

Vim Cheat Sheet Chinese Version

Sponsor  Warp A new terminal focused on developer productivity

Vim Cheat Sheet

全局

:h[elp] **关键字** - 打开关键字帮助

:sav[eas] **文件名** - 另存为

:clo[se] - 关闭当前窗口

:ter[minal] - 打开一个shell窗口

K - 打开光标所在单词的man页面

Tip 在终端中运行 `vimtutor` 以学习最基础的 Vim 命令。

移动光标

h - 左移光标

j - 下移光标

k - 上移光标

l - 右移光标

gj - 下移光标(折行文本)

gk - 上移光标(折行文本)

H - 移动到当前页面顶部

M - 移动到当前页面中间

L - 移动到当前页面底部

w - 移动到下个单词开头

W - 移动到下个单词开头(单词含标点)

e - 移动到下个单词结尾

E - 移动到下个单词结尾(单词含标点)

b - 移动到上个单词开头

B - 移动到上个单词开头(单词含标点)

ge - 移动到上个单词结尾

gE - 移动到上个单词结尾(单词含标点)

‰ - 跳转到配对的符号(默认支持的配对符号组: '()'，'{}', '[]') - 在 vim 中使用 `:h matchpairs` 获得更多信息

0 - 移动到行首

^ - 移动到行首的非空白符

\$ - 移动到行尾

g_ - 移动到行内最后一个非空白符

gg - 移动到文件第一行

G - 移动到文件最后一行

5gg or 5G - 移动到第五行

gd - 跳转到局部定义

编辑文本

r - 替换当前字符

R - 在 **ESC** 按下之前, 替换多个字符

J - 将下一行合并到当前行, 并在两部分文本之间插入一个空格

gJ - 将下一行合并到当前行, 两部分文本之间不含空格

gwip - 重新调整段落

g~ - 大小写转换操作修饰符

gu - 小写操作修饰符

gU - 大写操作修饰符

cc - 将光标所在的行删除, 然后进入插入模式

c\$ or C - 将光标处到行尾删除, 然后进入插入模式

ciw - 将光标所在的单词删除, 然后进入插入模式

cw or ce - 从光标位置开始, 修改单词

s - 删除当前字符, 然后进入插入模式

S - 清空当前行, 然后进入插入模式(同 cc)

xp - 当前字符后移

u - 撤销

U - 撤销上一次的改动行的操作

Ctrl + r - 重做 (取消撤销)

. - 再次执行上个命令

选择文本 (可视化模式)

v - 进入可视化模式, 移动光标高亮选择, 然后可以对选择的文本执行命令(比 y- 复制)

V - 进入可视化模式(行粒度选择)

o - 切换光标到选择区开头/结尾

Ctrl + v - 进入可视化模式(矩阵选择)

0 - 切换光标到选择区的角

aw - 选择当前单词

ab - 选择被 O 包裹的区域(含括号)

aB - 选择被 {} 包裹的区域(含花括号)

at - 选择被 <> 标签包裹的区域(含<>标签)

ib - 选择被 () 包裹的区域(不含括号)

iB - 选择被 {} 包裹的区域(不含花括号)

it - 选择被 <> 标签包裹的区域(不含<>标签)

Esc or Ctrl + c - 退出可视化模式

Tip 也可以使用 (和 { 分别代替 b 和 B

宏

qa - 录制宏 a

q - 停止录制宏

@a - 执行宏 a

@@ - 重新执行上次执行的宏

剪切, 复制, 粘贴

yy - 复制当前行

2yy - 复制 2 行

yw - 复制当前单词

yiw - 复制光标处的单词

yaw - 复制光标处的单词及其前后的空格

y\$ or Y - 复制, 从光标位置到行末

p - 在光标后粘贴

P - 在光标前粘贴

gp - 在光标后粘贴并把光标定位到粘贴的文本之后

gP - 在光标前粘贴并把光标定位到粘贴的文本之后

dd - 剪切当前行

2dd - 剪切 2 行

dw - 剪切当前单词

diw - 删除光标处的单词

daw - 删除光标处的单词及其前后的空格

:3,5d - 删除 3 至 5 行

Tip 你也可以使用以下字符来指定范围:

e.g.

:\$d - 从当前行到文件末尾

:,1d - 从当前行到文件开头

:10,\$d - 第 10 行到文件末尾

:g/{pattern}/d - 删除所有包含 pattern 的行

:g!/{pattern}/d - 删除所有不包含 pattern 的行

d\$ or D - 剪切, 从光标位置到行末(同 D)

x - 剪切当前字符

文字缩进

>> - 将当前行向右缩进, 宽度由 shiftwidth 控制

<< - 将当前行向左缩进, 宽度由 shiftwidth 控制

>% - 向右缩进 () 或 {} 内的区域 (光标需置于括号上)

gD - 跳转到全局定义
fx - 移动到字符 x 下次出现的位置
tx - 移动到字符 x 下次出现的位置的前一个字符
Fx - 移动到字符 x 上次出现的位置
Tx - 移动到字符 x 上次出现的位置的后一个字符
; - 重复之前的f、t、F、T操作
, - 反向重复之前的f、t、F、T操作
】 - 移动到下一个段落(当编辑代码时则为函数 / 代码块)
{ - 移动到上一个段落(当编辑代码时则为函数 / 代码块)
zz - 移动屏幕使光标居中
zt - 移动屏幕使光标位于屏幕顶部
zb - 移动屏幕使光标位于屏幕底部
Ctrl + e - 向下移动屏幕一行(保持光标不动)
Ctrl + y - 向上移动屏幕一行(保持光标不动)
Ctrl + b - 向上滚动一屏
Ctrl + f - 向下滚动一屏
Ctrl + d - 向下滚动半屏
Ctrl + u - 向上滚动半屏

Tip 命令前追加数字表示命令的重复次数, 比如 **4j** 表示向下移动四行

插入模式 - 插入/追加文本

i - 从光标前开始插入字符
I - 从行首开始插入字符
a - 从光标后开始插入字符
A - 从行尾开始插入字符
o - 在当前行之下另起一行, 开始插入字符
O - 在当前行之上另起一行, 开始插入字符
ea - 从当前单词末尾开始插入
Ctrl + h - 在插入模式下, 删除光标前的字符
Ctrl + w - 在插入模式下, 删除光标前的单词
Ctrl + j - 在插入模式下, 另起一行
Ctrl + t - 在插入模式下, 向右缩进, 宽度由 shiftwidth 控制
Ctrl + d - 在插入模式下, 向左缩进, 宽度由 shiftwidth 控制
Ctrl + n - 在插入模式下, 在光标之前插入自动补全的下一个匹配项
Ctrl + p - 在插入模式下, 在光标之前插入自动补全的上一个匹配项
Ctrl + rx - 插入寄存器 x 的内容
Ctrl + ox - 暂时进入正常模式以发出一个正常模式命令 x。
Esc or **Ctrl + c** - 退出插入模式

可视化模式命令

> - 向右缩进
< - 向左缩进
y - 复制
d - 剪切
~ - 大小写切换
u - 将选中文本转换为小写
U - 将选中文本转换为大写

寄存器

:reg[isters] - 显示寄存器内容
"xy - 复制内容到寄存器 x
"xp - 粘贴寄存器 x 中的内容
"+y - 复制内容到系统剪贴板寄存器
"+p - 粘贴系统剪贴板寄存器的内容

Tip 寄存器被存储在 `~/.viminfo` 中, 在下次重启 vim 时仍会加载

Tip 特殊寄存器:

- 0** - 上次复制
- "** - 未命名寄存器, 上次复制或删除
- %** - 当前文件名
- #** - 轮换文件名
- *** - 剪贴板内容(X11 primary)
- +** - 剪贴板内容(X11 clipboard)
- /** - 上次搜索的pattern
- :** - 上次执行的命令
- .** - 上次插入的文本
- - 上次剪切的短于一行的文本
- =** - 表达式寄存器
- _** - 黑洞寄存器

标记

:marks - 标记列表
ma - 设置当前位置为标记 a
`a - 跳转到标记 a 的位置
y`a - 复制当前位置到标记 a 的内容
`0 - 跳转到上次 Vim 退出时的位置
`" - 跳转到上次编辑该文件时的位置
`. - 跳转到上次修改的位置
`^ - 跳转回上次跳转前的位置
:ju[mps] - 列出跳转历史记录
Ctrl + i - 跳转至跳转历史中较晚的位置
Ctrl + o - 跳转回跳转历史中较早的位置
:changes - 列出修改历史记录
g, - 跳转至修改历史中较晚修改的位置
g; - 跳转至修改历史中较早修改的位置
Ctrl +] - 跳转到当前光标位置对应的 tag

Tip 可以使用反引号(`)或单引号(')跳转至标记位置。使用单引号会跳转至该标记所在行首(首个非空白字符)。

<% - 向左缩进 O 或 {} 内的区域(光标需置于括号上)

>ib - 向右缩进 O 内的区域

>at - 向右缩进 <> 标签内的区域

3== - 自动缩进 3 行

=% - 自动缩进 O 或 {} 内的区域(光标需置于括号上)

=iB - 自动缩进 {} 内的区域(光标需置于括号上)

gg=G - 自动缩进整个缓冲区

lp - 粘贴并调整缩进至当前行

退出

:w - 保存
:w !sudo tee % - 使用 sudo 保存当前文件
:wq or **:x** or **ZZ** - 保存并退出
:q - 退出(修改未保存时警告)
:q! or **ZQ** - 不保存强制退出
:wqa - 保存所有标签页并全部退出

查找/替换

/pattern - 查找 pattern
?pattern - 向上查找 pattern
\vpattern - pattern 中的非字母数字字符被视为正则表达式特殊字符(不需转义字符)
n - 查找下一个
N - 查找上一个
:%s/old/new/g - 替换全部
:%s/old/new/gc - (逐个)替换
:noh[lssearch] - 移除搜索结果的高亮显示

多文件搜索

:vim[grep] /pattern/ {`{file}`} - 在多个文件中搜索 pattern

e.g. **:vim[grep] /foo/ **/***

:cn[ext] - 移动至下一个

:cp[revious] - 移动至上一个

:cope[n] - 打开搜索结果列表

:ccl[ose] - 关闭 quickfix 窗口

标签

:tabnew or **:tabnew {page.words.file}** - 在新标签中打开文件
Ctrl + wT - 将窗口变成标签
gt or **:tabn[ext]** - 切换到下一个标签
gT or **:tabp[revious]** - 切换到上一个标签
#gt - 切换到第 # 个标签
:tabm[ove] # - 移动标签到第 # 位(下标从 0 开始)
:tabc[lose] - 关闭当前标签
:tabo[nly] - 关闭其他标签
:tabdo command - 在所有标签中执行命令(例如 `:tabdo :w`)

多文件编辑

:e[dit] 文件名 - 新建缓冲区打开 filename
:bn[ext] - 切换到下个缓冲区
:bp[revious] - 切换到上个缓冲区
:bd[elete] - 关闭缓冲区
:b[uffer]# - 切换到第 # 个缓冲区
:b[uffer] file - 用文件名切换缓冲区
:ls or :buffers - 列出所有打开的缓冲区
:sp[lit] 文件名 - 新建缓冲区打开 filename 并水平分割窗口
:vs[plit] 文件名 - 新缓冲区打开 filename 并垂直分割窗口
:vert[ical] ba[ll] - 垂直分割窗口编辑所有缓冲区
:tab ba[ll] - 标签页编辑所有缓冲区
Ctrl + ws - 水平分割窗口
Ctrl + vv - 垂直分割窗口
Ctrl + ww - 在窗口间切换
Ctrl + wq - 关闭窗口
Ctrl + wx - 当前窗口与下一个窗口交换位置
Ctrl + w= - 令所有窗口高 & 宽一致
Ctrl + wh - 切换到左侧窗口
Ctrl + wl - 切换到右侧窗口
Ctrl + wj - 切换到下侧窗口
Ctrl + wk - 切换到上侧窗口
Ctrl + wH - 使游标所在视窗全高并移至最左(最左垂直视窗)
Ctrl + wL - 使游标所在视窗全高并移至最右(最右垂直视窗)
Ctrl + wJ - 使游标所在视窗全宽并移至最下(最下水平视窗)
Ctrl + wK - 使游标所在视窗全宽并移至最上(最上水平视窗)

Diff

zf - 定义折叠修饰符
zd - 删除光标位置的折叠
za - 展开 & 关闭光标位置的折叠
zo - 展开光标位置的折叠
zc - 关闭光标位置的折叠
zr - 展开同级的所有折叠
zm - 关闭同级的所有折叠
zi - 开启 & 关闭折叠功能
]c - 光标移至下一处差异
[c - 光标移至上一处差异
do or :diffg[et] - 将另一缓冲区中的差异合并至当前缓冲区
dp or :diffpu[t] - 将当前缓冲区中的差异推送至另一缓冲区
:diffthis - 令当前窗口成为 diff 模式的窗口之一
:dif[fupdate] - 强制刷新 diff 的高亮与折叠
:difo[ff] - 令当前窗口退出 diff 模式

Tip 折叠命令(e.g. **za**)只作用于当前级别。使用大写字母(e.g. **zA**)令命令作用于全部级别。

Tip 可以直接在终端运行 **vimdiff** 查看文件间的不同。也可以将该程序设为 **git difftool** 的选项之一。

Additional Resources

Languages

العربية
 ວັດລາ
 Català
 Čeština
 Dansk
 Deutsch
 English
 Esperanto
 Español
 Persian
 Suomi
 Français
 עברית
 Hrvatski
 Magyar
 Bahasa Indonesia
 Italiano
 日本語
 한국어^{한국어}
 ດີວິຈານ
 Nederlands
 Norsk
 ...

About the vim cheat sheet

This project aims to be one of the most accessible vim guides available. We made sure to support mobile, desktop, and other languages.

♥ Please consider sponsoring this project ♥!

You can read about how to contribute (and help improve) by viewing our [README](#). There you can see how to set up this project, or how to contribute a new language. Here is a big thank you to our [contributors](#)!

This project is licensed under [The MIT License \(MIT\)](#).

Other places to find this document

This document was embedded in [DuckDuckGo](#).

More resources

Interactive Vim tutorial: [Open Vim](#)
 Vim quick reference from Vim help pages: [quickref.txt](#)
 List of all Vim ex (:) commands: [ex-cmd-index](#)

Polski
Português - Brasil
Português - Portugal
Romana
Русский
ଓଡ଼ିଆ
Slovenčina
Svenska
ລາວ
Türkçe
Українська
Tiếng Việt
简体中文
中文(台灣)

Checkout the source on [Github](#)

version: 3.2.0

REF

<https://github.com/rtorr/vim-cheat-sheet>

Vim Cheet Sheet English Version

← DEVHINTS.IO [Edit](#)

Vim cheatsheet

Introduction

Vim is a very efficient text editor. This reference was made for Vim 8.0.

For shortcut notation, see :help key-notation.

Exiting

:q	Close file
:qa	Close all files
:w	Save
:wq / :x	Save and close file
ZZ	Save and quit
ZQ	Quit without checking changes

Exiting insert mode

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



Redesign the way you jam with FigJam AI.
ads via Carbon

Editing

Editing		Clipboard		Visual mode	
a	Append	x	Delete character	v	Enter visual mode
A	Append from end of line	dd	Delete line (Cut)	V	Enter visual line mode
i	Insert	yy	Yank line (Copy)	<C-V>	Enter visual block mode
o	Next line	p	Paste	In visual mode	
O	Previous line	P	Paste before	d / x	Delete selection
s	Delete char and insert	"*p / "+p	Paste from system clipboard	s	Replace selection
S	Delete line and insert	"*y / "+y	Paste to system clipboard	y	Yank selection (Copy)
c	Delete until end of line and insert			See Operators for other things you can do.	
r	Replace one character				
R	Enter Replace mode				
u	Undo changes				
<C-R>	Redo changes				

Navigating

Directions		Character		Window	
h	j k l	Arrow keys	fc	Go forward to character c	zz Center this line
<C-U> / <C-D>		Half-page up/down	Fc	Go backward to character c	zt Top this line
<C-B> / <C-F>		Page up/down			zb Bottom this line
Words		Document		Search	
b / w	Previous/next word	gg	First line	n Next matching search pattern	
ge / e	Previous/next end of word	G	Last line	N Previous match	
Line		:{number}	Go to line {number}	*	Next whole word under cursor
0 (zero)	Start of line	{number}G	Go to line {number}	#	Previous whole word under cursor
^	Start of line (after whitespace)	{number}j	Go down {number} lines		
\$	End of line	{number}k	Go up {number} lines		

Operators

Usage		Operators list		Examples	
Operators let you operate in a range of text (defined by motion). These are performed in normal mode.		d	Delete	Combine operators with motions to use them.	
d	w	y	Yank (copy)	dd	(repeat the letter) Delete current line
Operator	Motion	c	Change (delete then insert)	dw	Delete to next word
		>	Indent right	db	Delete to beginning of word
		<	Indent left	2dd	Delete 2 lines
		=	Autoindent	dip	Delete a text object (inside paragraph)
		g~	Swap case	(in visual mode) d	Delete selection
		gU	Uppercase	See: :help motion.txt	
		gu	Lowercase		
		!	Filter through external program		
		See :help operator			

Text objects

Usage

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

v	i	p
Operator	[i]nside or [a]round	Text object

Text objects

p	Paragraph
w	Word
s	Sentence
[({ <	A [](), or {} block
' " `	A quoted string
b	A block [(
B	A block in [{
t	A XML tag block

Examples

vip	Select paragraph
vipipipi	Select more
yip	Yank inner paragraph
yap	Yank paragraph (including newline)
dip	Delete inner paragraph
cip	Change inner paragraph
See Operators for other things you can do.	

Diff

gvimdiff file1 file2 [file3]	See differences between files, in HMI
------------------------------	---------------------------------------

Misc

Tab pages

:tabedit [file]	Edit file in a new tab
:tabfind [file]	Open file if exists in new tab
:tabclose	Close current tab
:tabs	List all tabs
:tabfirst	Go to first tab
:tablast	Go to last tab
:tabn	Go to next tab
:tabp	Go to previous tab

Folds

zo / zO	Open
zc / zC	Close
za / zA	Toggle
zv	Open folds for this line
zM	Close all
zR	Open all
zm	Fold more (foldlevel += 1)
zr	Fold less (foldlevel -= 1)
zx	Update folds
Uppercase ones are recursive (eg, zO is open recursively).	

Marks

`^	Last position of cursor in insert mode
`.	Last change in current buffer
`"	Last exited current buffer
`0	In last file edited
`..	Back to line in current buffer where jumped from
`..	Back to position in current buffer where jumped from
`[To beginning of previously changed or yanked text
`]	To end of previously changed or yanked text
`<	To beginning of last visual selection
`>	To end of last visual selection
ma	Mark this cursor position as a
`a	Jump to the cursor position a
'a	Jump to the beginning of the line with position a
d'a	Delete from current line to line of mark a
d'a	Delete from current position to position of mark a
c'a	Change text from current line to line of a
y`a	Yank text from current position to position of a
:marks	List all current marks
:delm a	Delete mark a
:delm a-d	Delete marks a, b, c, d
:delm abc	Delete marks a, b, c

Navigation

%	Nearest/matching {[()]} ([{}])
.	Repeat last command

Misc

<p>[([{ [<</p> <p>]) Next</p> <p>[m Previous method start</p> <p>[M Previous method end</p>	<p>Previous (or { or <</p> <p>Next</p> <p>Previous method start</p> <p>Previous method end</p>	<p>]p Paste under the current indentation level</p> <p>:set ff=unix Convert Windows line endings to Unix line endings</p>
Command line		
<p><C-O></p> <p><C-I></p> <p>gf</p>	<p>Go back to previous location</p> <p>Go forward</p> <p>Go to file in cursor</p>	<p><C-R><C-W> Insert current word into the command line</p> <p><C-R>" Paste from " register</p> <p><C-X><C-F> Auto-completion of path in insert mode</p>
Text alignment		
<p><C-A></p> <p><C-X></p>	<p>Increment number</p> <p>Decrement</p>	<p>:center [width] :right [width] :left</p> <p>See :help formatting</p>
Windows		
<p>z{height}<Cr></p>	<p>Resize pane to {height} lines tall</p>	<p><C-R>=128/2 Shows the result of the division : '64'</p> <p>Do this in insert mode.</p>
Tags		
<p>:tag Classname</p> <p><C-]></p> <p>g]</p> <p><C-T></p> <p><C-O> <C-I></p> <p>:tselect Classname</p> <p>:tjump Classname</p>	<p>Jump to first definition of Classname</p> <p>Jump to definition</p> <p>See all definitions</p> <p>Go back to last tag</p> <p>Back/forward</p> <p>Find definitions of Classname</p> <p>Find definitions of Classname (auto-select 1st)</p>	<p>:cq :cquit</p> <p>Works like :qa, but throws an error. Great for aborting Git commands.</p>
Exiting with an error		
<p>:spell</p>	<p>Turn on US English spell checking</p>	
<p>]s</p>	<p>Move to next misspelled word after the cursor</p>	
<p>[s</p>	<p>Move to previous misspelled word before the cursor</p>	
<p>z=</p>	<p>Suggest spellings for the word under/after the cursor</p>	
<p>zg</p>	<p>Add word to spell list</p>	
<p>zw</p>	<p>Mark word as bad/misspelling</p>	
<p>zu / C-X (Insert Mode)</p>	<p>Suggest words for bad word under cursor from spellfile</p>	
Spell checking		
<p>Do these in visual or normal mode.</p>	<p>See :help spell</p>	
Also see		
<ul style="list-style-type: none"> Vim cheatsheet (vim.rotrr.com) Vim documentation (vimdoc.sourceforge.net) Interactive Vim tutorial (openvim.com) 		

▶ 15 Comments for this cheatsheet. Write yours!

The screenshot shows the Devhints.io homepage. On the left, there's a sidebar with a purple background. It features a circular icon with a white arrow pointing left. Below it, the text "Over 357 curated cheatsheets, by developers for developers." and a "Devhints home" button. The main content area has two sections: "Other Vim cheatsheets" and "Top cheatsheets".

Other Vim cheatsheets		Top cheatsheets	
Vimdiff cheatsheet	Vim scripting cheatsheet	Elixir cheatsheet	ES2015+ cheatsheet
Tabular cheatsheet	Projectionist cheatsheet	React.js cheatsheet	Vimdiff cheatsheet
Vim digraphs cheatsheet	Vim Easyalign cheatsheet	Vim scripting cheatsheet	Vue.js cheatsheet

Ref

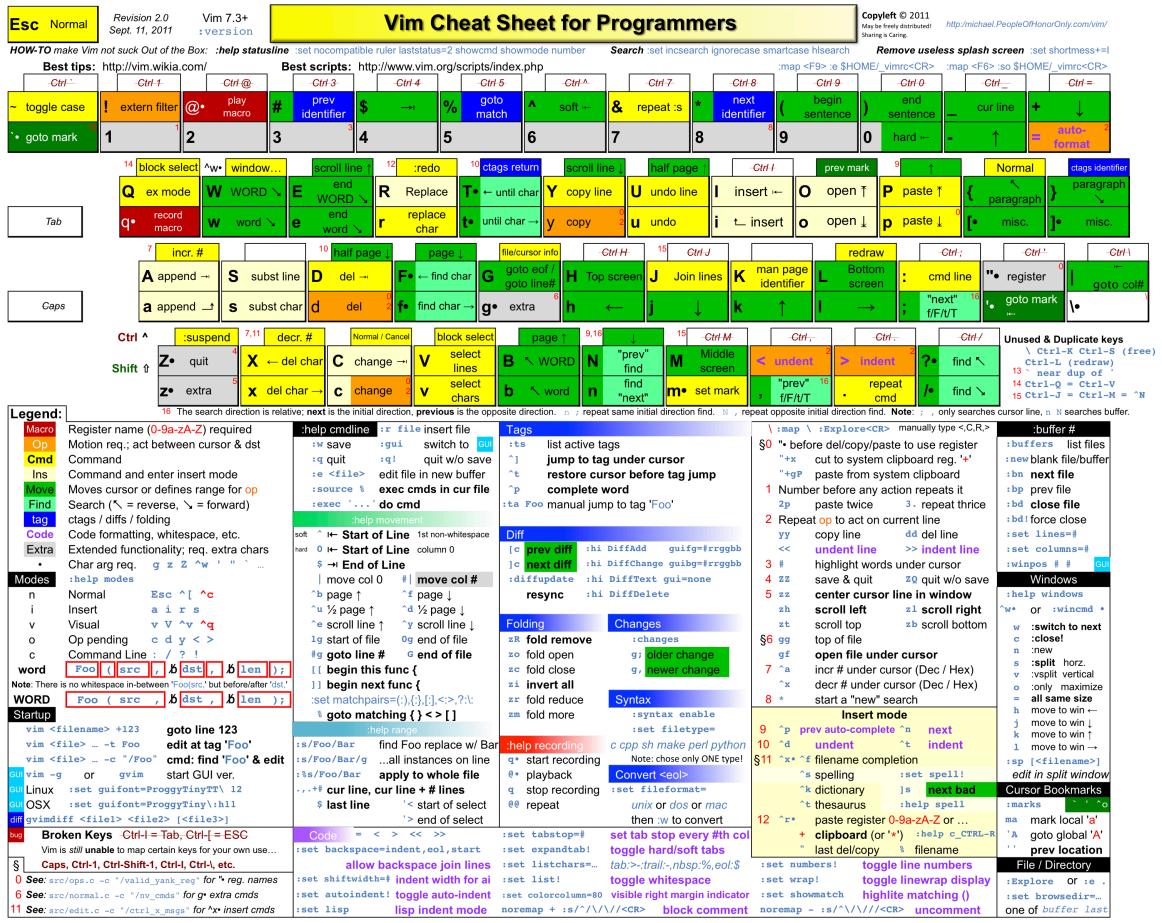
<https://devhints.io/vim>

More Describe Chinese

<https://catbro666.github.io/posts/d6ca5270/>

本文不会介绍基本的VIM使用方法，如果你对VIM的几种模式、基本的编辑、移动和选取等还不了解，推荐你先看一下这篇简明 VIM 练级攻略，写得还是不错的，基本上最常用的命令都有了。如果你不是vim重度用户的话，知道这些命令已经足够。只能说vim的水太深，根据自身情况在编辑效率和学习成本之间做个平衡吧。

这里有一个vim cheat sheet图，可以作为快速查阅用。要了解具体的命令推荐查看vim的help文档，内容写得非常详细。



Vim编辑命令一般格式

vim中编辑命令的结构

```
<number> <operator> <number> <text object or motion>
```

motion或对象前可以包含一个数字，如果operator前也带了一个数字，两者是相乘的关系。

例如 `2d3w`，删除6个单词。

操作符

操作符也是可选的，如果没有则变成了移动命令，而非编辑命令。

vim默认的operator有

命令	作用
c	change
d	delete
y	yank into register
~	swap case (only if 'tilde' is set)
g~	swap case
gu	make lowercase
gU	make uppercase
!	filter through an external program
=	filter through 'equalprg' or C-indenting if empty
gq	text formatting
g?	ROT13 encoding
>	shift right
<	shift left
zf	define a fold
g@	call function set with the 'operatorfunc' option

操作要么影响整行，要么影响起止位置之间的字符。移动有inclusive和exclusive之分。

如果操作pending了（即操作符已经输入了， motion还没有），有一些特殊映射可以使用 (:omap) 。

motion也可以使用命令，例如 d:call FindEnd()，如果命令多于一行不能重复。

可以在操作符后输入 v、V 或 CTRL-V，强制将motion转换为逐字符、逐行、逐块。其中对于v，如果本身就是逐字符的，则转换inclusive/exclusive。

扩展的操作符：

快捷键	命令	作用
nnoremap Y	:CopyText	拷贝到指定字符
nnoremap D	:DeleteText	删除到指定字符
nnoremap C	:ChangeText	改写到某个字符

成对符号的操作：

快捷键	例子	作用
cs""	"Hello world!" -> 'Hello world!'	"改成'
cs"""	'Hello world!' -> "Hello world!"	'改成"对"
cst"	"Hello world!" -> "Hello world!"	"对改成", t表示完整匹配""
ds"	"Hello world!" -> Hello world!	去掉"
dst	"Hello world!" -> Hello world!	t表示完整匹配", 而不是单个<"
ysiw]	Hello world! -> [Hello] world!	光标所在单词加括号, 右括号不加空格
ysiw{	Hello world! -> { Hello } world!	光标所在单词加括号, 左括号加空格
yss)	{ Hello } world! -> ({ Hello } world!)	整行加()对

可以跟vim-repeat配合, 使用 . 重复之前的操作。

左右移动

操作符	作用	inclusive/exclusive
h	左移	e
l	右移	e
0	移到行首	e
^	移到行首第一个非空字符	e
\$	移到行尾	i
g_	移到行尾第一个非空字符	i
g0	移到当前行屏幕最左字符	e
g^	移到屏幕行第一个非空字符	e
gm	类似g0，但是屏幕移动	e
g\$	屏幕行的最后一个非空字符	i
 	到当前行的屏幕中的第[count]列	e
f{char}	移动到右边第[count]个{char}字符出现处	i
F{char}	移动到左边第[count]个{char}字符出现处	e
t{char}	移动到右边第[count]个{char}字符前	i
T{char}	移动到左边第[count]个{char}字符后	e
;	重复最近一个f,t,F,T [count]次	
,	重复最近一个f,t,F,T 相反方向[count]次	

上下移动

操作符	作用	inclusive/exclusive
k	上移	i
j	下移	i
gk	上移显示行, 行折叠时或者与操作符使用时	e
gj	下移显示行	e
-	上移, 在第一个非空字符	
_	下移[count] -1行, 第一个非空字符	
G	到第[count]行, 默认最后一行, 第一个非空字符	
gg	到第[count]行, 默认第一行, 第一个非空字符	
:[range]	光标跳到[range]的最后一行, 相比G不改jumplist	
{count}%	跳到文件百分之{count}处, 第一个非空字符	

单词移动

操作符	作用	inclusive/exclusive
w	往前[count]个words	e
W	往前[count]个WORDs	e
e	往前到第[count]个words的最后, 空行不停	i
E	往前到第[count]个WORDs的最后, 空行不停	i
b	往回[count]个words	e
B	往回[count]个WORDs	e
ge	往回到第[count]个words的最后	i
gE	往回到第[count]个WORDs的最后	i
w	跳到指定单词处	

其中**word**由字母、数字、下划线的序列组成，或者其他非空字符的序列，以空白符分隔。可以用 `iskeyword` 选项修改。空行也当成一个word。

WORD由非空字符序列组成，以空白符分隔。空行也当成一个WORD。

特殊情况：

- `cw` 和 `cW` 当作 `ce` 和 `cE` 处理（如果光标在非空字符上的话）
- 当 `w` 跟操作符一起使用，当最后一个移过的单词是一行最后时，单词尾部成为操作文本的尾部，不

是下一行的开始。

文本对象移动

操作符	作用	inclusive/exclusive
(回移[count]个句子	e
)	前移[count]个句子	e
{	回移[count]个段落	e
}	前移[count]个段落	e
]]	前移[count]个小节或第1列中的下一个'{'。如果和操作符一起使用，则也停在第一列的'}'下	e
][前移[count]个小节或第1列中的下一个'{'	e
[[回移[count]个小节或第1列中的前一个'{'	e
[]	回移[count]个小节或第1列中的前一个'{'	e

句子：由 . 或 ! 或 ? 结尾，后面要么时行尾、要么是空格或tab。. 或 ! 或 ? 后面可以跟任意数目的'、]'、'''、''。

段落：由空行之后开始，或者由段落宏开始。小节的边界也是段落的边界。注意空行不是段落的边界。

文本的对象选取

这一系列只能在visual模式或者操作符之后才能用的命令。`a`命令选取n个对象，包括空白。`i`命令选取一个内部对象不带空白或者只选取空白，所以`i`命令肯定比`a`命令选得少。

`v3aw`, `d2iw`。

可以跟各种文本对象组合。如aw, iw, aW, iW, as, is句子, ap, ip段落, a], a[, i], i[是[]块, a), a(, ab, i), i(, ib是块, a>, a<, i>, i<是尖括号块, at, it是tag块, a}, a{, aB, i}, i{, iB是块, a", a', a ,i", i', i 是引用的字符串。

扩展文本对象

操作符	作用	inclusive/exclusive
if	函数文件对象, 内部代码块	
af	函数文件对象, 整个函数	
iF	函数参数对象, 整个函数	
aF	函数参数对象, 整个函数包含前后的空行	
i,	内部参数, 不包括后面的逗号和空格	
a,	参数, 包含逗号和空格	

当在操作符后面使用时

- 对于非块对象

a系列命令, 作用于对象及其后面的空白。如果对象后面没有空白或光标在对象之前的空白上, 对象前面的空白也包含进来。

i系列命令, 如果光标在对象上, 作用于对象; 如果光标在空白上, 作用于空白

- 对于块对象

作用于光标所在的块 (光标在块内或括号上)。a系列命令包括括号, i系列不包括括号。

当在Visual模式使用时

- 如果是刚进入Visual模式

选择一个对象, 就跟前面使用操作符时一样

- 如果不是刚进入visual模式

对于非块对象, visual区域扩展一个对象或下一个对象前的空白, 或者扩展两者。选取方法取决于光标运动方向。对于块对象, 往外扩展一层。

例子

下面是删除命令为例的一个例子

```

"dl"      delete character (alias: "x")           dl
"diw"    delete inner word                      diw
"daw"    delete a word                          daw
"diW"    delete inner WORD (see WORD)          diW
"daW"    delete a WORD (see WORD)              daW
"dgn"    delete the next search pattern match   dgn
"dd"     delete one line                        dd
"dis"    delete inner sentence                 dis
"das"    delete a sentence                     das
"bib"    delete inner '()' block                dib
"bab"    delete a '()' block                  dab
"rip"    delete inner paragraph               dip

```

```

"dap"    delete a paragraph           dap
"diB"    delete inner '{ }' block    diB
"daB"    delete a '{ }' block        daB

```

搜索替换

也是光标的跳转。在按回车之前可以按**ctrl+j/k**来上下选择。

快捷键	命令	作用
map /	(incsearch-forward)	往下搜索
map ?	(incsearch-backward)	往上搜索
map g/	(incsearch-stay)	按回车后光标不动

快捷键	命令	作用
nnoremap f	:LeaderfFile .	当前目录下文件搜索, tab选择
nnoremap F	:Ack!	文本正则匹配
rr		normal下替换掉光标所在单词, visual替换选中文本
nnoremap r	:ReplaceTo	全文替换单词

文件跳转

快捷键	命令	作用
nnoremap U	:GoToFunImpl	跳到函数实现
nnoremap a	:Switch	C++头文件和源文件切换
nnoremap u	:YcmCompleter GoToDeclaration	跳转的函数声明
nnoremap o	:YcmCompleter GoToInclude	跳转到include文件
Ctrl +]		ctags跳转到函数或变量定义处
Ctrl + t		ctags返回跳转前的地方

buffer切换

快捷键	命令	作用
nnoremap	:PreviousBuffer	切换到上一个buffer
nnoremap	:NextBuffer	切换到下一个buffer
nnoremap d	:CloseCurrentBuffer	关闭当前buffer
nnoremap D	:BufOnly	删除除当前buffer外的所有buffer

git

:Git 后加对应git命令，还有一些扩展命令，如 :Gread、:Ggrep 等。

git commit浏览器。

快捷键	命令	作用
nnoremap g	:GV	打开commit浏览器，可以加git log选项
nnoremap G	:GV!	只列出影响当前文件的commits
nnoremap gg	:GV?	当前文件revisions的位置列表

:GV 和 :GV? 可以用于visual模式，追踪选择行的变化。

在列表中界面

o 或 <cr>，单个commit，显示其内容。

o 或 <cr>，多个commits，显示所选范围的变化

o，打开一个新tab显示，而不是分隔窗口

.`，当前光标所在commit，开始一条命令` :Git [CURSOR] SHA

q 或 qq 关闭

自动生成与格式化

CPP辅助

快捷键	命令	作用
nnoremap y	:CopyCode	拷贝函数或变量
nnoremap p	:PasteCode	生成函数实现或变量定义
nnoremap fp	:FormatFunParam	格式化函数参数
nnoremap if	:FormatIf	格式化if-else
nnoremap t	:GenTryCatch	生成try-catch块

文本对齐。

快捷键	命令	作用
nnoremap =	:Tab /=	关于=号对齐
	:Tab /,/r1c1l0	逗号分隔，右对齐空一个，中对齐空一格左对齐不空

快速注释

快捷键	命令	作用
gcc		单行注释
gc		visual模式注释选择行
gcap		注释一整段
:7,17Commentary		注释所选行
gcgc		去掉所有相邻注释行的注释

新建文件时模版 `~/.vim/plugged/prepare-code/snippet`

vim与shell交互

- `:!command`

不退出vim，执行shell命令，例如 `:!ls -l`

- `:!!`

执行上一次的命令

- `:r !command 或 !!command`

将命令执行结果插入到当前行的下一行，例如 `:r date`

- `:start,end !command`

将指定行范围中的内容输入到shell命令进行处理，并用输出结果替换指定内容，例如 `:62,72 !sort`。

可以只指定单行, `:62 !tr "[a-z]" "[A-Z]"`, 将62行转成大写字母。

当前行可以用`.`表示, 例如`. !tr "[a-z]" "[A-Z]"`

- `:start,end w !command`

将指定行范围中的内容输入到shell命令进行处理, 但是只是显示结果, 不会改变当前文件的内容。例如, `:62,72 w !sort`

- `:shell`

不退出vim, 新开一个shell, exit之后回到vim。

- `:terminal` 或 `:vertical :term`

在新建的分割窗口中进入终端。可以用`Ctrl-W N`或`Ctrl-\ Ctrl-N`在终端buffer中进入Normal模式, 注意前者的N是大写的, 需要按住shift。Terminal-Normal模式下可以选择复制文本。点击`i`键可以返回到Terminal-Job模式。

其他命令

快捷键	命令	作用
<code>nnoremap t</code>	<code>:TagBarToggle .</code>	显式右边栏, 方法变量
<code>nnoremap n</code>	<code>:NERDTreeToggle .</code>	显式左边栏, 文件目录
<code>nnoremap ff</code>	<code>:YcmCompleter FixIt</code>	自动修复错误
<code>nmap</code>	<code>:YcmDiags</code>	诊断错误

快捷键映射

map系列命令语法

```
[<mode>][nore]map [<args>] {lhs} {rhs}
```

第一个可选字符表示模式

字符	模式
n	normal only
v	visual and select
o	operator-pending
x	visual only
s	select only
i	insert
c	command-line
I	insert, command-line, regexp-search

noremap 不递归映射

例如，下面的例子是正常的

```
noremap Y y
```

```
noremap y Y
```

常用的参数有

`<silent>` 表示静默映射，不会显示Vim在处理rhs过程中对界面产生的变化。

`<buffer>` 表示这个映射只是在当前的buffer中定义，而不是定义全局的映射

`<expr>` 表示{rhs}是一个 vim表达式，而不是按键序列，见下文

表达式的例子如下：

```
noremap <expr>0 col('.') == 1 ? '^': '0'
```

在normal或visual模式下，按0键可以实现让光标在首列和首个非空白字符之间的切换。

插件整理

[vimplus](#)是一个vim自动配置工具，基本上包含了下面所有的插件，可以傻瓜式一键安装使用。

ctags

跳转到标签定义处，需要事先生成索引文件

```
ctags -R .
```

快捷键	用法
Ctrl +]	跳转到函数或变量定义处
Ctrl + t	返回跳转前的地方

注意只有在索引所在目录下才能实现跳转

[chxuan/cpp-mode](#)

提供生成函数实现、函数声明/实现跳转、.h .cpp切换等功能

快捷键	命令	作用
nnoremap y	:CopyCode	拷贝函数或变量
nnoremap p	:PasteCode	生成函数实现或变量定义
nnoremap U	:GoToFunImpl	跳到函数实现
nnoremap a	:Switch	C++头文件和源文件切换
nnoremap fp	:FormatFunParam	格式化函数参数
nnoremap if	:FormatIf	格式化if-else
nnoremap t	:GenTryCatch	生成try-catch块

[chxuan/vim-edit](#)

快捷键	命令	作用
nnoremap Y	:CopyText	拷贝到指定字符
nnoremap D	:DeleteText	删除到指定字符
nnoremap C	:ChangeText	改写到某个字符
rr		normal下替换掉光标所在单词， visual替换选中文本
nnoremap r	:ReplaceTo	全文替换单词

[chxuan/prepare-code](#)

新建文件时自动生成代码，模板在 `~/.vim/plugged/prepare-code/snippet` 目录下

[chxuan/vim-buffer](#)

buffer切换

快捷键	命令	作用
nnoremap	:PreviousBuffer	切换到上一个buffer
nnoremap	:NextBuffer	切换到下一个buffer
nnoremap d	:CloseCurrentBuffer	关闭当前buffer
nnoremap D	:BufOnly	删除除当前buffer外的所有buffer

[preservim/tagbar](#)

右边栏，显式方法变量

快捷键	命令	作用
nnoremap t	:TagBarToggle .	显式右边栏

[ycm-core/YouCompleteMe](#)

快捷键	命令	作用
nnoremap u	:YcmCompleter GoToDeclaration	跳转的函数声明
nnoremap o	:YcmCompleter GoToInclude	跳转到include文件
nnoremap ff	:YcmCompleter FixIt	自动修复错误
nmap	:YcmDiags	诊断错误

[Yggdroot/LeaderF](#)

文件模糊搜索，按tab进行选择

快捷键	命令	作用
nnoremap f	:LeaderfFile .	当前目录下文件搜索， tab选择

[mileszs/ack.vim](#)

ack文本搜索

快捷键	命令	作用
nnoremap F	:Ack!	文本正则匹配

!表示不自动跳到第一个匹配。

可以考虑替换成[junegunn/fzf.vim]，使用rg/ag更快。

[easymotion/vim-easymotion](#)

normal模式光标快速跳转，visual模式选择到指定位置。

快捷键	命令	作用
map w	(easymotion-bd-w)	选取至指定位置
nmap w	(easymotion-overwin-w)	跳转到指定位置

[haya14busa/incsearch.vim](#)

文本搜索，其实跟前一个类似，也是光标的跳转。

在按回车之前可以按ctrl+j/k来上下选择。

快捷键	命令	作用
map /	(incsearch-forward)	往下搜索
map ?	(incsearch-backward)	往上搜索
map g/	(incsearch-stay)	按回车后光标不动

[jiangmiao/auto-pairs](#)

自动补全成对符号

[preservim/nerdtree](#)

显式左边栏，目录。

快捷键	命令	作用
nnoremap n	:NERDTreeToggle .	显式左边栏

[vim-nerdtree-syntax-highlight](#)

左边栏文件类型高亮

[godlygeek/tabular](#)

文本对齐。

快捷键	命令	作用
nnoremap =	:Tab /=	关于=号对齐
	:Tab /,/r1c1l0	逗号分隔, 右对齐空一个, 中对齐空一格左对齐不空

默认是左对齐, 下一个字段前空一格。可以通过一个字母加数字指定格式, 其中l/c/r分别表示左对齐/居中对齐/右对齐, 紧跟的数字表示下一个字段前插入的空格数。如果指定了多个格式会依次使用, 用完了再从第一个开始。注意这个分隔符本身也算一个字段。分隔符并不一定是单个字符, 其实可以是正则表达式。

[tptope/vim-fugitive](#)

集成Git, `:Git` 后加对应git命令, 还有一些扩展命令, 如 `:Gread`、`:Ggrep` 等。

[tptope/vim-surround](#)

快捷键	例子	作用
cs"	"Hello world!" -> 'Hello world!'	"改成'
cs"""	'Hello world!' -> "Hello world!"	'改成"对"
cst"	"Hello world!" -> "Hello world!"	"对改成", t表示完整匹配""
ds"	"Hello world!" -> Hello world!	去掉"
dst	"Hello world!" -> Hello world!	t表示完整匹配", 而不是单个<"
ysiw]	Hello world! -> [Hello] world!	光标所在单词加括号, 右括号不加空格
ysiw{	Hello world! -> { Hello } world!	光标所在单词加括号, 左括号加空格
yss)	{ Hello } world! -> ({ Hello } world!)	整行加()对

可以跟vim-repeat配合, 使用`.`重复之前的操作。

[tpope/vim-commentary](#)

快速注释

快捷键	命令	作用
gcc		单行注释
gc		visual模式注释选择行
gcap		注释一整段
:7,17Commentary		注释所选行
gcbc		去掉所有相邻注释行的注释

[tpope/vim-repeat](#)

. 操作支持map，支持tpope/vim-repeat等插件。

[tpope/vim-endwise](#)

if do等尾部补全

[octol/vim-cpp-enhanced-highlight](#)

cpp额外的语法高亮，增加了标准库/boost函数、容器、类型的高亮。

[vim-airline/vim-airline](#)

状态栏美化

[vim-airline/vim-airline-themes](#)

状态栏主题库

[ryanoasis/vim-devicons](#)

显示文件类型图标

[junegunn/vim-slash](#)

光标移动后清除搜索的高亮

[junegunn/gv.vim](#)

git commit浏览器。

快捷键	命令	作用
nnoremap g	:GV	打开commit浏览器，可以加git log选项
nnoremap G	:GV!	只列出影响当前文件的commits
nnoremap gg	:GV?	当前文件revisions的位置列表

:GV 和 :GV? 可以用于visual模式，追踪选择行的变化。

在列表中界面

- o 或 <cr>， 单个commit， 显示其内容。
- o 或 <cr>， 多个commits， 显示所选范围的变化
- o， 打开一个新tab显示， 而不是分隔窗口

.`， 当前光标所在commit， 开始一条命令` :Git [CURSOR] SHA

q 或 qq 关闭

[kana/vim-textobj-user](#)

自定义文本对象。例如下面定义了 ad / id 选取一个日期如 2013-03-16， at / it 选取一个时间如 22:04:21

```
call textobj#user#plugin('datetime', {
  'date': {
    'pattern': '\<\d\d\d\d-\d\d-\d\d\>',
    'select': ['ad', 'id'],
  },
  'time': {
    'pattern': '\<\d\d:\d\d:\d\d\>',
    'select': ['at', 'it'],
  },
})
```

[kana/vim-textobj-indent](#)

indented blocks of lines的文本对象， 不知道他指的啥

[kana/vim-textobj-syntax](#)

语法高亮项目的文本对象

[kana/vim-textobj-function](#)

函数的文本对象， af if aF iF

[sgur/vim-textobj-parameter](#)

函数参数文本对象， 默认是a, i,

[Shougo/echodoc.vim](#)

补全时在命令行显示函数签名。

[rhysd/clever-f.vim](#)

增强的f/F/t/T， 多次连续可以只按一个字符， 与第一大小写相同则同方向， 否则反方向。

[vim-scripts/indentpython.vim](#)

python缩进脚本

[rhysd/github-complete.vim](#)

github补全， 包括emoji、用户名、仓库名、issue号、链接URL补全。

[Drwalt](#)

vim画图工具，

快捷键	命令	作用
di	:DrawIt	开始画图模式
ds	:Dlstop	结束画图模式
a 选择模式		画箭头
b 选择模式		画矩形
e 选择模式		画椭圆
l 选择模式		画直线
c		画布增大
f		字符某个填充
s		追加空格至文本宽度 (默认78)
space		切换画图/擦除模式
r/R		替换模式
<>^v		插入箭头, 前面加粗箭头

[dense-analysis/ale](#)
