab
  :mksession work.vim - save session
  vim -S work.vim - open session
  :source work.vim - open session
  :mks! - save changes to session
  :tabe file.txt - open new tab
  :tabonly - close all but current
  :tabn - goto next tab file
  :tabp - goto previous tab file
  gt - goto next tab file (cycle)
  2gt - takes you to 2nd tab

.Split Window
  :split file - split, load file
```

```
  :vsplit file - vertical split
  ctrl-w up - move up a window
  ctrl-w left - move left a window
  ctrl-w ctrl-w - cycle windows
  ctrl-w _ - maximize vertically
  ctrl-w | - maximize horizontally
  ctrl-w = - make all equal size
  :sview file - split readonly
  :close - close current window
  :Sexplore - split, open file explorer
  :Vexplore - vertical file explorer
  :Vexplore! - open f. explorer on right
 Resizing Split Windows:
  ctrl-w 2 - - drop window height
  ctrl-w 2 + - rise window height
  ctrl-w 2 < - narrow window width
  ctrl-w 2 > - widen window width
 Moving Split Windows:
  ctrl+w R - rotate windows up/left
  ctrl+w r - rotate windows down/right
  ctrl+w L - Move current window right
  ctrl+w H - Move current window left
  ctrl+w J - Move current window down
  ctrl+w K - Move current window to top
  :help window-moving - to learn more

.Settings
  :set nu - numbered lines
  :set nonu - no numbered lines
  :set rnu - relative numbered lines
  :set nornu - no rel. numbered lines
  :colo morning - set colorscheme
  :syntax on - syntax highlighting on
  :set ruler - show current position
  :setlocal spell! - spell check on
  :setlocal nospell! - spell check off
  :set so=2 - top & bottom padding
  :set ignorecase - when searching
  :noh - unsets last search pattern
  :set magic - for regular expressions
  :set ai - auto indent
  :set si - smart indent
  :set wrap - wrap lines
  :set tabstop=4 - ␣ width of tab char
  :set softtabstop=0 - tabstop width
  :set expandtab - make tab key spaces
  :set noexpandtab - make tab key tabs
  :set shiftwidth=4 - width of indent
```

```
  :set smarttab - auto indent on
  :help smarttab - more details

.Inserting text
  i - insert
  I - insert at line beginning
  a - append
  A - append end of line
  o - open on lower line
  O - open on Upper line
  s - substitute character

.Insert Mode Shortcuts
  ctrl+w - delete word b4 cursor
  ctrl+u - delete line b4 cursor
  ctrl+n - autocomplete

.Text Objects
  ca" - change all quotes & "quoted"
  ci" - change inside quotes"
  4cw - change 4 words
  caw - change all word
  cit - change inside HTML tag(s)
  diw - delete in word
  di" - delete text inside quotes"
  da" - delete all quotes & "quoted"
  df[[:space:]] - delete to space include space
  dt[[:space:]] - delete till space
  va" - vis. select all quotes & "quoted"
  vi" - vis. select text inside quotes"
  ya" - yank all quotes & "quoted"
  yi" - yank text inside quotes"

.Navigation
          ^
          k
       <h   l>
          j
          v
  8j - move 8 lines down
  gj - down 1 wrapped line
  8gg - goto line 8
  :8 - goto line 8
  50% - goto middle of doc
  % - focus on the opposite brace
  gi - back to last insert
  H - high on the screen
  M - middle of the screen
```

```
  L - bottom of the screen
  zt - work on top of screen
  zz - Center workspace
  zb - work on bottom of screen
  ^ - first non-blank char
  g_ - last non-blank char
  ctrl+y & ctrl+e - scroll ONE line
  ctrl+u & ctrl+d - scroll HALF-page
  ctrl+b & ctrl+f - scroll FULL-page
  ctrl+O - Retrace moves backwards
  ctrl+I - Retrace moves forwards
  /pat - iterate all matching words
  iterate words same as current
     N - next one up
     n - next one down
  # - goto previous match of current
  G - goto to end of file
  fc - go forward to c
  Fc - go backward to c
  w - goto> next word
  W - goto> next word÷spaces
  e - goto> end of word
  E - goto> end of word÷spaces
  b - <goto beginning of word
  B - <goto begin of word÷spaces
  0 - goto beginning of line
  $ - goto end of line
  ma - sets local mark
  mA - sets global to path mark
  a - goto mark a
  :marks - list of all marks

.Change
  cc - change entire line
  cw - change word
  cW - change all to next space
  C - change to the end of line

.Delete
  :1,$d - delete all lines
       or :%d or ggdG
  x - delete> char to the right
  X - <delete char to the left
  D - delete to the end of line
  dd - delete current line
  "_d - black hole delete
  :d - delete current line
  dw - delete word
```

```
  dW - delete all to next space
  df? - delete> through first "?"
  dt? - delete> to first "?"
  dF? - <delete back through first "?"
  dT? - <delete back to first "?"

.Replace
  r - replace char not insert
  R - replace chars not insert
  :s/pattern/string/flags
  g - flag, replace all occurrences
  c - flag, confirm replaces

.Copy & Paste
  yy - yank/copy line
  5yy - yank 5 lines
  :12,16y - yank 5 lines
  :1,8t. - duplicate lines 1-8
  p - put/paste on lower line
  P - put/paste on Upper line
  20p - put same text twenty times

.Visual Mode
  v - enter visual mode
  V - enter visual Line mode
  ctrl+v - enter vis. Block mode
  ggvG - visual select all
  y - yank/copy selected
  o - cursor to opposite end
  O - cursor to opposite side
  gv - restore previous selection
  1v - selects area = to the last
  r - replaces select with char
  R - del. select, starts i mode
  va" - vis. select all "quoted"
  vi" - vis. select inside quotes"

Visual Block Routines
__Replaces block by the same text__
 select block, press c, change 1st
 line, press <Esc> twice, replaces
 block by the same text in 1st line.
 This also works with C or I or A

__Replaces block from clipboard__
 select what you want to put elsewhere,
 press d, select the code that you want
 it to replace, press p
```

__Paste over multiple areas(selections)__
 select what you want to put elsewhere,
 press y, select each area that you want
 it to replace, press "_d, press P
 one area at a time

.Format
  J - (Join) delete that line ending
  == - auto format indentation
  =i{ - indent inside {}
  gg=G - fix the indentation global
  :66,70s/^/# / - Commenting
  :66,70s/^#/ - Uncommenting
  :12,20> - Indent
  :12,20>>> - Indents 3 times
  5>> - Indents 5 lines
  5>>.. - Repeats 5>> twice

.Special
  :h - Great help info!
  ctrl+] - goto tag when caret is over it
  ctrl+G - display cursor location
  :tag tagname - jump to the tag
  :%retab - replace all tabs with spaces
  xp - transpose two letters
  & - repeat last :s cmd
  :1,8t. - duplicate lines 1-8
  :1,8!nl - line 1-8 number a list
  :%!nl -ba - Insert line numbers
  sort - sort the whole doc
  22,33sort - sort line 22-33
  :'a,.sort - from marker a, to caret
  :%sort! - sort in reverse
  :%sort u - remove duplicate lines
  :sort n - do numeric sort
  :sort i - case is ignored
  :help sort - more options
  ~ - toggle case
  . - repeat last cmd
  :! - drop to external cmd
  !!ls - insert output of cmd
  == - duplicate operator

.Code Folding
  zf - fold visible line selected
  2zfj - fold 3 lines down
  za - unfold last fold

```
  zR - unfold all

.Undo
  u - undo last change
  U - undo all changes to line

.Recording a macro
  qd - start recording to register d
  ... - your complex series of commands
  q - stop recording
  @d - execute your macro
  @@ - execute your macro again

.String Substitution
  :help pattern-overview
  :help sub-replace-special
  :help sub-replace-expression
  :set noignorecase
   make searches case sensitive
   (the default).
  :%s/foo/bar/g
   Find each occurrence of 'foo'
   (in all lines), and
   replace it with 'bar'.
  :s/foo/bar/g
   Find each occurrence of 'foo'
   (in the current line only),
   and replace it with 'bar'.
  :%s/foo/bar/gc
   Change each 'foo' to 'bar',
   but ask for confirmation first.
  :%s/\<foo\>/bar/gc
   Change only whole words exactly
   matching 'foo' to 'bar';
   ask for confirmation.
  :%s/foo/bar/gci
   Change each 'foo' (case insensitive
   due to the i flag) to 'bar';
   ask for confirmation.
  :%s/foo\c/bar/gc
   is the same because \c makes the
   search case insensitive. This may
   be wanted after using
   :set noignorecase to make searches
   case sensitive (the default).
  :%s/foo/bar/gcI
   Change each 'foo' (case sensitive
   due to the I flag) to 'bar';
```

```
   ask for confirmation.
  :%s/foo\C/bar/gc
   is the same because \C makes the
   search case sensitive. This may
   be wanted after using
   :set ignorecase to make searches
   case insensitive.

Use the c flag
When you need to confirm for each match what to do.
Vim will output something like: replace with foobar (y/n/a/q/l/^E/^Y)?
y - yes substitute this match
n - no skip this match
a - substitute this and ("all" remaining matches)
q - quit the command
l - to substitute this match and quit (think of "last")
^E - scroll the screen up by holding the Ctrl key and pressing E
^Y - scroll the screen down by holding the Ctrl key and pressing Y.
However, the last two choices are only available, if your Vim is a normal,
big or huge build or the insert_expand feature was enabled at compile time
(look for +insert_expand in the output of :version).

.Search range:
:s/foo/bar/g      Change each "foo" to "bar" in the current line.
:%s/foo/bar/g     Change each "foo" to "bar" in all the lines.
:5,12s/foo/bar/g  Change each "foo" to "bar" for all lines from line 5 to line 12
(inclusive).
:'a,'bs/foo/bar/g Change each "foo" to "bar" for all lines from mark a to mark b
inclusive.
:'<,'>s/foo/bar/g When compiled with +visual, change each "foo" to "bar" for all
lines within a visual selection.
                  Vim automatically appends the visual selection range ('<,'>)
for any ex command when you select
                  an area and enter :.
:.,$s/foo/bar/g    Change each "foo" to "bar" for all lines from the current
line (.) to the last line ($) inclusive.
:.,+2s/foo/bar/g   Change each "foo" to "bar" for the current line (.) and the
two next lines (+2).
:g/^baz/s/foo/bar/g Change each "foo" to "bar" in each line starting with "baz".

.When searching:
., *, \, [, ^, and $ are metacharacters.
+, ?, |, &, {, (, and ) must be escaped to use their special function.
\/ is / (use backslash + forward slash to search for forward slash)
\t is tab, \s is whitespace (space or tab)
\n is newline, \r is CR (carriage return = Ctrl-M)
After an opening [, everything until the next closing ] specifies a collection.
(see :help collection)
```

Character ranges can be represented with a -;
for example a letter a, b, c, or the number 1 can be matched with [1a-c].
Negate the collection with [^ instead of [;
For example [^1a-c] matches any character except a, b, c, or 1.
\{#\} is used for repetition.
/foo.\{2\} will match foo and the two following characters.
The \ is not required on the closing } so /foo.\{2} will do the same thing.
\(foo\) makes a backreference to foo. Parenthesis without escapes are literally matched.
Here the \ is required for the closing \).

.When replacing:
\r is newline, **\n** is a null byte (0x00).
\& is ampersand (& is the text that matches the search pattern).
\0 inserts the text matched by the entire pattern
\1 inserts the text of the first backreference.
\2 inserts the second backreference, and so on.

You can use other delimiters with substitute:
:s#http://www.example.com/index.html#http://example.com/#

Save typing by using \zs and \ze to set the start and end of a pattern.
For example, instead of:
:s/Copyright 2007 All Rights Reserved/Copyright 2008 All Rights Reserved/
Use:
:s/Copyright \zs2007\ze All Rights Reserved/2008/

.Using the current word or registersEdit
:%s//bar/g
Replace each match of the last search pattern with 'bar'.
For example, you might first place the cursor on the word foo then press * to search for that word.
The above substitute would then change all words exactly matching 'foo' to 'bar'.
:%s/foo/<c-r><c-w>/g
Replace each occurrence of 'foo' with the word under the cursor.
<c-r><c-w> means that you press Ctrl-R then Ctrl-W.
The word under the cursor will be inserted as though you typed it.
:%s/foo/<c-r><c-a>/g
Replace each occurrence of 'foo' with the WORD under the cursor (delimited by whitespace).
<c-r><c-a> means that you press Ctrl-R then Ctrl-A.
The WORD under the cursor will be inserted as though you typed it.
:%s/foo/<c-r>a/g
Replace each occurrence of 'foo' with the contents of register 'a'.
<c-r>a means that you press Ctrl-R then a.
The contents of register 'a' will be inserted as though you typed it.
:%s/foo/<c-r>0/g

Same as above, using register 0 which contains the text from the most recent yank command.
Examples of yank (copy) commands are yi(which copies the text inside parentheses around the cursor,
and y$ which copies the text from the cursor to the end of the line. After a yank command which did
not specify a destination register, the copied text can be entered by pressing Ctrl-R then 0.
:%s/foo/\=@a/g
Replace each occurrence of 'foo' with the contents of register 'a'.
\=@a is a reference to register 'a'.
The contents of register 'a' is not shown in the command.
This is useful if the register contains many lines of text.
:%s//<c-r>//g
Replace each match of the last search pattern with the / register (the last search pattern).
After pressing Ctrl-R then / to insert the last search pattern
(and before pressing Enter to perform the command), you could edit the text to make any required change.
:%s/<c-r>*/bar/g
Replace all occurrences of the text in the system clipboard (in the * register) with 'bar' (see next example if multiline).
On some systems, selecting text (in Vim or another application) is all that is required to place that text in the * register.
:%s/<c-r>a/bar/g
Replace all occurrences of the text in register 'a' with 'bar'.
<c-r>a means that you press Ctrl-R then a. The contents of register 'a' will be inserted as though you typed it.
Any newlines in register 'a' are inserted as ^M and are not found.
The search works if each ^M is manually replaced with '\n' (two characters: backslash, 'n').
This replacement can be performed while you type the command:
:%s/<c-r>=substitute(@a,"\n",'\\n','g')<CR>/bar/g
The "\n" (double quotes) represents the single character newline; the '\\n' (single quotes) represents two backslashes
followed by 'n'. The substitute() function is evaluated by the <c-r>= (Ctrl-R =) expression register; it replaces each
newline with a single backslash followed by 'n'.
The <CR> indicates that you press Enter to finish the = expression.
:%s/<c-r>0/bar/g
Same as above, using register 0 which contains the text from the most recent yank command.

.Additional Examples
:%s/foo/bar/
On each line, replace the first occurrence of "foo" with "bar".
:%s/.*\zsfoo/bar/
On each line, replace the last occurrence of "foo" with "bar".

```
:%s/\<foo\>//g
On each line, delete all occurrences of the whole word "foo".
:%s/\<foo\>.*//
On each line, delete the whole word "foo" and all following text (to end of
line).
:%s/\<foo\>.\{5}//
On each line, delete the first occurrence of the whole word "foo" and the
following five characters.
:%s/\<foo\>\zs.*//
On each line, delete all text following the whole word "foo" (to end of line).
:%s/.*\<foo\>//
On each line, delete the whole word "foo" and all preceding text (from beginning
of line).
:%s/.*\ze\<foo\>//
On each line, delete all the text preceding the whole word "foo" (from beginning
of line).
:%s/.*\(\<foo\>\).*/\1/
On each line, delete all the text preceding and following the whole word "foo".
:%s/\<foo\(bar\)\@!/toto/g
On each line, replace each occurrence of "foo" (which starts a word and is not
followed by "bar") by "toto".
:s/^\(\w\)/\u\1/
If the first character at the beginning of the current line only is lowercase,
switch it to uppercase using \u (see switching case of characters).
:%s/\(.*\n\)\{5\}/&\r/
Insert a blank line every 5 lines.
The pattern searches for \(.*\n\) (any line including its line ending) repeated
five times (\{5\}).
The replacement is & (the text that was found), followed by \r (newline).
:%s/\<foo\(\a*\)\>/\=len(add(list, submatch(1)))?submatch(0):submatch(0)/g
Get a list of search results. (the list must exist)
Sets the modified flag, because of the replacement, but the content is unchanged.
Note: With a recent enough Vim (version 7.3.627 or higher), you can simplify this
to:
:%s/\<foo\(\a*\)\>/\=add(list, submatch(1))/gn
This has the advantage, that the buffer won't be marked modified and no extra
undo state is created.
The expression in the replacement part is executed in the sandbox and not allowed
to modify the buffer.

vim search and append/modify
----------------------------
.Input:
The quick brown fox jumps over the lazy dog!
Vim String Substitution Command:
:%s/^The quick brown\(.\{-}\)lazy dog!$/The old red\1leg-hold trap!/g
The .\{-} will match "any number of characters before the "lazy dog!",
non-greedily" (:help /\{) The \( and \) mark it as a captured group
```

```
that will then be used in the substitution as \1 (the first captured group).
Running this gives you:
.Output:
The old red fox jumps over the leg-hold trap!

.Input
  **ZQ** -exit, ignore changes
Vim String Substitution Command:
:%s/^\s\+\*\*\(.\{-}\)\*\* \`\(.\{-}\)\`/printf \"${Command}   \1${Description} \2${NC}\\n\"/g
Modified String after running the Vim substitution:
.Output
printf "${Command}   ZQ${Description} -exit, ignore changes${NC}\n"
HOW IT WORKS:
: ................. command will follow
% ................. matches all lines in the whole file
s/ ................ substitution string
^ ................. the beginning of the line
\s ................ one space, the first in this case
\+ ................ any number of the preceding character
\*\* .............. literally match **
\( expresion \) ..... save whatever matches the inner expression to the register
\1
.\{-} ............. match any number of any kind of character
\*\* \` ........... literally match ** `
\( expresion \) ..... save whatever matches the inner expression to the register
\2
\` ................ literally match `
/replacement string/  Substitute every string that matches the above criteria
with everything that is between the //
like this: ......... /printf \"${Command}   \1${Description} \2${NC}\\n\"/
g ................. last but not least g is the global flag, each occurrence in
the line is changed,
                     rather than just the first


.Recording a macro
Each register is identified by a letter a to z.
To enter a macro, type:
q<letter><commands>q
To execute the macro <number> times (once by default), type:
<number>@<letter>
So, the complete process looks like:
qd - start recording to register d
... - your complex series of commands
q - stop recording
@d - execute your macro
@@ - execute your macro again
```