vim

Vim??vi??Emacs?????Unix?
??????????? Vim???????????·???1991????

- Vim ????
- Vim FAQ
- vimrc
- Cheat Sheets

Vim ????

????

?????????????	????	????
i	?????????????	? a ??????????????????????????
a	?????????????	
o	??????????	
h j k l	????????h--??, j--??, k--??, l--??	??????????????????????????
:1 : 22	??????? ?????22?	?????????????
gg	??????	
G	??????	?????????????????
0 / $	??????/??	
w / W	?????????/??	????
b / B	?????????/??	????
x	??????	
J (Shift + j)	?????????	???????
{	?????????	
}	?????????	
:wq	?????	

:w new.txt	????? new.txt	
:w !sudo tee %	? sudo ????	????????????? sudo ???????
:q!	?????	??????????????????????????? ?
:set nu!	??????	
:set hlsearch	Highlight ??????	??????????
:set cuc!	??????	?? json, yaml ??
: syntax on	?? Hihjlight ??	
d dd	??????? ???????	
10dd	??????10?(???)	
y yy	??????? ???????	
3yy	??????3?(???)	
p	??	
/keyword	????	
/\KeyWord	????(?????)	
/S[Uu]SE	???????????	
*	???????????	
:%s/old/new/g :%s/old/new/gc	???????old????????new?? gc: ????	
:s#/usr/local/bin#/usr/sbin#g	? # ??????	
:10,30 s/old/new/g	?10-30??? old ? new	

:%s/old1|OLD2/new/gc	???? old1?OLD2 ? new	
:/<table>/,/<\/table>/g/^$/d	?? <table> ... </table> ?????	
:%s/\t/ /g	?? Tab ?4???	??????
/\s\+$	?????????? Highlight	????? :set hlsearch ??, not support AIX
:%s/\s\+$//	???????????	not support AIX
~	?????	
:%s/[ctrl+V and ctrl+M]//g	?????? ^M	
ggVG	????	?? Ctrl a
dG	??????????	
dgg	??????????	
:1,$d	??????	
:1,22d	?? 1-22 ?	
:23,$d	?? 23 ???????	
:! ls -l	???? SHELL ??	:!! ?????
:vsplit my.file	??????????? my.file	ctrl + w + w ????
:split my.file	??????	ctrl + w + w ????
:sort :sort! :sort u :sort i :sort n	?? ???? ???????? ???????? ?????	
u Ctrl + r	Undo Redo	

:g/^$/d :4,100g/^$/d :g/pattern/d :!g/pattern/d	??????? ?4-100?????? ???? pattern ?? ????? pattern ??	[range]g/pattern/cmd

?????

- ??????????'.
- ???????:terminal
  - :below terminal , :bel term
  - :vertical terminal , :vert term
  - ????: Ctrl + w + w
  - ????: Ctrl + w + N ???????: hi ?
- ????????:find your-script.sh ??? tab ????? * ???
  - :find *.sh

Visual mode

- ???????: V
- ??????: Shift + V
  - ????: Shift + V
  - ?????????: V
- ???????: Ctrl + V
  - ????????: I
  - ????????: A
  - ???????????: $
- ?????
  - ??: ?/??
  - ?????: w
  - ????????: f .
  - ?????: }
- ????: Esc
- ???????: y
- ????: p
- ????????: d *(Delete)*
- ????????: d > Move to target line > p
- ????????: r *(Replace)*
- ????????: c *(Change)*
- ????????: u *(Undo)*
- ????????("?')????: i + " ? '

????????

- ?? MyKeyWord? /MyKeyWord
- ?????????????*

- ??????`:set hlsearch`
- ?????????`:set incsearch`
- ??`{},[]()`??????`%`
- ?????????????`{`?`[{`
- ????????????????`zt` (Top), `zz` (Middle), `zb` (Bottom)
- ????????????`H` (Top), `M` (Middle), `L` (Last Line)
- ?????`Ctrl+b` (???), `Ctrl+f` (???)
- ?????`Ctrl+y` (??), `Ctrl+e` (??)

??????

- ????????? `/old` -> `cw` -> ?????? new -> ?????? `Esc` -> ????? `n` -> ?????? `.`
- ???????
  1. ????shift + `v`
  2. ????
  3. ??????`:s/old/new/g` (*???????????*)

????

```
# 複製指定的行號, 然後移到要貼上的行, 輸入 p 貼上
:<行號>yank

# 複製指定的行號, 貼上至目前行的下方. 使用 copy 或 t
:<行號>copy.
:<行號>t.

# 複製目前行下方第 10 行文字, 並貼上至目前行
:+10t.

# 複製目前行上方第 10 行文字, 並貼上至目前行
:-10t.

# 複製連續多行文字(33 ~ 44)至目前位置
:33,44t.
```

????????????????

1. Ctrl + `v`
2. ?????????
3. ??????
4. Shift + `I`
5. ????????
6. Esc ?????

??????????`;`

1. ?????????
2. Ctrl + `v`
3. ???????
4. Shift + `$`
5. Shift + `A`
6. ???????? `;`
7. Esc ?????

????

1. Shift + `v`
2. ????
3. `:norm i#` (??????)
4. `:norm x` (??????)

????

1. Shift + `v` (Visually Seleting)
2. ????
3. `:w /path/to/newfile` (Save as newfile)
4. `gv` (reselect the last visual selection)
5. `d` (delete that selection)

?????? (auto-complete)

1. ??????
2. ?? /etc/profile??????
3. Ctrl + `x` -> Ctrl `f`

Tutorials

- Youtube Channel - Vim (TheFrugalComputerGuy)
- VIM Colors Online
- VimTricks

Vim FAQ

Err: A line cannot be longer than 2048 characters.

???? VI ?????????? 2048 ????????????????????????

```
cut -c1-2047 before.log > after.log
```

?? 3X MB ???????

> ❝ ex: 0602-101 Out of memory saving lines for undo.

```
# the number must be less than 8 digits.
vi -y 99999999 your-file
```

??????

????????????(autoindent)????????????

Solution: ???????? `:set paste`

0602-140 The window is too small to display the current line.

AIX ??????????????????

Solution:

1. ? putty ?????????????? 9 ???
2. ?? Vim ?? Vi?

Unable to paste from clipboard on AIX

? Windows ??????? putty ?????????

Solution: `:set mouse=v`

vimrc

vimrc simple

```
"
" Fixed for Python virtualenv as the follow error.
" ModuleNotFoundError: No module named 'powerline'
"
python3 import sys;sys.path.append("/home/alang/.local/lib/python3.10/site-packages")
"
" Powerline with Vim
set laststatus=2
python3 from powerline.vim import setup as powerline_setup
python3 powerline_setup()
python3 del powerline_setup

"
syntax on               " Enable syntax highlighting
set number              " Show line numbers
set relativenumber      " Show relative line numbers
set tabstop=4           " Set tab width to 4 spaces
set shiftwidth=4        " Set indentation width to 4 spaces
set expandtab           " Use spaces instead of tabs
set cursorline          " Highlight the current line

" Moving Lines
" Ctrl-j: Move down, Ctrl-k: Move down
nnoremap <c-j> :m .+1<CR>==
nnoremap <c-k> :m .-2<CR>==
inoremap <c-j> <Esc>:m .+1<CR>==gi
inoremap <c-k> <Esc>:m .-2<CR>==gi
vnoremap <c-j> :m '>+1<CR>gv=gv
vnoremap <c-k> :m '<-2<CR>gv=gv

" Preserve last editing position
silent! source $VIMRUNTIME/defaults.vim
```

```
" Copy to Clipboard
" Prerequisites: apt install vim-gtk3
" Usage: Press v to select the lines, and then press y to copy
set clipboard=unnamedplus
```

vimrc advanced

```
"
" minimal vimrc with no (extra) plugins
"

"regular settings
"----------------
" ui
set number
set ruler
set wildmenu
set showcmd
set showmatch

" encoding/format
set encoding=utf-8
set fileformats=unix,dos,mac

" searching
set hlsearch
set incsearch
set ignorecase
set smartcase

" indent
set shiftwidth=4
set tabstop=4
set softtabstop=4
set autoindent

" allow syntax and filetype plugins
syntax enable
filetype plugin indent on
```

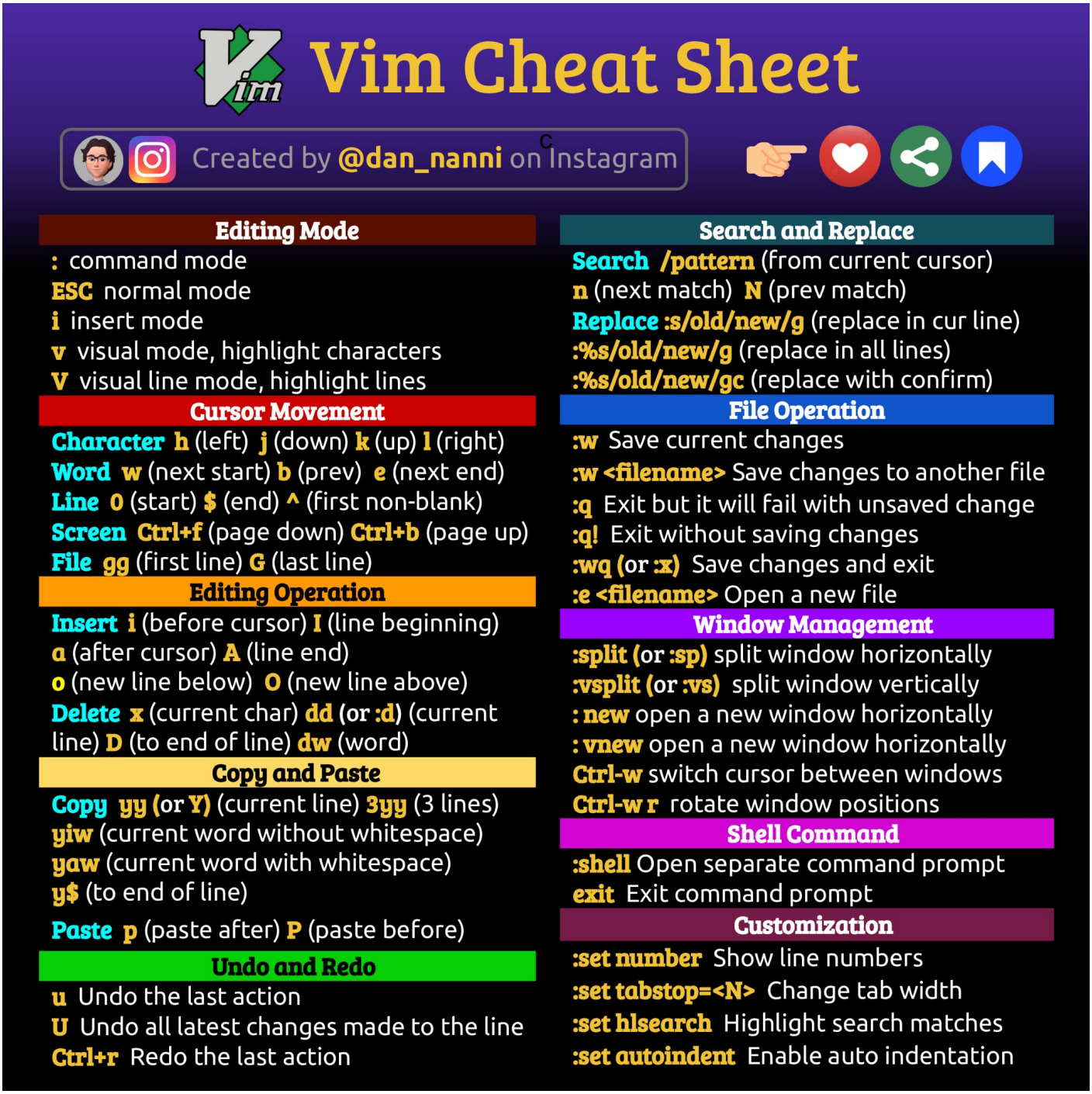
```
runtime macros/matchit.vim

" Moving Lines
" Ctrl-j: Move down, Ctrl-k: Move up
nnoremap <c-j> :m .+1<CR>==
nnoremap <c-k> :m .-2<CR>==
inoremap <c-j> <Esc>:m .+1<CR>==gi
inoremap <c-k> <Esc>:m .-2<CR>==gi
vnoremap <c-j> :m '>+1<CR>gv=gv
vnoremap <c-k> :m '<-2<CR>gv=gv

" autocmds
"---------
augroup general
    autocmd!
    "keep equal proportions when windows resized
    autocmd VimResized * wincmd =
    "save cursor position in a file
    autocmd BufReadPost * if line("'\"") > 1 && line("'\"")
                \ <= line("$") | exe "normal! g'\"" | endif
augroup END

augroup languages
    autocmd!
    autocmd BufNewFile,BufRead *.bash set syntax=sh
    autocmd FileType python xnoremap <leader>r <esc>:'<,'>:w !python3<CR>
    autocmd FileType go set noexpandtab
    autocmd FileType html :syntax sync fromstart
    autocmd FileType html,javascript,css,json,yaml,sh
                \ setlocal ts=2 sts=2 sw=2 expandtab
augroup END
```

Learning

- A Minimal Vimrc
- Configure Vim with vimrc

Cheat Sheets

PDF: vim-commands-cheat-sheet-by-pnap.pdf



:h operator / :h navigation

[operator][count][motion]

Operator	
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

starting cursor position

(use text-objects)

iw — iW — i(

Up-down motions (:h up-down-motions)

Key	Motion
gg	first line
^b	up 1 page
^u	up ½ page
k	up 1 line
j	down 1 line
^d	down ½ page
^f	down 1 page
G	last line

Left-right motions (:h left-right-motions)

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

Tabs and spaces

	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

Help (: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

Tags (: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

Keycodes (: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

Word motions (:h word-motions)

7 words

http://www.vimcheatsheet.com

1 WORD

SEARCHING (:h pattern-searches)

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

Marks (: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.

Normal mode commands

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 (file-searching)	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 (jump-motions)	^i	jump forward (jump-motions)
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

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

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

COOL INSERT MODE STUFF (:h insert.txt)

^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.

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

COMMAND-LINE MODE ONLY (:h cmdline.txt)

^f	edit using Normal mode	cmdwin
^r^w	insert word under cursor	cmdline-editing
^d	completion suggestions	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-\=)

File commands

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
:make	Run a compiler and enter quickfix mode (:h quickfix)
:!	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

Options (: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.

Buffers (: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

Windows (: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...
MML
Adam.
jdiems
Lin Qiu
@webjoe
grantok
storrgie
David O.
Digininja
Dave Remy
Ralph Bean
Arve Leken
Rajat Ghai
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
@andersosthus
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!

GoSquared — VI Help Sheet

Modes & Controls

Command Mode	ESC (commands preceded by :)
Insertion Mode	Entered on insertion or change

Starting VI (command line)

vi <filename>	Edit *filename*
vi -r <filename>	Edit last version of *filename* after crash
vi + n <filename>	Edit *filename* at line *n*
vi + <filename>	Edit *filename* at end of file
vi +/str <filename>	Edit *filename* at first occurance of *str*

In insertion mode the following should be preceded by ESC:

:w	Save
:x	Save & Exit
:q	Exit if no changes made
:q!	Exit & discard any changes

Cursor Navigation

h or ◄	Cursor left
j or ▼	Cursor down
k or ▲	Cursor up
l or ►	Cursor right
w	Next word
W	Next blank delimited word
b	Start of word
B	Start of blank delimited word
e	End of word
E	End of blank delimited word
(	Back a sentence
)	Forward a sentence
{	Back a paragraph
}	Forward a paragraph
0	Beginning of line
$	End of the line
1G	Start of file
G	End of file
:n	*n*th line of file
f<char>	Forward to *char*
F<char>	Back to *char*
H	Top of screen
M	Middle of screen
L	Bottom of screen
%	Matching bracket
gg	Start of document

Inserting Text

i	Insert before cursor
a	Append after cursor
I	Insert before line
A	Append after line
o	Add new line after current line
O	Add new line before current line
r	Overwrite one character
R	Overwrite many characters
:r <file>	Reads *file* and inserts it after this line
p	Put after the position or line
P	Put before the position or line
C	Rewrite the whole line

Deleting Text

x	Delete character to right of cursor
X	Delete character to left of cursor
D	Delete the rest of line
dd or :d	Delete current line
ndw	Deletes the next n words
ndb	Deletes the previous n words
ndd	Deletes n lines starting with current
:x,yd	Delete lines x through y
:r <file>	Reads *file* and inserts it after this line
d{nav_cmd}	Overwrite many characters
:r <file>	Reads *file* and inserts it after this line

Searching

/string	Search forward for string
?string	Search backwards for string
n	Go to next match
N	Go to previous match
:set ic	Ignore case while searching
:set noic	Case-sensitive searching
:set nu	Turn on line numbers
:x,yg/str	Search for str from line x to line y
:g/str/cmd	Run cmd on lines containing str
*	Search for next instance of current word
#	Search for last instance of current word

Replacing

:s/pt/str/flag	Replace pattern with string

Flags

g	Replace all occurrences of pattern
c	Confirm replaces
&	Repeat last :s command

Other

u
Undo last change

J
Join lines

nJ
Join next n lines

.
Repeat last command

U
Undo all changes to line

:N
Open split screen

v
Visual mode

ctrl + c
Escape insert mode