user.letter

command word user.letter	
air	*a*
bat	*b*
cap	*c*
drum	*d*
each	*e*
fine	*f*
gust	*g*
harp	*h*
sit	*i*
jury	*j*
crunch	*k*
look	*l*
made	*m*
near	*n*
odd	*o*
pit	*p*
quench	*q*
red	*r*
sun	*s*
trap	*t*
urge	*u*
vest	*v*
whale	*w*
plex	*x*
yank	*y*
zip	*z*

user.number_key

command word user.number_key	
zero	*0*
one	*1*
two	*2*
three	*3*
four	*4*
five	*5*
six	*6*
seven	*7*
eight	*8*
nine	*9*

user.modifier_key

command word user.modifier_key	
alt	*alt*
control	*ctrl*
shift	*shift*
super	*super*

user.special_key

command word user.special_key	
end	*end*
enter	*enter*
escape	*escape*
home	*home*
insert	*insert*
pagedown	*pagedown*
pageup	*pageup*
space	*space*
tab	*tab*
delete	*backspace*
forward delete	*delete*
page up	*pageup*
page down	*pagedown*
menu key	*menu*
print screen	*printscr*

user.symbol_key

command word user.symbol_key	
dot	.
quote	'
L square	*[*
left square	*[*
square	*[*
R square	*]*
right square	*]*
slash	*/*
backslash	**
minus	-
dash	-
equals	=
plus	+
tilde	~
bang	*!*
dollar	*$*
down score	* _ *
under score	* _ *
paren	*(*
L paren	*(*
left paren	*(*
R paren	*)*
right paren	*)*
brace	*{*
left brace	*{*
R brace	*}*
right brace	*}*
angle	<
left angle	<
less than	<
rangle	>
R angle	>
right angle	>
greater than	>
star	***
pound	#
hash	#
percent	*%*
caret	^
amper	*&*
pipe	|
dubquote	"
double quote	"
**** *	*
,	,
back tick	`
comma	,
period	.
semicolon	*;*
colon	*:*
forward slash	*/*
question mark	*?*
exclamation mark	*!*
exclamation point	*!*
dollar sign	*$*
asterisk	***
hash sign	#
number sign	#
percent sign	*%*

at sign	*@*
and sign	&
ampersand	&

user.arrow_key

command word user.arrow_key	
down	*down*
left	*left*
right	*right*
up	*up*

user.punctuation

command word user.punctuation	
**** *	*
,	,
back tick	`
comma	,
period	.
semicolon	;
colon	:
forward slash	/
question mark	?
exclamation mark	!
exclamation point	!
dollar sign	$
asterisk	***
hash sign	#
number sign	#
percent sign	%
at sign	*@*
and sign	&
ampersand	&

user.function_key

command word user.function_key	
F one	*f1*
F two	*f2*
F three	*f3*
F four	*f4*
F five	*f5*
F six	*f6*
F seven	*f7*
F eight	*f8*
F nine	*f9*
F ten	*f10*
F eleven	*f11*
F twelve	*f12*

formatters

command word user.formatters	**allcaps** EXAMPLE OF FORMATTING WITH ALLCAPS
alldown	example of formatting with alldown
camel	exampleOfFormattingWithCamel
dotted	example.of.formatting.with.dotted
dubstring	"example of formatting with dubstring"
dunder	__example__offormattingwithdunder
hammer	ExampleOfFormattingWithHammer
kebab	example-of-formatting-with-kebab
packed	example::of::formatting::with::packed
padded	example of formatting with padded
slasher	/example /of/formatting /with/slasher
smash	exampleofformattingwithsmash
snake	example_of_formatting_with_snake
string	'example of formatting with string'
title	Example of Formatting With Title

1password

password new `user.password_new()`
password dup `user.password_duplicate()`
password edit `user.password_edit()`
password delete `user.password_delete()`

1password global

password fill `user.password_fill()`
password show `user.password_show()`

amethyst

window next `key("alt-shift-j")`
window previous `key("alt-shift-k")`
window full `key("alt-shift-d")`
window tall `key("alt-shift-a")`
window middle `key("alt-shift-")`
window move main `key("alt-shift-enter")`
window grow `key("alt-shift-l")`
window shrink `key("alt-shift-h")`
window reevaluate `key("alt-shift-z")`

anaconda

anaconda `"conda "`
anaconda help `"conda --help\n"`
anaconda version `"conda --version\n"`
anaconda environment list `"conda env list\n"`
anaconda environment create `"conda env create -f "`
anaconda environment remove `"conda env remove -n "`
anaconda activate `"conda activate "`
anaconda clean `"conda clean "`
anaconda compare `"conda compare "`
anaconda config `"conda config "`
anaconda create `"conda create "`
anaconda info `"conda info "`
anaconda init `"conda init "`
anaconda install `"conda install "`
anaconda list `"conda list "`
anaconda package `"conda package "`
anaconda remove `"conda remove "`
anaconda uninstall `"conda uninstall "`
anaconda run `"conda run "`
anaconda search `"conda search "`
anaconda update `"conda update "`
anaconda upgrade `"conda upgrade "`
anaconda build `"conda build "`
anaconda convert `"conda convert "`
anaconda debug `"conda debug "`
anaconda develop `"conda develop "`
anaconda environment `"conda env "`
anaconda index `"conda index "`
anaconda inspect `"conda inspect "`
anaconda metapackage `"conda metapackage "`
anaconda render `"conda render "`
anaconda server `"conda server "`
anaconda skeleton `"conda skeleton "`
anaconda verify `"conda verify "`

discord

[channel] mentions last
`user.discord_mentions_last()`
[channel] mentions next
`user.discord_mentions_next()`
oldest unread
`user.discord_oldest_unread()`
toggle pins
`user.discord_toggle_pins()`
toggle inbox
`user.discord_toggle_inbox()`
toggle (members | member list)
`user.discord_toggle_members()`
pick emoji
`user.discord_emoji_picker()`
pick (jif | gif | gift)
`user.discord_gif_picker()`
mark inbox channel read
`user.discord_mark_inbox_read()`
[toggle] (mute | unmute)
`user.discord_mute()`
[toggle] (deafen | undeafen)
`user.discord_deafen()`
answer call
`user.discord_answer_call()`
decline call
`user.discord_decline_call()`

eclipse

please [<user.text>] `key(ctrl-3) insert(user.text or "")`
bar explore `key(alt-shift-w p)`
bar outline `key(alt-shift-q o)`
panel output `key(alt-shift-q) sleep(200ms) key(c)`
panel problems `key(alt-shift-q) sleep(200ms) key(x)`
panel errors `key(alt-shift-q) sleep(200ms) key(l)`
panel breakpoints `key(alt-shift-q) sleep(200ms) key(b)`
panel search `key(alt-shift-q) sleep(200ms) key(s)`
panel variables `key(alt-shift-q) sleep(200ms) key(v)`
show settings `key(alt-w p)`
show shortcuts `key(ctrl-shift-l)`
file hunt [<user.text>] `key(ctrl-shift-r) sleep(50ms) insert(text or "")`
file create `key(ctrl-n)`
file open folder `key(alt-shift-w x)`
file rename `key(alt-shift-w p enter f2)`
file reveal `key(alt-shift-w p enter)`
imports fix `key(ctrl-shift-o)`
refactor rename `key(alt-shift-r)`
refactor this `key(alt-shift-i)`
(go declaration | follow) `key(f3)`
go back `key(alt-left)`
go forward `key(alt-right)`
go marks `key(alt-end)`
toggle mark `key(ctrl-alt-b down enter)`
go next mark `key(alt-pagedown)`
go last mark `key(alt-pageup)`
break point `key(ctrl-shift-b)`
step over `key(f6)`
debug step into `key(f5)`
debug step out [of] `key(f7)`
debug continue `key(f8)`
copy line down `key(ctrl-alt-down)`
copy line up `key(ctrl-alt-up)`

file manager

title force
`user.file_manager_refresh_title()`
manager show
`user.file_manager_toggle_pickers()`
manager refresh
`user.file_manager_update_lists()`
go desk
`user.file_manager_open_user_direct`
go docks
`user.file_manager_open_user_direct`
go downloads
`user.file_manager_open_user_direct`
go pictures
`user.file_manager_open_user_direct`
go profile
`user.file_manager_open_user_direct`
go talon home
`user.file_manager_open_directory(p`
go talon user
`user.file_manager_open_directory(p`
go user
`user.file_manager_open_directory(p`
go back
`user.file_manager_go_back()`
go forward
`user.file_manager_go_forward()`
daddy
`user.file_manager_open_parent()`
follow <number> `directory = user.file_manager_get_directory_by - 1) user.file_manager_open_directory(d`
follow {user.file_manager_directories}
`user.file_manager_open_directory(f`
open <number> `file = user.file_manager_get_file_by_inde - 1) user.file_manager_open_file(file)`
folder <number> `directory = user.file_manager_get_directory_by - 1) user.file_manager_select_directory`
file <number> `file = user.file_manager_get_file_by_inde - 1) user.file_manager_select_file(file`
file {user.file_manager_files}
`user.file_manager_select_file(file`
folder new <user.text>
`user.file_manager_new_folder(text)`
properties show
`user.file_manager_show_properties(`
terminal here
`user.file_manager_terminal_here()`
folder next
`user.file_manager_next_folder_page`
folder last
`user.file_manager_previous_folder_`
file next
`user.file_manager_next_file_page()`
file last
`user.file_manager_previous_file_pa`

firefox

tab search `browser.focus_address() insert("% ")`
tab search <user.text>
`browser.focus_address() insert("% {text}") key(down)`

gdb

until <number> `"until {number}"`
force clear all break points `insert("d br\n") insert("y\n")`
break [on] clipboard `insert("break ") key(ctrl-shift-v) key(enter)`
list [source] `"list\n"`
info source `"info source\n"`
print `"p "`
print [variable] <user.text> `"p {text}"`
print hex `"p/x "`
print hex [variable] <user.text> `"p/x {text}"`
print string `"p/s "`
hex dump <number> bytes `"x/{number}bx "`
hex dump <number> (half|short) words `"x/{number}hx "`

hex dump <number> (d|long) words `"x/{number}dx "`
hex dump <number> quad words `"x/{number}gx "`
hex dump `"x/100gx "`
hex dump highlighted
```
insert("x/100gx ")
edit.copy()          edit.paste()
key(enter)
```
hex dump clipboard
```
insert("x/100gx ")          edit.paste()
key(enter)
```
source `"source \t\t"`
(list|show|info) display `"info display\n"`
display assembly line `"display /i $pc\n"`
display source `"display "`
enable display <number_small> `"enable display {number_small}\n"`
disable display <number_small> `"disable display {number_small}\n"`
undisplay `"undisplay\n"`
(list|show|info) local `"info local "`
(list|show|info) local typed `"info local -t "`
(list|show|info) variable `"info variable "`
(list|show|info) variable typed `"info variable -t "`
(list|show|info) locals `"info local\n"`
(list|show|info) variables `"info variables\n"`
info threads `"info threads\n"`
restart [program] `"r\n"`
continue `"c\n"`
back trace `"bt\n"`
debug quit `"quit\n"`
debug force quit `"quit\ny\n"`
(show|info) (inf|inferiors) `"info inferiors\n"`
inferior <number_small> `"inferior {number_small}\n"`
inferior `"inferior "`
resume main (inf|inferior)
```
insert("inferior 1\n")
insert("c\n")
```
resume [from] (inf|inferior) <number_small>
```
insert("inferior {number_small}\n")
insert("c\n")
```
set args `"set args "`
show follow (fork|forks) [mode] `"show follow-fork-mode\n"`
[set] follow (fork|forks) [mode] child `"set follow-fork-mode child\n"`
[set] follow (fork|forks) [mode] parent `"set follow-fork-mode parent\n"`
show detach on fork `"show detach-on-fork\n"`
set detach on fork `"set detach-on-fork on\n"`
unset detach on fork `"set detach-on-fork off\n"`
show list size `"show listsize\n"`
set list size <number_small> `"set listsize {number_small}\n"`
clear screen `"shell clear\n"`

generic browser

(address bar | go address | go url) `browser.focus_address()`
go home `browser.go_home()`
go forward `browser.go_forward()`
go (back | backward) `browser.go_back()`
go private `browser.open_private_window()`
bookmark show `browser.bookmarks()`
bookmark bar `browser.bookmarks_bar()`
bookmark it `browser.bookmark()`
bookmark tabs `browser.bookmark_tabs()`
(refresh | reload) it `browser.reload()`
(refresh | reload) it hard `browser.reload_hard()`
show downloads `browser.show_downloads()`
show extensions `browser.show_extensions()`
show history `browser.show_history()`
show cache `browser.show_clear_cache()`
dev tools `browser.toggle_dev_tools()`

generic debugger

step into `user.debugger_step_into()`
step over `user.debugger_step_over()`
step line `user.debugger_step_line()`
step over line `user.debugger_step_over_line()`
step out `user.debugger_step_out()`
continue `user.debugger_continue()`
debug start `user.debugger_start()`
debug stop `user.debugger_stop()`
debug exit `user.debugger_exit()`
debug detach `user.debugger_detach()`
debug restart `user.debugger_restart()`
show registers `user.debugger_show_registers()`
get register `user.debugger_get_register()`
set register `user.debugger_set_register()`
break now `user.debugger_break_now()`
break here `user.debugger_break_here()`
(list|show) (breaks|break points) `user.debugger_show_breakpoints()`
(set|add) (break|break point) `user.debugger_add_sw_breakpoint()`
(set|add) hardware (break|break point) `user.debugger_add_hw_breakpoint()`
clear all (breaks|break points) `user.debugger_clear_all_breakpoint`
clear (break|break point) `user.debugger_clear_breakpoint()`
clear (break|break point) <number_small> `user.debugger_clear_breakpoint_id(`
disable all (breaks|break points) `user.debugger_disable_all_breakpoi`
disable (break|break point) `user.debugger_disable_breakpoint()`
disable (break|break point) <number_small> `user.debugger_disable_breakpoint_i`
enable all (breaks|break points) `user.debugger_enable_all_breakpoir`
enable (break|break point) `user.debugger_enable_breakpoint()`
enable (break|break point) <number_small> `user.debugger_enable_breakpoint_ic`
(stack|back) trace `user.debugger_backtrace()`
disassemble `user.debugger_disassemble()`
disassemble here `user.debugger_disassemble_here()`
disassemble clipboard `user.debugger_disassemble_clipboar`
jump to address `user.debugger_goto_address()`
jump to clipboard `user.debugger_goto_clipboard()`
jump to highlighted `user.debugger_goto_highlighted()`
dump string `user.debugger_dump_ascii_string()`
dump unicode [string] `user.debugger_dump_unicode_string(`
dump pointers

`user.debugger_dump_pointers()`
list modules
`user.debugger_list_modules()`
inspect type
`user.debugger_inspect_type()`
clear line
`user.debugger_clear_line()`

generic terminal

lisa
`user.terminal_list_directories()`
lisa all
`user.terminal_list_all_directories`
katie [<user.text>]
`user.terminal_change_directory(tex or "")`
katie root
`user.terminal_change_directory_roo`
clear screen
`user.terminal_clear_screen()`
run last `user.terminal_run_last()`
kill all `user.terminal_kill_all()`

i3wm

port <number_small>
`user.system_command("i3-msg workspace {number_small}")`
port ten `user.system_command("i3-msg workspace 10")`
(port flip|flipper)
`user.system_command("i3-msg workspace back_and_forth")`
port right `user.system_command("i3-msg workspace next")`
port left `user.system_command("i3-msg workspace prev")`
(win|window) left
`user.system_command("i3-msg focus left")`
(win|window) right
`user.system_command("i3-msg focus right")`
(win|window) up
`user.system_command("i3-msg focus up")`
(win|window) down
`user.system_command("i3-msg focus down")`
((win|window) kill|murder)
`user.system_command("i3-msg kill")`
(win|window) stacking
`user.system_command("i3-msg layout stacking")`
(win|window) default
`user.system_command("i3-msg layout toggle split")`
(win|window) tabbed
`user.system_command("i3-msg layout tabbed")`
reload i three config
`user.system_command("i3-msg reload")`
restart i three
`user.system_command("i3-msg restart")`
(full screen|scuba)
`user.system_command("i3-msg fullscreen")`
toggle floating
`user.system_command("i3-msg floating toggle")`
focus floating
`user.system_command("i3-msg focus mode_toggle")`
center window
`user.system_command("i3-msg move position center")`
resize mode
`user.system_command('i3-msg mode "resize"')`
focus parent
`user.system_command("i3-msg focus parent")`
focus child
`user.system_command("i3-msg focus child")`
grow window
`user.system_command('i3-msg mode "resize"') key(right:10) key(down:10) key(escape) sleep(200ms) user.system_command("i3-msg move position center")`
shrink window
`user.system_command('i3-msg mode "resize"') key(left:10) key(up:10) key(escape) sleep(200ms) user.system_command("i3-msg move position center")`
horizontal (shell|terminal)
`user.system_command("i3-msg split h") user.i3wm_shell()`
vertical (shell|terminal)
`user.system_command("i3-msg split v") user.i3wm_shell()`
(shuffle|move (win|window) [to] port) <number_small>
`user.system_command("i3-msg move container to workspace {number_small}")`
(shuffle|move (win|window) [to] port ten) `user.system_command("i3-msg move container to workspace 10")`
(shuffle|move (win|window) [to] last port) `user.system_command("i3-msg move container to workspace back_and_forth")`
(shuffle|move (win|window) left)
`user.system_command("i3-msg move left")`
(shuffle|move (win|window) right)
`user.system_command("i3-msg move right")`
(shuffle|move (win|window) up)
`user.system_command("i3-msg move up")`
(shuffle|move (win|window) down)
`user.system_command("i3-msg move down")`
(win|window) horizontal
`user.system_command("i3-msg split h")`
(win|window) vertical
`user.system_command("i3-msg split v")`
make scratch
`user.system_command("i3-msg move scratchpad")`
[(show|hide)] scratch
`user.system_command("i3-msg scratchpad show")`
next scratch
`user.system_command("i3-msg scratchpad show") user.system_command("i3-msg scratchpad show")`
launch `user.i3wm_launch()`
launch <user.text>
`user.i3wm_launch() sleep(100ms) insert("{text}")`
lock screen `user.i3wm_launch()`
(launch shell|koopa)
`user.i3wm_shell()`
new scratch (shell|window)
`user.i3wm_shell() sleep(200ms) user.system_command("i3-msg move scratchpad") user.system_command("i3-msg scratchpad show")`

jetbrains

complete `user.idea("action CodeCompletion")`
perfect `user.idea("action CodeCompletion,action CodeCompletion")`

smart `user.idea("action SmartTypeCompletion")`
(done | finish) `user.idea("action EditorCompleteStatement")`
grab <number> `user.idea_grab(number)`
(action | please) `user.idea("action GotoAction")`
(action | please) <user.text> `user.idea("action GotoAction") insert(text)`
refactor `user.idea("action Refactorings.QuickListPopupAction"`
refactor <user.text> `user.idea("action Refactorings.QuickListPopupAction" insert(text)`
extract variable `user.idea("action IntroduceVariable")`
extract field `user.idea("action IntroduceField")`
extract constant `user.idea("action IntroduceConstant")`
extract parameter `user.idea("action IntroduceParameter")`
extract interface `user.idea("action ExtractInterface")`
extract method `user.idea("action ExtractMethod")`
refactor in line `user.idea("action Inline")`
refactor move `user.idea("action Move")`
refactor rename `user.idea("action RenameElement")`
rename file `user.idea("action RenameFile")`
fix (format | formatting) `user.idea("action ReformatCode")`
fix imports `user.idea("action OptimizeImports")`
(go declaration | follow) `user.idea("action GotoDeclaration")`
go implementation `user.idea("action GotoImplementation")`
go usage `user.idea("action FindUsages")`
go type `user.idea("action GotoTypeDeclaration")`
go test `user.idea("action GotoTest")`
go back `user.idea("action Back")`
go forward `user.idea("action Forward")`
find (everywhere | all) `user.idea("action SearchEverywhere")`
find (everywhere | all) <user.text> [over] `user.idea("action SearchEverywhere") sleep(500ms) insert(text)`
(search | find) class `user.idea("action GotoClass")`
(search | find) file `user.idea("action GotoFile")`
(search | find) path `user.idea("action FindInPath")`
(search | find) symbol `user.idea("action GotoSymbol")`
(search | find) symbol <user.text> `user.idea("action GotoSymbol") insert(text) key("enter")`
recent `user.idea("action RecentFiles")`
surround [this] with <user.text> [over] `idea("action SurroundWith") sleep(500ms) insert(text)`
insert generated <user.text> [over] `user.idea("action Generate") sleep(500ms) insert(text)`
insert template <user.text> [over] `idea("action InsertLiveTemplate") sleep(500ms) insert(text)`
create (template|snippet) `user.idea("action SaveAsTemplate")`
toggle recording `user.idea("action StartStopMacroRecording")`
change (recording | recordings) `user.idea("action EditMacros")`
play recording `user.idea("action PlaybackLastMacro")`
play recording <user.text> [over] `idea("action PlaySavedMacrosAction") insert(text) sleep(500ms) Key("enter")`
go mark `user.idea("action ShowBookmarks")`
toggle mark `user.idea("action ToggleBookmark")`
go next mark `user.idea("action GotoNextBookmark")`
go last mark `user.idea("action GotoPreviousBookmark")`
toggle mark <number> `user.idea("action ToggleBookmark{number}")`
go mark <number> `user.idea("action GotoBookmark{number}")`
expand deep `user.idea("action ExpandRegionRecursively")`
expand all `user.idea("action ExpandAllRegions")`
collapse deep `user.idea("action CollapseRegionRecursively")`
collapse all `user.idea("action CollapseAllRegions")`
go next (method | function) `user.idea("action MethodDown")`
go last (method | function) `user.idea("action MethodUp")`
clippings `user.idea("action PasteMultiple")`
copy path `user.idea("action CopyPaths")`
copy reference `user.idea("action CopyReference")`
copy pretty `user.idea("action CopyAsRichText")`
create sibling `user.idea("action NewElementSamePlace")`
create sibling <user.text> [over] `user.idea("action NewElementSamePlace") sleep(500ms) insert(text)`
create file `user.idea("action NewElement")`
create file <user.text> [over] `user.idea("action NewElement") sleep(500ms) insert(text)`
go task `user.idea("action tasks.goto")`
go browser task `user.idea("action tasks.open.in.browser")`
switch task `user.idea("action tasks.switch")`
clear task `user.idea("action tasks.close")`
configure servers `user.idea("action tasks.configure.servers")`
git pull `user.idea("action Vcs.UpdateProject")`
git commit `user.idea("action CheckinProject")`
git push `user.idea("action CheckinProject")`
git log `user.idea("action Vcs.ShowTabbedFileHistory")`
git browse `user.idea("action Github.Open.In.Browser")`
git (gets | gist) `user.idea("action Github.Create.Gist")`
git (pull request | request) `user.idea("action Github.Create.Pull.Request")`
git (view | show | list) (requests | request) `user.idea("action Github.View.Pull.Request")`
git (annotate | blame) `user.idea("action Annotate")`
git menu `user.idea("action Vcs.QuickListPopupAction")`
toggle project `user.idea("action`

ActivateProjectToolWindow")
toggle find user.idea("action ActivateFindToolWindow")
toggle run user.idea("action ActivateRunToolWindow")
toggle debug user.idea("action ActivateDebugToolWindow")
toggle events user.idea("action ActivateEventLogToolWindow")
toggle terminal user.idea("action ActivateTerminalToolWindow")
toggle git user.idea("action ActivateVersionControlToolWindow")
toggle structure user.idea("action ActivateStructureToolWindow")
toggle database user.idea("action ActivateDatabaseToolWindow")
toggle database changes user.idea("action ActivateDatabaseChangesToolWindow"
toggle make user.idea("action ActivatemakeToolWindow")
toggle to do user.idea("action ActivateTODOToolWindow")
toggle docker user.idea("action ActivateDockerToolWindow")
toggle favorites user.idea("action ActivateFavoritesToolWindow")
toggle last user.idea("action JumpToLastWindow")
toggle pinned user.idea("action TogglePinnedMode")
toggle docked user.idea("action ToggleDockMode")
toggle floating user.idea("action ToggleFloatingMode")
toggle windowed user.idea("action ToggleWindowedMode")
toggle split user.idea("action ToggleSideMode")
toggle tool buttons user.idea("action ViewToolButtons")
toggle toolbar user.idea("action ViewToolBar")
toggle status [bar] user.idea("action ViewStatusBar")
toggle navigation [bar] user.idea("action ViewNavigationBar")
toggle power save user.idea("action TogglePowerSave")
toggle whitespace user.idea("action EditorToggleShowWhitespaces")
toggle indents user.idea("action EditorToggleShowIndentLines")
toggle line numbers user.idea("action EditorToggleShowLineNumbers")
toggle (bread crumbs | breadcrumbs) user.idea("action EditorToggleShowBreadcrumbs")
toggle gutter icons user.idea("action EditorToggleShowGutterIcons")
toggle wrap user.idea("action EditorToggleUseSoftWraps")
toggle parameters user.idea("action ToggleInlineHintsAction")
toggle fullscreen user.idea("action ToggleFullScreen")
toggle distraction [free mode] user.idea("action ToggleDistractionFreeMode")
toggle presentation [mode] user.idea("action TogglePresentationMode")
toggle comment code.toggle_comment()
change scheme user.idea("action QuickChangeScheme")
(toggle | pop) (doc | documentation) user.idea("action QuickJavaDoc")
(pop deaf | toggle definition) user.idea("action QuickImplementations")
pop type user.idea("action ExpressionTypeInfo")
pop parameters user.idea("action ParameterInfo")
go breakpoints user.idea("action ViewBreakpoints")
toggle [line] breakpoint user.idea("action ToggleLineBreakpoint")
toggle method breakpoint user.idea("action ToggleMethodBreakpoint")
run menu user.idea("action ChooseRunConfiguration")
run test user.idea("action RunClass")
run test again user.idea("action Rerun")
debug test user.idea("action DebugClass")
step over user.idea("action StepOver")
step into user.idea("action StepInto")
step smart user.idea("action SmartStepInto")
step to line user.idea("action RunToCursor")
continue user.idea("action Resume")
(grow | shrink) window right user.idea("action ResizeToolWindowRight")
(grow | shrink) window left user.idea("action ResizeToolWindowLeft")
(grow | shrink) window up user.idea("action ResizeToolWindowUp")
(grow | shrink) window down user.idea("action ResizeToolWindowDown")
go next (error | air) user.idea("action GotoNextError")
go last (error | air) user.idea("action GotoPreviousError")
fix next (error | air) user.idea("action GotoNextError")
user.idea("action ShowIntentionActions")
fix last (error | air) user.idea("action GotoPreviousError")
user.idea("action ShowIntentionActions")
select less user.idea("action EditorUnSelectWord")
select (more|this) user.idea("action EditorSelectWord")
expand <number> until <number> user.select_range(number_1, number_2)
user.idea("action ExpandRegion")
collapse <number> until <number> user.select_range(number_1, number_2)
user.idea("action CollapseRegion")
paste <number> until <number> user.select_range(number_1, number_2)
user.idea("action EditorPaste")
refactor <number> until <number> user.select_range(number_1, number_2)
user.idea("action Refactorings.QuickListPopupAction"
clone <number> user.line_clone(number)
clear last <user.text> [over] user.idea("find prev {text}, action EditorBackSpace")
clear next <user.text> [over] user.idea("find next {text}, action EditorBackSpace")
comment last <user.text> [over] user.idea("find prev {text}, action CommentByLineComment")
comment next <user.text> [over] user.idea("find next {text},

action CommentByLineComment")
go last <user.text> [over] user.idea("find prev {text}, action EditorRight")
go next <user.text> [over] user.idea("find next {text}, action EditorRight")
paste last <user.text> [over] user.idea("find prev {text}, action EditorRight, action EditorPaste")
paste next <user.text> [over] user.idea("find next {text}, action EditorRight, action EditorPaste")
refactor last <user.text> [over] user.idea("find prev {text}, action Refactorings.QuickListPopupAction"
refactor next <user.text> [over] user.idea("find next {text}, action Refactorings.QuickListPopupAction"
replace last <user.text> [over] user.idea("find prev {text}, action EditorPaste")
replace next <user.text> [over] user.idea("find next {text}, action EditorPaste")
select last <user.text> [over] user.idea("find prev {text}")
select next <user.text> [over] user.idea("find next {text}")
select camel left user.extend_camel_left()
select camel right user.extend_camel_right()
go camel left user.camel_left()
go camel right user.camel_right()
blacken user.idea("action BLACKReformatCode")

kubectl

cube [control] "kubectl "
cube create "kubectl create "
cube expose "kubectl expose "
cube run "kubectl run "
cube set "kubectl set "
cube run container "kubectl run-container "
cube explain "kubectl explain "
cube get "kubectl get "
cube edit "kubectl edit "
cube delete "kubectl delete "
cube rollout "kubectl rollout "
cube rolling-update "kubectl rolling-update "
cube scale "kubectl scale "
cube auto scale "kubectl autoscale "
cube certificate "kubectl certificate "
cube top "kubectl top "
cube drain "kubectl drain "
cube taint "kubectl taint "
cube (cord | cordon) "kubectl cordon "
cube (uncord | uncordon) "kubectl uncordon "
cube cluster (info | information) "kubectl cluster-info "
cube describe "kubectl describe "
cube logs "kubectl logs "
cube attach "kubectl attach "
cube exec "kubectl exec "
cube port forward "kubectl port-forward "
cube proxy "kubectl proxy "
cube copy "kubectl cp "
cube auth "kubectl auth "
cube diff "kubectl diff "
cube apply "kubectl apply "
cube patch "kubectl patch "
cube replace "kubectl replace "
cube wait "kubectl wait "
cube convert "kubectl convert "
cube customize "kubectl kustomize "
cube label "kubectl label "
cube annotate "kubectl annotate "
cube completion "kubectl completion "
cube (interface | API) "kubectl api "
cube interface resources "kubectl api-resources "
cube interface versions "kubectl api-versions "
cube config "kubectl config "
cube help "kubectl help "
cube plugin "kubectl plugin "
cube version "kubectl version "
cube {user.kubectl_action} [{user.kubectl_object}] insert("kubectl {kubectl_action} ") insert(kubectl_object or "")
cube detach key("ctrl-p") key("ctrl-q")
cube shell insert("kubectl exec -it -- /bin/bash") key("left:13")

linux dunst

show notifications key(ctrl-)`
dismiss [notifications] user.system_command('dunstctl close')
dismiss all [notifications] user.system_command('dunstctl close-all')

linux keepassx

open database key(ctrl-o)
save database key(ctrl-s)
close database key(ctrl-w)
lock database key(ctrl-l)
quit key(ctrl-q)
[add] new entry key(ctrl-n)
clone entry key(ctrl-k)
(view|edit) entry key(ctrl-e)
delete entry key(ctrl-d)
copy user [name] key(ctrl-b)
copy password key(ctrl-c)
open (earl|url|link) key(ctrl-u)
copy (earl|url|link) key(ctrl-alt-u)
find key(ctrl-f)
find <user.text> key(ctrl-f) insert("{text}")

linux signal

show shortcuts key("ctrl-/")
(next|nav|navigate) [by] (sec|section) key("ctrl-t")
(prev|previous) (chat|conversation) key("alt-down")
next (chat|conversation) key("alt-up")
(prev|previous) unread key("alt-shift-down")
next unread key("alt-shift-up")
[open] (pref|preferences) key("ctrl-,")
open conversation menu key("ctrl-shift-l")
search key("ctrl-f")
search chat key("ctrl-shift-f")
focus (chat|composer) key("ctrl-shift-t")
open media key("ctrl-shift-m")
open emoji key("ctrl-shift-j")
open sticker key("ctrl-shift-s")
record [voice] message key("ctrl-shift-v")
archive chat key("ctrl-shift-a")
unarchive chat key("ctrl-shift-u")
(first|top) message key("ctrl-up")
(last|bottom) message key("ctrl-down")

close chat `key("ctrl-shift-c")`
send it `key("enter")`
message details `key("ctrl-d")`
reply [message] `key("ctrl-shift-r")`
react [message] `key("ctrl-shift-e")`
save attachment `key("ctrl-s")`
delete [message] `key("ctrl-shift-d")`
send message `key("ctrl-enter")`
expand chat `key("ctrl-shift-x")`
attach [file] `key("ctrl-u")`
remove [link] preview `key("ctrl-p")`
remove [link] attachment `key("ctrl-shift-p")`

linux taskwarrior

task version `"task --version\n"`
task commands `"task commands\n"`
task help `"task help\n"`
task list `"task list\n"`
task list orphans `"task project:list\n"`
task list untagged `"task tags.none:list\n"`
task list <user.text> `"task list {text}\n"`
task list project `"task list project:"`
task list project <user.text> `"task list project:{text}\n"`
task add `"task add "`
task add <user.text> `"task add {text}\n"`
task undo `"task undo\n"`
(tasks|task next) `"task next\n"`
task <number> edit `"task {number} edit"`
task <number> done `"task {number} done"`
task <number> delete `"task {number} delete"`

linux terminal

run last `key(up) key(enter)`
rerun <user.text> `key(ctrl-r) insert(text)`
rerun search `key(ctrl-r)`
kill all `key(ctrl-c)`
go tab <number> `key("alt-{number}")`

linux termite

shell yank `key("y")`
shell select `key("ctrl-shift-space")`
shell insert `key("escape")`
visual line `key("v")`
visual line mode `key("V")`

linux tmux

mux `"tmux "`
mux new session `insert('tmux new ')`
mux sessions `key(ctrl-b) key(s)`
mux name session `key(ctrl-b) key($)`
mux kill session `insert('tmux kill-session -t ')`
mux new window `key(ctrl-b) key(c)`
mux window <number> `key(ctrl-b ) key('{number}')`
mux previous window `key(ctrl-b) key(p)`
mux next window `key(ctrl-b) key(n)`
mux rename window `key(ctrl-b) key(,)`
mux close window `key(ctrl-b) key(&)`
mux split horizontal `key(ctrl-b) key(%)`
mux split vertical `key(ctrl-b) key(")`
mux next pane `key(ctrl-b) key(o)`
mux move <user.arrow_key> `key(ctrl-b) key(arrow_key)`
mux close pane `key(ctrl-b) key(x)`
mux pane numbers `key(ctrl-b) key(q)`

mac datagrip

run `key(cmd-enter)`
run it `key(cmd-enter) sleep(50ms) key(enter)`
back `key(alt-left)`
fwack `key(alt-right)`
erase `key(alt-backspace)`
move up `key(cmd-shift-up)`
move down `key(cmd-shift-down)`

mac desktops

desk <number> `user.desktop(number)`
window move desk <number> `user.window_move_desktop(number)`
window move desk left `user.window_move_desktop_left()`
window move desk right `user.window_move_desktop_right()`

mac finder

preferences `key(cmd-,)`
options `key(cmd-j)`
search `key(cmd-alt-f)`
sort by none `key(ctrl-alt-cmd-0)`
sort by name `key(ctrl-alt-cmd-1)`
sort by kind `key(ctrl-alt-cmd-2)`
sort by date opened `key(ctrl-alt-cmd-3)`
sort by date added `key(ctrl-alt-cmd-4)`
sort by date modified `key(ctrl-alt-cmd-5)`
sort by size `key(ctrl-alt-cmd-6)`
icon view `key(cmd-1)`
column view `key(cmd-3)`
list view `key(cmd-2)`
gallery view `key(cmd-4)`
copy path `key(alt-cmd-c)`
trash it `key(cmd-backspace)`
hide [finder] `key(cmd-h)`
hide others `app.window_hide_others()`

mac rstudio

run that `key("cmd-enter")`
run document `key("cmd-alt-r")`
run from top `key("cmd-alt-b")`
run to end `key("cmd-alt-e")`
run (function|funk) `key("cmd-alt-f")`
run section `key("cmd-alt-t")`
run previous chunks `key("cmd-alt-p")`
run chunk `key("cmd-alt-c")`
run next chunk `key("cmd-alt-n")`
run all `key("cmd-shift-s")`
run knitter `key("cmd-shift-k")`
run profiler `key("cmd-shift-alt-p")`
jump back `key("cmd-f9")`
jump forward `key("cmd-f10")`
close all tabs `key("cmd-shift-w")`
indent lines `key("cmd-i")`
toggle comment `key("cmd-shift-c")`
reformat comment `key("cmd-shift-/")`

reformat R code key("cmd-shift-a")
line up key("alt-up")
line down key("alt-down")
duplicate line up key("cmd-alt-up")
duplicate line [down] key("cmd-alt-down")
select to paren key("ctrl-shift-e")
select to matching paren key("ctrl-shift-alt-e")
jump to matching key("ctrl-p")
expand selection key("shift-alt-cmd-up")
reduce selection key("shift-alt-cmd-down")
add cursor up key("ctrl-alt-up")
add cursor down key("ctrl-alt-down")
move active cursor up key("ctrl-alt-shift-up")
move active cursor down key("ctrl-alt-shift-down")
delete line key("cmd-d")
delete word left key("alt-backspace")
delete word right key("alt-delete")
assign that key("alt--")
pipe that key("cmd-shift-m")
insert knitter chunk key("cmd-alt-i")
fold that key("cmd-alt-l")
unfold that key("cmd-shift-alt-l")
fold all key("cmd-alt-o")
unfold all key("cmd-shift-alt-o")
find and replace key("cmd-f")
find next key("cmd-g")
find previous key("cmd-shift-g")
find with selection key("cmd-e")
find in files key("cmd-shift-f")
run replace key("cmd-shift-j")
run spell check key("f7")
go to source key("ctrl-1")
go to console key("ctrl-2")
go to help key("ctrl-3")
go to history key("ctrl-4")
go to files key("ctrl-5")
go to (plots|plot) key("ctrl-6")
go to packages key("ctrl-7")
go to environment key("ctrl-8")
go to git key("ctrl-9")
go to build key("ctrl-0")
go to terminal key("alt-shift-t")
go to omni key("ctrl-.")
go to line key("cmd-shift-alt-g")
go to section key("cmd-shift-alt-j")
go to tab key("ctrl-shift-.")
go to previous tab key("ctrl-f11")
go to next tab key("ctrl-f12")
go to first tab key("ctrl-shift-f11")
go to last tab key("ctrl-shift-f12")
zoom source key("ctrl-shift-1")
(zoom|show) all key("ctrl-shift-0")
help that key("f1")
define that key("f2")
previous plot key("cmd-alt-f11")
next plot key("cmd-alt-f12")
restart R session key("cmd-shift-f10")
dev tools build key("cmd-shift-b")
dev tools load all key("cmd-shift-l")
dev tools test key("cmd-shift-t")
dev tools check key("cmd-shift-e")
dev tools document key("cmd-shift-d")
toggle breakpoint key("shift-f9")
debug next key("f10")
debug step into (function|funk) key("shift-f4")
debug finish (function|funk) key("shift-f6")
debug continue key("shift-f5")
debug stop key("shift-f8")
run git diff key("ctrl-alt-d")
run git commit key("ctrl-alt-m")

mac terminal

rerun search key(ctrl-r)
suspend key(ctrl-z)
resume insert("fg") key(enter)

mac slack

workspace <number> key("cmd-{number}")
(slack | lack) [channel] info key(cmd-shift-i)
(move | next) focus key(ctrl-)`
[next] (section | zone) key(f6)
(previous | last) (section | zone) key(shift-f6)
(slack | lack) [direct] messages key(cmd-shift-k)
(slack | lack) threads key(cmd-shift-t)
(slack | lack) (history [next] | back | backward) key(cmd-[)
(slack | lack) forward key(cmd-])
[next] (element | bit) key(tab)
(previous | last) (element | bit) key(shift-tab)
(slack | lack) (my stuff | activity) key(cmd-shift-m)
(slack | lack) directory key(cmd-shift-e)
(slack | lack) (starred [items] | stars) key(cmd-shift-s)
(slack | lack) unread [messages] key(cmd-j)
(go | undo | toggle) full key(ctrl-cmd-f)
grab left key(shift-up)
grab right key(shift-down)
add line key(shift-enter)
(slack | lack) (slap | slaw | slapper) key(cmd-right shift-enter)
(slack | lack) (react | reaction) key(cmd-shift-\\)
(insert command | commandify) key(cmd-shift-c)
insert code insert("``````") key(left left left) key(shift-enter) key(shift-enter) key(up)
(slack | lack) (bull | bullet | bulleted) [list] key(cmd-shift-8)
(slack | lack) (number | numbered) [list] key(cmd-shift-7)
(slack | lack) (quotes | quotation) key(cmd-shift->)
bold key(cmd-b)
(italic | italicize) key(cmd-i)
(strike | strikethrough) key(cmd-shift-x)
(clear | scrap | scratch) key(cmd-a backspace)
(slack | lack) snippet key(cmd-shift-enter)
([toggle] mute | unmute) key(m)
(slack | lack) ([toggle] video) key(v)
(slack | lack) invite key(a)
(slack | lack) shortcuts key(cmd-/)
emote <user.text> "{text}"
toggle left sidebar key(cmd-shift-d)
toggle right sidebar key(cmd-.)

win slack

workspace <number> key("ctrl-{number}")
(slack | lack) [channel] info key(ctrl-shift-i)
(move | next) focus key(ctrl-)`
[next] (section | zone) key(f6)
(previous | last) (section | zone) key(shift-f6)
(slack | lack) [direct] messages key(ctrl-shift-k)
(slack | lack) threads key(ctrl-shift-t)
(slack | lack) (history [next] | back | backward) key(alt-left)
(slack | lack) forward key(alt-right)

[next] (element | bit) key(tab)
(previous | last) (element | bit) key(shift-tab)
(slack | lack) (my stuff | activity) key(ctrl-shift-m)
(slack | lack) directory key(ctrl-shift-e)
(slack | lack) (starred [items] | stars) key(ctrl-shift-s)
(slack | lack) unread [messages] key(ctrl-j)
grab left key(shift-up)
grab right key(shift-down)
add line key(shift-enter)
(slack | lack) (react | reaction) key(ctrl-shift-\\)
(insert command | commandify) key(ctrl-shift-c)
insert code insert("``````") key(left left left) key(shift-enter) key(shift-enter) key(up)
(slack | lack) (bull | bullet | bulleted) [list] key(ctrl-shift-8)
(slack | lack) (number | numbered) [list] key(ctrl-shift-7)
(slack | lack) (quotes | quotation) key(ctrl-shift-9)
bold key(ctrl-b)
(italic | italicize) key(ctrl-i)
(strike | strikethrough) key(ctrl-shift-x)
(clear | scrap | scratch) key(ctrl-a backspace)
(slack | lack) snippet key(ctrl-shift-enter)
([toggle] mute | unmute) key(m)
(slack | lack) ([toggle] video) key(v)
(slack | lack) invite key(a)
(slack | lack) shortcuts key(ctrl-/)
emote <user.text> "{text}"
toggle left sidebar key(ctrl-shift-d)
toggle right sidebar key(ctrl-.)

teams

show shortcuts key(ctrl-.)
go [to] search key(ctrl-e)
show commands key(ctrl-/)
open filter key(ctrl-shift-f)
go to key(ctrl-g)
open (apps|applications) key(ctrl-)`
[start] new chat key(ctrl-n)
open settings key(ctrl-,)
open help key(f1)
close key(escape)
open activity key(ctrl-1)
open chat key(ctrl-2)
open teams key(ctrl-3)
open calendar key(ctrl-4)
open planner key(ctrl-5)
open calls key(ctrl-6)
open files key(ctrl-7)
go [to] (prev|previous) [list] item key(alt-up)
go [to] next [list] item key(alt-down)
move [selected] team up key(ctrl-shift-up)
move [selected] team down key(ctrl-shift-down)
go [to] (prev|previous) section key(ctrl-shift-f6)
go [to] next section key(ctrl-f6)
go [to] compose [box] key(c)
[expand] compose [box] key(ctrl-shift-x)
send key(ctrl-enter)
attach file key(ctrl-o)
[start] new line key(shift-enter)
reply [to] [thread] key(r)
accept video call key(ctrl-shift-a)
accept audio call key(ctrl-shift-s)
decline call key(ctrl-shift-d)
start audio call key(ctrl-shift-c)
start video call key(ctrl-shift-u)
toggle mute key(ctrl-shift-m)
starch screen share session key(ctrl-shift-e)
toggle video key(ctrl-shift-o)
go [to] sharing toolbar key(ctrl-shift-space)
decline screen share key(ctrl-shift-d)
accept screen share key(ctrl-shift-a)
schedule [a] meeting key(alt-shift-n)
go to current time key(alt-.)
go to (prev|previous) (day|week) key(ctrl-alt-left)
go to next (day|week) key(ctrl-alt-right)
view day key(ctrl-alt-1)
view work week key(ctrl-alt-2)
view week key(ctrl-alt-3)
(safe|send) meeting request key(ctrl-s)
join [from] meeting [details] key(alt-shift-j)
go to suggested time key(alt-shift-s)

visual studio

panel solution key(ctrl-alt-l)
panel properties key(f4)
panel output key(ctrl-alt-o)
panel class key(ctrl-shift-c)
panel errors key(ctrl-\ ctrl-e)
panel design key(shift-f7)
panel marks key(ctrl-k ctrl-w)
panel breakpoints key(ctrl-alt-b)
show settings key(alt-t o)
fullscreen switch key(shift-alt-enter)
wrap switch key(ctrl-e ctrl-w)
file hunt [<user.text>] key(ctrl-shift-t) insert(text or "")
file create key(ctrl-n)
file rename key(ctrl-[s f2)
file reveal key(ctrl-[s)
hint show key(ctrl-shift-space)
definition show key(f12)
definition peek key(alt-f12)
references find key(shift-f12)
format that key(ctrl-k ctrl-d)
format selection key(ctrl-k ctrl-f)
imports fix key(ctrl-r ctrl-g)
refactor field key(ctrl-r ctrl-e)
refactor interface key(ctrl-r ctrl-i)
refactor method key(ctrl-r ctrl-m)
refactor reorder parameters key(ctrl-r ctrl-o)
refactor remove parameters key(ctrl-r ctrl-v)
refactor that key(ctrl-r ctrl-r)
(go declaration | follow) key(ctrl-f12)
go back key(ctrl--)
go forward key(ctrl-shift--)
go implementation key(f12)
go recent [<user.text>] key(ctrl-1 ctrl-r) sleep(100ms) insert(text or "")
go type [<user.text>] key(ctrl-1 ctrl-t) sleep(100ms) insert(text or "")
go member [<user.text>] key(alt-\) sleep(100ms) insert(text or "")
go usage key(shift-f12)
go marks key(ctrl-k ctrl-w)
toggle mark key(ctrl-k ctrl-k)
go next mark key(ctrl-k ctrl-n)
go last mark key(ctrl-k ctrl-p)
fold toggle key(ctrl-m ctrl-m)
fold toggle all key(ctrl-m ctrl-l)
fold definitions key(ctrl-m ctrl-o)
break point key(f9)
step over key(f10)
debug step into key(f11)
debug step out [of] key(f10)
debug start key(f5)

debug stopper `key(shift-f5)`
debug continue `key(f5)`

vscode

window reload
`user.vscode("workbench.action.relo`
window close
`user.vscode("workbench.action.clos`
please [<user.text>]
`user.vscode("workbench.action.show`
`insert(user.text or "")`
bar explore
`user.vscode("workbench.view.explor`
bar extensions
`user.vscode("workbench.view.extens`
bar outline
`user.vscode("outline.focus")`
bar run
`user.vscode("workbench.view.debug"`
bar search
`user.vscode("workbench.view.search`
bar source
`user.vscode("workbench.view.scm")`
bar switch
`user.vscode("workbench.action.togg`
symbol hunt [<user.text>]
`user.vscode("workbench.action.gotc`
`sleep(50ms) insert(text`
`or "")`
panel control
`user.vscode("workbench.panel.repl.`
panel output
`user.vscode("workbench.panel.outpu`
panel problems
`user.vscode("workbench.panel.marke`
panel switch
`user.vscode("workbench.action.togg`
panel terminal
`user.vscode("workbench.panel.termi`
focus editor
`user.vscode("workbench.action.focu`
show settings
`user.vscode("workbench.action.oper`
show shortcuts
`user.vscode("workbench.action.oper`
show snippets
`user.vscode("workbench.action.oper`
centered switch
`user.vscode("workbench.action.togg`
fullscreen switch
`user.vscode("workbench.action.togg`
theme switch
`user.vscode("workbench.action.sele`
wrap switch
`user.vscode("editor.action.toggleW`
zen switch
`user.vscode("workbench.action.togg`
file hunt [<user.text>]
`user.vscode("workbench.action.quic`
`sleep(50ms) insert(text`
`or "")`
file copy path
`user.vscode("copyFilePath")`
file create sibling
`user.vscode_and_wait("explorer.new`
file create
`user.vscode("workbench.action.file`
file rename
`user.vscode("fileutils.renameFile"`
`sleep(150ms)`
file move
`user.vscode("fileutils.moveFile")`
`sleep(150ms)`
file open folder
`user.vscode("revealFileInOS")`
file reveal
`user.vscode("workbench.files.actic`
save ugly
`user.vscode("workbench.action.file`
suggest show
`user.vscode("editor.action.trigger`
hint show
`user.vscode("editor.action.trigger`
definition show
`user.vscode("editor.action.revealI`
definition peek
`user.vscode("editor.action.peekDef`
definition side
`user.vscode("editor.action.revealI`
references show
`user.vscode("editor.action.goToRef`
references find
`user.vscode("references-`
`view.find")`
format that
`user.vscode("editor.action.formatI`
format selection
`user.vscode("editor.action.formatS`
imports fix
`user.vscode("editor.action.organiz`
problem next
`user.vscode("editor.action.marker.`
problem last
`user.vscode("editor.action.marker.`
problem fix
`user.vscode("problems.action.showQ`
rename that
`user.vscode("editor.action.rename"`
refactor that
`user.vscode("editor.action.refactc`
whitespace trim
`user.vscode("editor.action.trimTra`
language switch
`user.vscode("workbench.action.edit`
refactor rename
`user.vscode("editor.action.rename"`
refactor this
`user.vscode("editor.action.refactc`
(go declaration | follow)
`user.vscode("editor.action.revealI`
go back
`user.vscode("workbench.action.navi`
go forward
`user.vscode("workbench.action.navi`
go implementation
`user.vscode("editor.action.goToImp`
go type
`user.vscode("editor.action.goToTyp`
go usage `user.vscode("references-`
`view.find")`
go recent [<user.text>]
`user.vscode("workbench.action.oper`
`sleep(50ms) insert(text`
`or "") sleep(250ms)`
go marks
`user.vscode("workbench.view.extens`
toggle mark
`user.vscode("bookmarks.toggle")`
go next mark
`user.vscode("bookmarks.jumpToNext"`
go last mark
`user.vscode("bookmarks.jumpToPrevi`
fold that
`user.vscode("editor.fold")`
unfold that
`user.vscode("editor.unfold")`
fold those
`user.vscode("editor.foldAllMarkerR`
unfold those
`user.vscode("editor.unfoldRecursiv`
fold all
`user.vscode("editor.foldAll")`
unfold all
`user.vscode("editor.unfoldAll")`
fold comments
`user.vscode("editor.foldAllBlockCo`
git branch
`user.vscode("git.branchFrom")`
git branch this
`user.vscode("git.branch")`
git checkout [<user.text>]
`user.vscode("git.checkout")`
`sleep(50ms) insert(text`
`or "")`
git commit [<user.text>]
`user.vscode("git.commitStaged")`
`sleep(100ms)`
`user.insert_formatted(text or "",`
`"CAPITALIZE_FIRST_WORD")`
git commit undo
`user.vscode("git.undoCommit")`
git commit ammend
`user.vscode("git.commitStagedAmenc`
git diff
`user.vscode("git.openChange")`

git ignore
`user.vscode("git.ignore")`
git merge `user.vscode("git.merge")`
git output
`user.vscode("git.showOutput")`
git pull
`user.vscode("git.pullRebase")`
git push `user.vscode("git.push")`
git push focus
`user.vscode("git.pushForce")`
git rebase abort
`user.vscode("git.rebaseAbort")`
git reveal
`user.vscode("git.revealInExplorer"`
git revert
`user.vscode("git.revertChange")`
git stash `user.vscode("git.stash")`
git stash pop
`user.vscode("git.stashPop")`
git status
`user.vscode("workbench.scm.focus")`
git stage `user.vscode("git.stage")`
git stage all
`user.vscode("git.stageAll")`
git unstage
`user.vscode("git.unstage")`
git unstage all
`user.vscode("git.unstageAll")`
pull request
`user.vscode("pr.create")`
change next `key(alt-f5)`
change last `key(shift-alt-f5)`
break point
`user.vscode("editor.debug.action.t`
step over
`user.vscode("workbench.action.debu`
debug step into
`user.vscode("workbench.action.debu`
debug step out [of]
`user.vscode("workbench.action.debu`
debug start
`user.vscode("workbench.action.debu`
debug pause
`user.vscode("workbench.action.debu`
debug stopper
`user.vscode("workbench.action.debu`
debug continue
`user.vscode("workbench.action.debu`
debug restart
`user.vscode("workbench.action.debu`
debug console
`user.vscode("workbench.debug.actic`
terminal external
`user.vscode("workbench.action.term`
terminal new
`user.vscode("workbench.action.term`
terminal next
`user.vscode("workbench.action.term`
terminal last
`user.vscode("workbench.action.term`
terminal split
`user.vscode("workbench.action.term`
terminal zoom
`user.vscode("workbench.action.togg`
terminal trash
`user.vscode("workbench.action.term`
terminal toggle
`user.vscode_and_wait("workbench.ac`
terminal scroll up
`user.vscode("workbench.action.term`
terminal scroll down
`user.vscode("workbench.action.term`
terminal <number_small>
`user.vscode_terminal(number_small)`
copy line down
`user.vscode("editor.action.copyLir`
copy line up
`user.vscode("editor.action.copyLir`
select less
`user.vscode("editor.action.smartSe`
select (more|this)
`user.vscode("editor.action.smartSe`
minimap
`user.vscode("editor.action.toggleM`
maximize
`user.vscode("workbench.action.mini`
restore
`user.vscode("workbench.action.ever`
replace here `user.replace("")`
`key(cmd-alt-l)`
hover show
`user.vscode("editor.action.showHov`
join lines
`user.vscode("editor.action.joinLir`
full screen
`user.vscode("workbench.action.togg`
curse undo
`user.vscode("cursorUndo")`
select word
`user.vscode("editor.action.addSele`
skip word
`user.vscode("editor.action.moveSel`
cell next
`user.vscode("jupyter.gotoNextCellI`
cell last
`user.vscode("jupyter.gotoPrevCellI`
cell run above
`user.vscode("jupyter.runallcellsab`
cell run
`user.vscode("jupyter.runcurrentcel`
install local
`user.vscode("workbench.extensions.`

github

focus search `key(s)`
go to notifications `insert("gn")`
go to dashboard `insert("gd")`
show keyboard shortcuts `key(?)`
move selection down `key(j)`
move selection up `key(k)`
toggle selection `key(x)`
open selection `key(o)`
go to code `insert("gc")`
go to issues `insert("gi")`
go to pull requests `insert("gp")`
go to wiki `insert("gw")`
find file `key(t)`
jump to line `key(l)`
switch (branch|tag) `key(w)`
expand url `key(y)`
(show|hide) [all] in line notes `key(i)`
create [an] issue `key(c)`
search (issues|[pull] requests) `key(/)`
(filter by|edit) labels `key(l)`
(filter by|edit) milestones `key(m)`
(filter by|edit) assignee `key(a)`
reply `key(r)`
submit comment `key(ctrl-enter)`
preview comment `key(ctrl-shift-p)`
git hub full screen `key(ctrl-shift-l)`
close form `key(escape)`
parent commit `key(p)`
other parent commit `key(o)`
mark as read `key(y)`
mute thread `key(shift-m)`
open issue `key(o)`

gitlab

show shortcuts `key(?)`
go to projects [page] `key(shift-p)`
go to groups [page] `key(shift-g)`
go to activity [page] `key(shift-a)`
go to milestones [page] `key(shift-l)`
go to snippets [page] `key(shift-s)`
search page `key(s)`
go to issues [page] `key(shift-i)`
go to merge requests [page]
`key(shift-m)`
go to to do [list] [page] `key(shift-t)`
(show|hide) performance bar `key(p)`
edit last comment `key(1)`
toggle mark down [preview]
`key(ctrl-shift-p)`
go [to] project home [page]
`insert("gp")`
go [to] project activity [feed]
`insert("gv")`
go [to] project releases [list]
`insert("gr")`
go [to] project files [list] `insert("gf")`
go [to] project file search [page]
`key(t)`
go [to] project (commit|commits) [list]

insert("gc")
go [to] (repository|repo) graph [page] insert("gn")
go [to] (repository|repo) charts insert("gd")
go [to] project issues [list] insert("gi")
go [to] new issues [list] insert("i")
go [to] project issues boards [list] insert("gb")
go [to] project merge requests [list] insert("gm")
go [to] jobs [list] insert("gj")
go [to] project metrics insert("gl")
go [to] project environments insert("ge")
go [to] project cubes insert("gk")
go [to] project snippets [list] insert("gs")
go [to] project wiki insert("gw")
edit description key(e)
change assignee key(a)
change milestone key(m)
change label key(l)
right comment key(r)
next [unresolved] discussion key(n)
previous [unresolved] discussion key(p)
next file key(])
previous file key([)
back to files key(escape)
open permalink key(y)
edit page key(e)

outlook

new message key(n)
send [this] message key(alt-s)
reply [to] [this] message key(r)
reply all [to] [this] message key(ctrl-shift-r)
forward [this] message key(ctrl-shift-f)
save [draft] key(ctrl-s)
discard [draft] key(esc)
insert [a] [hyper] link key(ctrl-k)
(select|unselect) [this] message key(ctrl-space)
select all [messages] key(ctrl-a)
clear all [messages] key(esc)
select first [message] key(home)
select last [message] key(and)
open [this] message key(o)
open [this] message [in] [a] new window key(shift-enter)
close [this] message key(esc)
[open] [the] next (item|message) key(ctrl-.)
[open] [the] (prev|previous) item key(ctrl-,)
next reading [pane] (item|message) key(.)
(prev|previous) [pane] (item|message) key(,)
(expand|collapse) [conversation] key(x)
go [to] mail key(ctrl-shift-1)
go [to] calendar key(ctrl-shift-2)
go [to] people key(ctrl-shift-3)
go [to] to do key(ctrl-shift-4)
go [to] inbox key(g) key(i)
go to drafts key(g) key(d)
go to sent key(g) key(s)
search [email] key(alt-q)
show help key(?)
undo [last] [action] key(ctrl-z)
delete [this] [message] key(delete)
(perm|permanently) delete [this] [message] key(shift+delete)
new folder key(shift-e)
mark [this] [(item|message)] as read key(q)
mark [this] [(item|message)] as unread key(u)
flag [this] [(item|message)] key(insert)
archive key(e)
mark [this] [message] [as] junk key(j)
moved to [a] folder key(v)
categorize [this] message key(c)

protonmail

open help key(?)
[focus] search key(/)
confirm active key(enter)
close active key(escape)
open command [palette] key(shift-space)
new message key(c)
send message key(ctrl-enter)
save message key(ctrl-s)
(go|jump) [to] inbox key(g) key(i)
(go|jump) [to] draft key(g) key(d)
(go|jump) [to] sent key(g) key(s)
(go|jump) [to] starred key(g) key(.)
(go|jump) [to] archive key(g) key(a)
(go|jump) [to] spam key(g) key(x)
(go|jump) [to] trash key(g) key(t)
(prev|previous) message key(up)
next message key(down)
exit message key(left)
enter message key(right)
(show|display) newer [message] key(k)
(show|display) older [message] key(j)
open message key(enter)
go back key(escape)
select all key(*) key(a)
(deselect|unselect) all key(*) key(n)
select [the] (message|conversation) key(x)
mark [as] read key(r)
mark [as] unread key(u)
star (message|conversation) key(.)
move to inbox key(i)
move to trash key(t)
move to archive key(a)
move to spam key(s)
reply to (message|conversation) key(shift-r)
reply all [to] (message|conversation) key(shift-a)
forward (message|conversation) key(shift-f)
(prev|previous) contact key(up)
next contact key(down)
enter contact key(right)
delete contact key(t)
exit contact key(left)
save contact key(ctrl-s)

twitter

(show shortcuts|shortcuts help) key(?)
next tweet key(j)
previous tweet key(k)
page down key(space)
load new tweet key(.)
go home insert("gh")
go explore insert("ge")
go notifications insert("gn")
go mentions insert("gr")
go profile insert("gp")
go likes insert("gl")
go lists insert("gi")
go direct messages insert("gm")
go settings insert("gs")
go book marks insert("gb")
go to user insert("gu")
display settings insert("gd")
new tweet key(n)
send tweet key(ctrl-enter)
new direct message key(m)

search key(/)
like message key(l)
reply message key(r)
re tweet [message] key(t)
share tweet key(s)
bookmark key(b)
mute account key(urge)
block account key(x)
open details key(enter)
expand photo key(o)

win explorer

go <user.letter>
user.file_manager_open_volume("{letter}:")
go app data
user.file_manager_open_directory("
go program files
user.file_manager_open_directory("

win outlook

archive key(alt h o 1)
new e-mail key(ctrl-n)
calendar key(ctrl-2)
inbox key(ctrl-1)
Reply key(ctrl-r)
Reply all key(ctrl-shift-r)
Forward key(ctrl-f)
accept key(shift-f10 c c enter)

win windows terminal

settings open key(ctrl-,)
focus left key(ctrl-alt-shift-left)
focus right key(ctrl-alt-shift-right)
focus up key(ctrl-alt-shift-up)
focus down key(ctrl-alt-shift-down)
split right key(ctrl-shift-h)
split down key(ctrl-h)
term menu key(ctrl-shift-f1)
run last key(up enter)
kill all key(ctrl-c) insert("y") key(enter)

win ubuntu

go <user.letter>
user.file_manager_open_volume("/mnt/{letter}")

win windbg

register <user.registers> key(@) insert("{registers}")
open help insert(".hh\n")
add microsoft symbols insert("srv*C:\\symbols*http://msc/download/symbols;\n")
force reload symbols insert(".reload /f\n")
reload symbols insert(".reload\n")
loaded modules insert("lm l\n")
display pointers insert("dps ")
dereference pointer insert("poi()") edit.left()
show version key(ctrl-alt-w)
view command key(alt-1)
view watch key(alt-2)
view locals key(alt-3)
view registers key(alt-4)
view memory key(alt-5)
view call stack key(alt-6)
view disassembly key(alt-7)
view scratch pad key(alt-8)
view (processes|threads) key(alt-9)
dump function params "r @rcx,@rdx,@r8,@r9\n"
(lib|library) <user.windows_dlls> "{windows_dlls}"

cheatsheet

print cheatsheet user.cheatsheet()

batch

soft exit "exit /B 1\n"
hard exit "exit 1\n"
echo "echo "
echo off "@echo off\n"
call "call "
call shell "call cmd \\c "
if error "if errorlevel 1 "
go to "goto "
delayed expansion "SETLOCAL EnableDelayedExpansion\n"
arg <number_small> "%{number_small}"

block comment

block comment user.code_block_comment()
block comment line edit.line_start() user.code_block_comment_prefix() key(space) edit.line_end() key(space) user.code_block_comment_suffix()
block comment line <user.text> over edit.line_start() user.code_block_comment() insert(user.text)
block comment <user.text> over user.code_block_comment() insert(user.text)
block comment <user.text> user.code_block_comment() insert(user.text)
(line | inline) block comment <user.text> over edit.line_end() user.code_block_comment_prefix() key(space) insert(user.text) key(space) user.code_block_comment_suffix()
(line | inline) block comment <user.text> edit.line_end() user.code_block_comment_prefix() key(space) insert(user.text) key(space) user.code_block_comment_suffix()
open block comment user.code_block_comment_prefix()
close block comment user.code_block_comment_suffix()

c

funky <user.text> user.code_default_function(text)
static funky <user.text> user.code_private_static_function(
state define "#define "
state undefine "#undef "
state if define "#ifdef "
state pre if "#if "
state error "#error "
state pre else if "#elif "
state pre end "#endif "
state pragma "#pragma "
state default "default:\nbreak;"
push brackets edit.line_end() insert("{}") edit.left() key(enter) key(enter) edit.up()
<user.c_variable> <phrase> insert("{c_variable} ")

insert(user.formatted_text(phrase, "PRIVATE_CAMEL_CASE,NO_SPACES"))
<user.c_variable> <user.letter>
insert("{c_variable} {letter} ")
cast to <user.c_cast> "{c_cast}"
standard cast to <user.stdint_cast>
"{stdint_cast}"
<user.c_types> "{c_types}"
<user.c_pointers> "{c_pointers}"
<user.c_signed> "{c_signed}"
standard <user.stdint_types>
"{stdint_types}"
int main insert("int main()")
edit.left()
toggle includes
user.code_toggle_libraries()
include <user.code_libraries>
user.code_insert_library(code_libr
"") key(end enter)

comment

comment user.code_comment()
comment line edit.line_start()
user.code_comment()
comment line <user.text> over
edit.line_start()
user.code_comment()
insert(user.text)
insert(" ")
comment <user.text> over
user.code_comment()
insert(user.text)
comment <user.text>
user.code_comment()
insert(user.text)
(line | inline) comment <user.text>
over edit.line_end()
user.code_comment()
insert(user.text)
(line | inline) comment <user.text>
edit.line_end()
user.code_comment()
insert(user.text)

csharp

funky <user.text>
user.code_default_function(text)
pro funky <user.text>
user.code_protected_function(text)
pub funky <user.text>
user.code_public_function(text)
static funky <user.text>
user.code_private_static_function(
pro static funky <user.text>
user.code_protected_static_functic
pub static funky <user.text>
user.code_public_static_function(t

go

variadic "..."
logical and " && "
logical or " || "
state comment "// "
[line] comment <user.text> key("cmd-right") insert(" // ")
insert(user.formatted_text(text, "sentence"))
state (funk | func | fun) "func "
function (Annette | init) [over] "func init() {\n"
function <user.text> [over]
insert("func ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
insert("(") sleep(100ms)
method <user.text> [over]
insert("meth ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
sleep(100ms)
state var "var "
variable [<user.text>] [over]
insert("var ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
sleep(100ms)
of type [<user.text>] [over] insert(" ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state break "break"
state (chan | channel) " chan "
state go "go "
state if "if "
if <user.text> [over] insert("if ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
spawn <user.text> [over] insert("go ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state else if " else if "
else if <user.text> [over] insert(" else if ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state else " else "
else <user.text> [over] insert(" else {") key("enter")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state while "while "
while <user.text> [over]
insert("while ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state for "for "
for <user.text> [over] insert("for ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state for range "forr "
range <user.text> [over] insert("forr ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state format "fmt"
format <user.text> [over]
insert("fmt.")
insert(user.formatted_text(text, "PUBLIC_CAMEL_CASE"))
state switch "switch "
switch <user.text> [over]
insert("switch ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state select "select "
state (const | constant) " const "
constant <user.text> [over]
insert("const ")
insert(user.formatted_text(text, "PUBLIC_CAMEL_CASE"))
state case " case "
state default " default:"
case <user.text> [over] insert("case ")
insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))
state type " type "
type <user.text> [over] insert("type ")
insert(user.formatted_text(text, "PUBLIC_CAMEL_CASE"))
state true " true "
state false " false "
state (start | struct | struck) insert(" struct {") key("enter")
(struct | struck) <user.text> [over]
insert(" struct {")
key("enter")
insert(user.formatted_text(text, "PUBLIC_CAMEL_CASE"))
[state] empty interface " interface{} "
state interface insert(" interface {") key("enter")
interface <user.text> [over] insert(" interface {")
key("enter")
insert(user.formatted_text(text, "PUBLIC_CAMEL_CASE"))

state string `" string "`
[state] (int | integer | ant) `"int"`
state slice `" []"`
slice of `"[]"`
[state] (no | nil) `"nil"`
state (int | integer | ant) 64 `" int64 "`
state tag `insert(" ``")`
`key("left")`
field tag <user.text> [over] `insert("``") key("left")`
`sleep(100ms)`
`insert(user.formatted_text(text, "snake"))  insert(" ")`
`sleep(100ms)`
state return `" return "`
return <user.text> [over]
`insert("return ")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
map of string to string `"`
`map[string]string "`
map of <user.text> [over]
`insert("map[")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
`key("right")  sleep(100ms)`
receive `" <- "`
make `"make("`
loggers [<user.text>] [over]
`insert("logrus.")`
`insert(user.formatted_text(text, "PUBLIC_CAMEL_CASE"))`
length <user.text> [over]
`insert("len(")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
append <user.text> [over]
`insert("append(")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
state (air | err) `"err"`
error `" err "`
loop over [<user.text>] [over]
`insert("forr ")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
item <user.text> [over] `insert(", ")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
value <user.text> [over] `insert(": ")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
address of [<user.text>] [over]
`insert("&")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
pointer to [<user.text>] [over]
`insert("*")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`
swipe [<user.text>] [over]
`key("right")  insert(", ")`
`insert(user.formatted_text(text, "PRIVATE_CAMEL_CASE"))`

javascript

(op | is) strict equal `" === "`
(op | is) strict not equal `" !== "`
state const `"const "`
state let `"let "`
state var `"var "`
state async `"async "`
state await `"await "`
state map `insert(".map()")`
`key(left)`
state filter `insert(".filter()")`
`key(left)`
state reduce `insert(".reduce()")`
`key(left)`
state spread `"..."`
funky <user.text>
`user.code_default_function(text)`
pro funky <user.text>
`user.code_protected_function(text)`
pub funky <user.text>
`user.code_public_function(text)`

operators

op dereference
`user.code_operator_indirection()`
op address of
`user.code_operator_address_of()`
op arrow
`user.code_operator_structure_deref`
op lambda
`user.code_operator_lambda()`
op subscript
`user.code_operator_subscript()`
op (equals | assign)
`user.code_operator_assignment()`
op (minus | subtract)
`user.code_operator_subtraction()`
op (minus | subtract) equals
`user.code_operator_subtraction_ass`
op (plus | add)
`user.code_operator_addition()`
op (plus | add) equals
`user.code_operator_addition_assigr`
op (times | multiply)
`user.code_operator_multiplication(`
op (times | multiply) equals
`user.code_operator_multiplication_`
op divide
`user.code_operator_division()`
op divide equals
`user.code_operator_division_assigr`
op mod
`user.code_operator_modulo()`
op mod equals
`user.code_operator_modulo_assignme`
(op (power | exponent) | to the power [of]) `user.code_operator_exponent()`
(op | is) equal
`user.code_operator_equal()`
(op | is) not equal
`user.code_operator_not_equal()`
(op | is) (greater | more)
`user.code_operator_greater_than()`
(op | is) (less | below) [than]
`user.code_operator_less_than()`
(op | is) greater [than] or equal
`user.code_operator_greater_than_or`
(op | is) less [than] or equal
`user.code_operator_less_than_or_eq`
(op | is) in `user.code_operator_in()`
(op | logical) and
`user.code_operator_and()`
(op | logical) or
`user.code_operator_or()`
[op] bitwise and
`user.code_operator_bitwise_and()`
[op] bitwise or
`user.code_operator_bitwise_or()`
(op | logical | bitwise) (ex | exclusive) or
`user.code_operator_bitwise_exclusi`
(op | logical | bitwise) (left shift | shift left)
`user.code_operator_bitwise_left_sh`
(op | logical | bitwise) (right shift | shift right)
`user.code_operator_bitwise_right_s`
(op | logical | bitwise) (ex | exclusive) or equals
`user.code_operator_bitwise_exclusi`
[(op | logical | bitwise)] (left shift | shift left) equals
`user.code_operator_bitwise_left_sh`
[(op | logical | bitwise)] (left right | shift right) equals
`user.code_operator_bitwise_right_s`
(op | pad) colon `" : "`

programming

block `user.code_block()`
is not (none|null)
`user.code_is_not_null()`
is (none|null) `user.code_is_null()`
state if `user.code_state_if()`
state else if

user.code_state_else_if()
state else user.code_state_else()
state self user.code_self()
self dot user.code_self()
insert(".")
state while user.code_state_while()
state for user.code_state_for()
state for in
user.code_state_for_each()
state switch
user.code_state_switch()
state case user.code_state_case()
state do user.code_state_do()
state goto user.code_state_go_to()
state return
user.code_state_return()
state import user.code_import()
from import
user.code_from_import()
state class user.code_type_class()
state include user.code_include()
state include system
user.code_include_system()
state include local
user.code_include_local()
state type deaf
user.code_type_definition()
state type deaf struct
user.code_typedef_struct()
state (no | nil | null) user.code_null()
state break user.code_break()
state next user.code_next()
state true user.code_true()
state false user.code_false()
toggle funk
user.code_toggle_functions()
funk <user.code_functions>
user.code_insert_function(code_fur
"")
funk cell <number>
user.code_select_function(number
- 1, "")
funk wrap <user.code_functions>
user.code_insert_function(code_fur
edit.selected_text())
funk wrap <number>
user.code_select_function(number
- 1, edit.selected_text())
dock string
user.code_document_string()

python

dunder in it "__init__"
state (def | deaf | deft) "def "
self taught "self."
pie test "pytest"
state past "pass"
funky <user.text>
user.code_default_function(text)
pub funky <user.text>
user.code_public_function(text)
raise {user.python_exception}
user.insert_cursor("raise
{python_exception}([|])")
is type {user.python_type_list}
insert(": {python_type_list}")
returns [type] {user.python_type_list}
insert(" -> {python_type_list}")
type {user.python_type_list} insert("
{python_type_list}")
dock {user.python_docstring_fields}
insert("
{python_docstring_fields}")
edit.left()
dock type {user.python_type_list}
user.insert_cursor(":type [|]:
{python_type_list}")
dock returns type
{user.python_type_list}
user.insert_cursor(":rtype [|]:
{python_type_list}")
toggle imports
user.code_toggle_libraries()
import <user.code_libraries>
user.code_insert_library(code_libr
"") key(end enter)

r

toggle library
user.code_toggle_libraries()
library <user.code_libraries>
user.code_insert_library(code_libr
"") key(end enter)
(chain|pipe that) key(end) "
%>%" key(enter)
state na insert("NA")
function define <user.text>
user.code_private_function(text)

snippets

snip {user.snippets}
user.snippet_insert(user.snippets)
snip hunt <user.text>
user.snippet_search(user.text)
snip hunt user.snippet_search("")
snip create user.snippet_create()
snip show user.snippet_toggle()

sql

select "SELECT "
star "*"
from "FROM "
select star from "SELECT * FROM "
where "WHERE "
order by "ORDER BY "
descending " DESC"
ascending " ASC"
dot i d ".id"
is not null " IS NOT NULL"
is null " IS NULL"
inner join insert("INNER JOIN ON
") key(left)
key(left) key(left)
key(left)

talon

dot talon insert(".talon")
action block insert("action():")
edit.left() edit.left()
setting block
insert("settings():\n\t")
win require insert("os:
windows\n")
mac require insert("os: mac\n")
linux require insert("os: linux\n")
title require insert("win.title: ")
app require insert("app: ")
tag require insert("tag: ")
tag set insert("tag(): ")
key <user.keys> over "{keys}"
key <user.modifiers> over
"{modifiers}"
toggle funk
user.code_toggle_functions()
funk <user.code_functions>
user.code_insert_function(code_fur
"")
funk cell <number>
user.code_select_function(number
- 1, "")
funk wrap <user.code_functions>
user.code_insert_function(code_fur
edit.selected_text())
funk wrap <number>
user.code_select_function(number
- 1, edit.selected_text())

typescript

(op | is) strict equal " === "
(op | is) strict not equal " !== "
state const "const "
state let "let "
state var "var "

state async `"async "`
state await `"await "`
state map `insert(".map()") key(left)`
state filter `insert(".filter()") key(left)`
state reduce `insert(".reduce()") key(left)`
state spread `"..."`
funky <user.text> `user.code_default_function(text)`
pro funky <user.text> `user.code_protected_function(text)`
pub funky <user.text> `user.code_public_function(text)`

vimscript

assign [<user.vimscript_scope>] (variable|var) [<user.text>] [over] `insert("let ") insert(vimscript_scope or '') user.code_private_variable_formatt`
[<user.vimscript_scope>] (variable|var) [<user.text>] [over] `insert(vimscript_scope or '') user.code_private_variable_formatt`
<user.vimscript_functions> `insert("{vimscript_functions} ")`
state command `"command! "`
state end if `"endif"`
state end for `"endfor"`
state end while `"endwhile"`
state end function `"endfunction"`
state continue `"continue"`

abbreviate

(abbreviate|abreviate|brief) {user.abbreviation} `"{abbreviation}"`

extensions

dot pie `".py"`
dot talon `".talon"`
dot mark down `".md"`
dot shell `".sh"`
dot vim `".vim"`
dot see `".c"`
dot see sharp `".cs"`
dot com `".com"`
dot net `".net"`
dot org `".org"`
dot exe `".exe"`
dot (bin | bend) `".bin"`
dot (jason | jay son) `".json"`

formatters

phrase <user.text> `user.insert_formatted(text, "NOOP")`
phrase <user.text> over `user.insert_formatted(text, "NOOP")`
{user.prose_formatter} <user.prose> `user.insert_formatted(prose, prose_formatter)`
{user.prose_formatter} <user.prose> over `user.insert_formatted(prose, prose_formatter)`
<user.format_text>+ `user.insert_many(format_text_list)`
<user.format_text>+ over `user.insert_many(format_text_list)`
<user.formatters> that `user.formatters_reformat_selectior`
word <user.word> `user.insert_formatted(user.word, "NOOP")`
format help `user.formatters_help_toggle()`
recent list `user.toggle_phrase_history()`
recent repeat <number_small> `insert(user.get_recent_phrase(numb`
recent copy <number_small> `clip.set_text(user.get_recent_phra`
select that `user.select_last_phrase()`
nope that | scratch that `user.clear_last_phrase()`
nope that was <user.formatters> `user.formatters_reformat_last(form`

git

git add patch `"git add . -p\n"`
git add `"git add "`
git add everything `"git add -u\n"`
git bisect `"git bisect "`
git blame `"git blame "`
git branch `"git branch "`
git remote branches `"git branch --remote\n"`
git branch <user.text> `"git branch {text}"`
git checkout `"git checkout "`
git checkout master `"git checkout master\n"`
git checkout main `"git checkout main\n"`
git checkout <user.text> `"git checkout {text}"`
git cherry pick `"git cherry-pick "`
git cherry pick continue `"git cherry-pick --continue "`
git cherry pick abort `"git cherry-pick --abort "`
git cherry pick skip `"git cherry-pick --skip "`
git clone `"git clone "`
git clean everything `"git clean -dfx"`
git commit message <user.text> `"git commit -m '{text}'"`
git commit `"git commit\n"`
git diff (colour|color) words `"git diff --color-words "`
git diff `"git diff "`
git diff cached `"git diff --cached\n"`
git fetch `"git fetch\n"`
git fetch <user.text> `"git fetch {text}"`
git fetch prune `"git fetch --prune\n"`
git in it `"git init\n"`
git log all `"git log\n"`
git log all changes `"git log -c\n"`
git log `"git log "`
git log changes `"git log -c "`
git merge `"git merge "`
git merge <user.text> `"git merge {text}"`
git move `"git mv "`
git new branch `"git checkout -b "`
git pull `"git pull\n"`
git pull origin `"git pull origin "`
git pull rebase `"git pull --rebase\n"`
git pull fast forward `"git pull --ff-only\n"`
git pull <user.text> `"git pull {text} "`
git push `"git push\n"`
git push origin `"git push origin "`
git push up stream origin `"git push -u origin"`
git push <user.text> `"git push {text} "`
git push tags `"git push --tags\n"`
git rebase `"git rebase\n"`
git rebase continue `"git rebase --continue"`
git rebase skip `"git rebase --skip"`
git remove `"git rm "`
git (remove|delete) branch `"git branch -d "`
git (remove|delete) remote branch

"git push --delete origin "
git reset "git reset "
git reset soft "git reset --soft "
git reset hard "git reset --hard "
git restore "git restore "
git restore staged "git restore --staged "
git remote show origin "git remote show origin\n"
git remote add upstream "git remote add upstream "
git show "git show "
git stash pop "git stash pop\n"
git stash "git stash\n"
git stash apply "git stash apply\n"
git stash list "git stash list\n"
git stash show "git stash show"
git status "git status\n"
git submodule add "git submodule add "
git tag "git tag "
git edit config "git config --local -e\n"
git clone clipboard insert("git clone ") edit.paste() key(enter)
git diff highlighted edit.copy() insert("git diff ") edit.paste() key(enter)
git diff clipboard insert("git diff ") edit.paste() key(enter)
git add highlighted edit.copy() insert("git add ") edit.paste() key(enter)
git add clipboard insert("git add ") edit.paste() key(enter)
git commit highlighted edit.copy() insert("git add ") edit.paste() insert("\ngit commit\n")

git add patch

yank key(y) key(enter)
near key(n) key(enter)
quench key(q) key(enter)
drum key(d) key(enter)
air key(a) key(enter)

help

help alphabet user.help_alphabet(user.get_alphat
help context user.help_context()
help active user.help_context_enabled()
help search <user.text> user.help_search(text)
help context {user.help_contexts} user.help_selected_context(help_cc

help open

help next user.help_next()
help previous user.help_previous()
help <number> user.help_select_index(number - 1)
help return user.help_return()
help close user.help_hide()

history

command history user.history_toggle()
command history clear user.history_clear()
command history less user.history_less()
command history more user.history_more()

keys

go <user.arrow_keys> key(arrow_keys)
<user.letter> key(letter)
(ship | uppercase) <user.letters> [(lowercase | sunk)] user.insert_formatted(letters, "ALL_CAPS")
<user.symbol_key> key(symbol_key)
<user.function_key> key(function_key)
<user.special_key> key(special_key)
<user.modifiers> <user.unmodified_key> key("{modifiers}-{unmodified_key}")

mac macro

macro record user.macro_record()
macro stop user.macro_stop()
macro play user.macro_play()

media

volume up key(volup)
volume down key(voldown)
set volume <number> user.media_set_volume(number)
(volume|media) mute key(mute)
media play next key(next)
media play previous key(prev)
media (play | pause) key(play)

messaging

previous (workspace | server) user.messaging_workspace_previous(
next (workspace | server) user.messaging_workspace_next()
channel user.messaging_open_channel_picker
channel <user.text> user.messaging_open_channel_picker insert(user.formatted_text(user.te "ALL_LOWERCASE"))
channel up user.messaging_channel_previous()
channel down user.messaging_channel_next()
([channel] unread last | gopreev) user.messaging_unread_previous()
([channel] unread next | goneck) user.messaging_unread_next()
go (find | search) user.messaging_open_search()
mark (all | workspace | server) read user.messaging_mark_workspace_reac
mark channel read user.messaging_mark_channel_read()
upload file user.messaging_upload_file()

microphone selection

microphone show user.microphone_selection_toggle()
microphone pick <number_small> user.microphone_select(number_smal

mouse

control mouse user.mouse_toggle_control_mouse()
zoom mouse user.mouse_toggle_zoom_mouse()
camera overlay user.mouse_toggle_camera_overlay()

run calibration
`user.mouse_calibrate()`
touch `mouse_click(0)`
`user.grid_close()`
righty `mouse_click(1)`
`user.grid_close()`
midclick `mouse_click(2)`
`user.grid_close()`
<user.modifiers> touch
`key("{modifiers}:down")`
`mouse_click(0)`
`key("{modifiers}:up")`
`user.grid_close()`
<user.modifiers> righty
`key("{modifiers}:down")`
`mouse_click(1)`
`key("{modifiers}:up")`
`user.grid_close()`
(dubclick | duke) `mouse_click()`
`mouse_click()`
`user.grid_close()`
(tripclick | triplick) `mouse_click()`
`mouse_click()`
`mouse_click()`
`user.grid_close()`
drag `user.mouse_drag()`
`user.grid_close()`
wheel down
`user.mouse_scroll_down()`
wheel down here
`user.mouse_move_center_active_winc`
`user.mouse_scroll_down()`
wheel tiny [down] `mouse_scroll(20)`
wheel tiny [down] here
`user.mouse_move_center_active_winc`
`mouse_scroll(20)`
wheel downer
`user.mouse_scroll_down_continuous(`
wheel downer here
`user.mouse_move_center_active_winc`
`user.mouse_scroll_down_continuous(`
wheel up `user.mouse_scroll_up()`
wheel up here
`user.mouse_scroll_up()`
wheel tiny up `mouse_scroll(-20)`
wheel tiny up here
`user.mouse_move_center_active_winc`
`mouse_scroll(-20)`
wheel upper
`user.mouse_scroll_up_continuous()`
wheel upper here
`user.mouse_move_center_active_winc`
`user.mouse_scroll_up_continuous()`
wheel gaze
`user.mouse_gaze_scroll()`
wheel gaze here
`user.mouse_move_center_active_winc`
`user.mouse_gaze_scroll()`
wheel stop
`user.mouse_scroll_stop()`
wheel stop here
`user.mouse_move_center_active_winc`
`user.mouse_scroll_stop()`
wheel left `mouse_scroll(0, -40)`
wheel left here
`user.mouse_move_center_active_winc`
`mouse_scroll(0, -40)`
wheel tiny left `mouse_scroll(0, -20)`
wheel tiny left here
`user.mouse_move_center_active_winc`
`mouse_scroll(0, -20)`
wheel right `mouse_scroll(0, 40)`
wheel right here
`user.mouse_move_center_active_winc`
`mouse_scroll(0, 40)`
wheel tiny right `mouse_scroll(0, 20)`
wheel tiny right here
`user.mouse_move_center_active_winc`
`mouse_scroll(0, 20)`
curse yes `user.mouse_show_cursor()`
curse no `user.mouse_hide_cursor()`
copy mouse position
`user.copy_mouse_position()`

multiple cursors

cursor multiple
`user.multi_cursor_enable()`
cursor stop
`user.multi_cursor_disable()`
cursor up
`user.multi_cursor_add_above()`
cursor down
`user.multi_cursor_add_below()`
cursor less
`user.multi_cursor_select_fewer_occ`
cursor more
`user.multi_cursor_select_more_occu`
cursor all
`user.multi_cursor_select_all_occur`
cursor lines
`user.multi_cursor_add_to_line_ends`

repeater

<user.ordinals>
`core.repeat_command(ordinals-1)`
(repeat that|twice)
`core.repeat_command(1)`
repeat that <number_small> [times]
`core.repeat_command(number_small)`

screenshot

grab window
`user.screenshot_window()`
grab screen `user.screenshot()`
grab selection
`user.screenshot_selection()`
grab window clip
`user.screenshot_window_clipboard()`
grab screen clip
`user.screenshot_clipboard()`

search engines

{user.search_engine} hunt <user.text>
`user.search_with_search_engine(sea`
`user.text)`
{user.search_engine} (that|this) `text`
`= edit.selected_text()`
`user.search_with_search_engine(sea`
`text)`

splits

split right
`user.split_window_right()`
split left `user.split_window_left()`
split down
`user.split_window_down()`
split up `user.split_window_up()`
split (vertically | vertical)
`user.split_window_vertically()`
split (horizontally | horizontal)
`user.split_window_horizontally()`
split flip `user.split_flip()`
split window `user.split_window()`
split clear `user.split_clear()`
split clear all
`user.split_clear_all()`
split next `user.split_next()`
split last `user.split_last()`
go split <number>
`user.split_number(number)`

standard

zoom in `edit.zoom_in()`
zoom out `edit.zoom_out()`
scroll up `edit.page_up()`
scroll down `edit.page_down()`
copy that `edit.copy()`
cut that `edit.cut()`
paste that `edit.paste()`
undo that `edit.undo()`
redo that `edit.redo()`
paste match
`edit.paste_match_style()`

file save `edit.save()`
wipe `key(backspace)`
(pad | padding) `insert("  ")`
`key(left)`
slap `edit.line_end()`
`key(enter)`

tabs

tab (open | new) `app.tab_open()`
tab last `app.tab_previous()`
tab next `app.tab_next()`
tab close `app.tab_close()`
tab (reopen|restore)
`app.tab_reopen()`
go tab <number>
`user.tab_jump(number)`
go tab final `user.tab_final()`

talon helpers

talon copy context pie
`user.talon_add_context_clipboard_p`
talon copy context
`user.talon_add_context_clipboard()`
talon copy title `title = win.title()`
`clip.set_text(title)`
talon dump context `name =`
`app.name()  executable =`
`app.executable()  bundle =`
`app.bundle()  title =`
`win.title()  print("Name:`
`{name}")`
`print("Executable: {executable}")`
`print("Bundle: {bundle}")`
`print("Title: {title}")`

win window management

window (new|open)
`app.window_open()`
window next `app.window_next()`
window last `app.window_previous()`
window close `app.window_close()`
focus <user.running_applications>
`user.switcher_focus(running_applic`
running list
`user.switcher_toggle_running()`
launch <user.launch_applications>
`user.switcher_launch(launch_applic`
snap <user.window_snap_position>
`user.snap_window(window_snap_posit`
snap next [screen]
`user.move_window_next_screen()`
snap last [screen]
`user.move_window_previous_screen()`
snap screen <number>
`user.move_window_to_screen(number)`
snap <user.running_applications>
<user.window_snap_position>
`user.snap_app(running_applications`
`window_snap_position)`
snap <user.running_applications>
[screen] <number>
`user.move_app_to_screen(running_ap`
`number)`

dictation mode

press <user.keys> `key("{keys}")`
<user.prose> `auto_insert(prose)`
new line `"\n"`
new paragraph `"\n\n"`
cap <user.word> `result =`
`user.formatted_text(word,`
`"CAPITALIZE_FIRST_WORD")`
`auto_insert(result)`
go up <number_small> (line|lines)
`edit.up()`
`repeat(number_small - 1)`
go down <number_small> (line|lines)
`edit.down()`
`repeat(number_small - 1)`
go left <number_small> (word|words)
`edit.word_left()`
`repeat(number_small - 1)`
go right <number_small>
(word|words) `edit.word_right()`
`repeat(number_small - 1)`
go line start `edit.line_start()`
go line end `edit.line_end()`
select left <number_small>
(word|words)
`edit.extend_word_left()`
`repeat(number_small - 1)`
select right <number_small>
(word|words)
`edit.extend_word_right()`
`repeat(number_small - 1)`
select left <number_small>
(character|characters)
`edit.extend_left()`
`repeat(number_small - 1)`
select right <number_small>
(character|characters)
`edit.extend_right()`
`repeat(number_small - 1)`
clear left <number_small>
(word|words)
`edit.extend_word_left()`
`repeat(number_small - 1)`
`edit.delete()`
clear right <number_small>
(word|words)
`edit.extend_word_right()`
`repeat(number_small - 1)`
`edit.delete()`
clear left <number_small>
(character|characters)
`edit.extend_left()`
`repeat(number_small - 1)`
`edit.delete()`
clear right <number_small>
(character|characters)
`edit.extend_right()`
`repeat(number_small - 1)`
`edit.delete()`
formatted <user.format_text>
`user.dictation_insert_raw(format_t`
format selection <user.formatters>
`user.formatters_reformat_selectior`
scratch that
`user.clear_last_phrase()`
scratch selection `edit.delete()`
select that
`user.select_last_phrase()`
spell that <user.letters>
`auto_insert(letters)`
spell that <user.formatters>
<user.letters> `result =`
`user.formatted_text(letters,`
`formatters)`
`user.auto_format_pause()`
`auto_insert(result)`
`user.auto_format_resume()`
escape <user.text>
`auto_insert(user.text)`

dragon modes

dragon mode `user.dragon_mode()`
talon mode `user.talon_mode()`

language modes

force see sharp
`user.code_set_language_mode("cshar`
force see plus plus
`user.code_set_language_mode("cplus`
force go (lang|language)
`user.code_set_language_mode("go")`
force java
`user.code_set_language_mode("java"`
force java script
`user.code_set_language_mode("javas`
force type script
`user.code_set_language_mode("types`

force markdown
`user.code_set_language_mode("markc`
force python
`user.code_set_language_mode("pythc`
force are language
`user.code_set_language_mode("r")`
force talon [language]
`user.code_set_language_mode("talor`
clear language modes
`user.code_clear_language_mode()`
[enable] debug mode
`mode.enable("user.gdb")`
disable debug mode
`mode.disable("user.gdb")`

modes

dictation mode
```
mode.disable("sleep")
mode.disable("command")
mode.enable("dictation")
user.code_clear_language_mode()
mode.disable("user.gdb")
```
command mode
```
mode.disable("sleep")
mode.disable("dictation")
mode.enable("command")
```

sleep mode wav2letter

<phrase> `skip()`

wake up

welcome back `user.mouse_wake()`
```
user.history_enable()
user.talon_mode()
```
sleep all
```
user.switcher_hide_running()
user.history_disable()
user.homophones_hide()
user.help_hide()
user.mouse_sleep()
speech.disable()
user.engine_sleep()
```
talon sleep `speech.disable()`
talon wake `speech.enable()`

mouse grid

M grid `user.grid_select_screen(1)`
`user.grid_activate()`
grid win `user.grid_place_window()`
`user.grid_activate()`
grid <user.number_key>+
`user.grid_activate()`
`user.grid_narrow_list(number_key_l`
grid screen [<number>]
`user.grid_select_screen(number or 1)  user.grid_activate()`

mouse grid open

<user.number_key>
`user.grid_narrow(number_key)`
grid off `user.grid_close()`
grid reset `user.grid_reset()`
grid back `user.grid_go_back()`

win draft global

draft show `user.draft_hide()`
`user.draft_show()`
draft show <user.draft_window_position>
```
user.draft_hide()
user.draft_show()
user.draft_named_move(draft_windov
```
draft show small `user.draft_hide()`
```
user.draft_show()
user.draft_resize(600, 200)
```
draft show large `user.draft_hide()`
```
user.draft_show()
user.draft_resize(800, 500)
```
draft empty `user.draft_show("")`
draft edit `text = edit.selected_text()`
```
key(backspace)
user.draft_show(text)
```
draft edit all `edit.select_all()`
```
text = edit.selected_text()
key(backspace)
user.draft_show(text)
```

win draft window

replace <user.draft_anchor> with <user.text> `user.draft_select("{draft_anchor}")  result = user.formatted_text(text, "NOOP")`
`insert(result)`
cursor <user.draft_anchor>
`user.draft_position_caret("{draft_anchor}")`
cursor before <user.draft_anchor>
`user.draft_position_caret("{draft_anchor}")`
cursor after <user.draft_anchor>
`user.draft_position_caret("{draft_anchor}", 1)`
select <user.draft_anchor>
`user.draft_select("{draft_anchor}")`
select <user.draft_anchor> through <user.draft_anchor>
`user.draft_select("{draft_anchor_1}", "{draft_anchor_2}")`
clear <user.draft_anchor>
`user.draft_select("{draft_anchor}", "", 1)`
`key(backspace)`
clear <user.draft_anchor> through <user.draft_anchor>
`user.draft_select(draft_anchor_1, draft_anchor_2, 1)`
`key(backspace)`
<user.formatters> word <user.draft_anchor>
`user.draft_select("{draft_anchor}", "", 1)`
`user.formatters_reformat_selectior`
<user.formatters> <user.draft_anchor> through <user.draft_anchor>
`user.draft_select(draft_anchor_1, draft_anchor_2, 1)`
`user.formatters_reformat_selectior`

win draft window open

draft hide `user.draft_hide()`
draft submit `content = user.draft_get_text()`
```
user.draft_hide()
insert(content)
```

find and replace

hunt this `user.find("")`
hunt this <user.text>
`user.find(text)`
hunt all `user.find_everywhere("")`
hunt all <user.text>
`user.find_everywhere(text)`
hunt case
`user.find_toggle_match_by_case()`
hunt word
`user.find_toggle_match_by_word()`
hunt expression
`user.find_toggle_match_by_regex()`

hunt next `user.find_next()`
hunt previous `user.find_previous()`
replace this [<user.text>]
`user.replace(text or "")`
replace all
`user.replace_everywhere("")`
replace <user.text> all
`user.replace_everywhere(text)`
replace confirm that
`user.replace_confirm()`
replace confirm all
`user.replace_confirm_all()`
clear last <user.text> [over]
`user.select_previous_occurrence(te`
`sleep(100ms)`
`edit.delete()`
clear next <user.text> [over]
`user.select_next_occurrence(text)`
`sleep(100ms)`
`edit.delete()`
clear last clip
`user.select_previous_occurrence(cl`
`edit.delete()`
clear next clip
`user.select_next_occurrence(clip.t`
`sleep(100ms)`
`edit.delete()`
comment last <user.text> [over]
`user.select_previous_occurrence(te`
`sleep(100ms)`
`code.toggle_comment()`
comment last clip
`user.select_previous_occurrence(cl`
`sleep(100ms)`
`code.toggle_comment()`
comment next <user.text> [over]
`user.select_next_occurrence(text)`
`sleep(100ms)`
`code.toggle_comment()`
comment next clip
`user.select_next_occurrence(clip.t`
`sleep(100ms)`
`code.toggle_comment()`
go last <user.text> [over]
`user.select_previous_occurrence(te`
`sleep(100ms)` `edit.right()`
go last clip
`user.select_previous_occurrence(cl`
`sleep(100ms)` `edit.right()`
go next <user.text> [over]
`user.select_next_occurrence(text)`
`edit.right()`
go next clip
`user.select_next_occurrence(clip.t`
`edit.right()`
paste last <user.text> [over]
`user.select_previous_occurrence(te`
`sleep(100ms)` `edit.right()`
`edit.paste()`
paste next <user.text> [over]
`user.select_next_occurrence(text)`
`sleep(100ms)` `edit.right()`
`edit.paste()`
replace last <user.text> [over]
`user.select_previous_occurrence(te`
`sleep(100ms)` `edit.paste()`
replace next <user.text> [over]
`user.select_next_occurrence(text)`
`sleep(100ms)` `edit.paste()`
select last <user.text> [over]
`user.select_previous_occurrence(te`
select next <user.text> [over]
`user.select_next_occurrence(text)`
select last clip
`user.select_previous_occurrence(cl`
select next clip
`user.select_next_occurrence(clip.t`

generic editor

find it `edit.find()`
next one `edit.find_next()`
go word left `edit.word_left()`
go word right `edit.word_right()`
go left `edit.left()`
go right `edit.right()`
go up `edit.up()`
go down `edit.down()`
go line start `edit.line_start()`
go line end `edit.line_end()`
go way left `edit.line_start()`
`edit.line_start()`
go way right `edit.line_end()`
go way down `edit.file_end()`
go way up `edit.file_start()`
go page down `edit.page_down()`
go page up `edit.page_up()`
select line `edit.select_line()`
select all `edit.select_all()`
select left `edit.extend_left()`
select right `edit.extend_right()`
select up `edit.extend_line_up()`
select down
`edit.extend_line_down()`
select word `edit.select_word()`
select word left
`edit.extend_word_left()`
select word right
`edit.extend_word_right()`
select way left
`edit.extend_line_start()`
select way right
`edit.extend_line_end()`
select way up
`edit.extend_file_start()`
select way down
`edit.extend_file_end()`
indent [more] `edit.indent_more()`
(indent less | out dent)
`edit.indent_less()`
clear line `edit.delete_line()`
clear left `key(backspace)`
clear right `key(delete)`
clear up `edit.extend_line_up()`
`edit.delete()`
clear down `edit.extend_line_down()`
`edit.delete()`
clear word `edit.delete_word()`
clear word left
`edit.extend_word_left()`
`edit.delete()`
clear word right
`edit.extend_word_right()`
`edit.delete()`
clear way left
`edit.extend_line_start()`
`edit.delete()`
clear way right
`edit.extend_line_end()`
`edit.delete()`
clear way up
`edit.extend_file_start()`
`edit.delete()`
clear way down
`edit.extend_file_end()`
`edit.delete()`
clear all `edit.select_all()`
`edit.delete()`
copy all `edit.select_all()`
`edit.copy()`
copy word `edit.select_word()`
`edit.copy()`
copy word left
`edit.extend_word_left()`
`edit.copy()`
copy word right
`edit.extend_word_right()`
`edit.copy()`
copy line `edit.select_line()`
`edit.copy()`
cut all `edit.select_all()`
`edit.cut()`
cut word `edit.select_word()`
`edit.cut()`
cut word left
`edit.extend_word_left()`
`edit.cut()`
cut word right
`edit.extend_word_right()`
`edit.cut()`
cut line `edit.select_line()`
`edit.cut()`

homophones

phones <user.homophones_canonical>
`user.homophones_show(homophones_ca`
phones that
`user.homophones_show_selection()`
phones force <user.homophones_canonical>
`user.homophones_force_show(homophc`
phones force
`user.homophones_force_show_selecti`
phones hide `user.homophones_hide()`

homophones open

choose <number_small> `result = user.homophones_select(number_smal insert(result) user.homophones_hide()`
choose <user.formatters> <number_small> `result = user.homophones_select(number_smal insert(user.formatted_text(result, formatters)) user.homophones_hide()`

line commands

lend `edit.line_end()`
bend `edit.line_start()`
go <number>
`edit.jump_line(number)`
go <number> end
`edit.jump_line(number) edit.line_end()`
comment [line] <number>
`user.select_range(number, number) code.toggle_comment()`
comment <number> until <number>
`user.select_range(number_1, number_2) code.toggle_comment()`
clear [line] <number>
`edit.jump_line(number) user.select_range(number, number) edit.delete()`
clear <number> until <number>
`user.select_range(number_1, number_2) edit.delete()`
copy [line] <number>
`user.select_range(number, number) edit.copy()`
copy <number> until <number>
`user.select_range(number_1, number_2) edit.copy()`
cut [line] <number>
`user.select_range(number, number) edit.cut()`
cut [line] <number> until <number>
`user.select_range(number_1, number_2) edit.cut()`
(paste | replace) <number> until <number>
`user.select_range(number_1, number_2) edit.paste()`
(select | cell | sell) [line] <number>
`user.select_range(number, number)`
(select | cell | sell) <number> until <number>
`user.select_range(number_1, number_2)`
tab that `edit.indent_more()`
tab [line] <number>
`edit.jump_line(number) edit.indent_more()`
tab <number> until <number>
`user.select_range(number_1, number_2) edit.indent_more()`
retab that `edit.indent_less()`
retab [line] <number>
`user.select_range(number, number) edit.indent_less()`
retab <number> until <number>
`user.select_range(number_1, number_2) edit.indent_less()`
drag [line] down
`edit.line_swap_down()`
drag [line] up `edit.line_swap_up()`
drag up [line] <number>
`user.select_range(number, number) edit.line_swap_up()`
drag up <number> until <number>
`user.select_range(number_1, number_2) edit.line_swap_up()`
drag down [line] <number>
`user.select_range(number, number) edit.line_swap_down()`
drag down <number> until <number>
`user.select_range(number_1, number_2) edit.line_swap_down()`
clone (line|that) `edit.line_clone()`

numbers

<user.number_string>
`"{number_string}"`

symbols

question [mark] `"?"`
(downscore | underscore) `"_"`
double dash `"--"`
(bracket | brack | left bracket) `"{"`
(rbrack | are bracket | right bracket) `"}"`
triple quote `"'''"`
(dot dot | dotdot) `".."`
ellipses `"..."`
(comma and | spamma) `", "`
plus `"+"`
arrow `"->"`
dub arrow `"=>"`
new line `"\\n"`
carriage return `"\\r"`
line feed `"\\r\\n"`
empty dubstring `'""' key(left)`
empty escaped (dubstring|dub quotes) `'\\"\\"' key(left) key(left)`
empty string `"''" key(left)`
empty escaped string `"\\'\\'" key(left) key(left)`
(inside parens | args) `insert("()") key(left)`
inside (squares | list) `insert("[]") key(left)`
inside (bracket | braces) `insert("{}") key(left)`
inside percent `insert("%%") key(left)`
inside quotes `insert('""') key(left)`
angle that `text = edit.selected_text() user.paste("<{text}>")`
(bracket | brace) that `text = edit.selected_text() user.paste("{{{text}}}")`
(parens | args) that `text = edit.selected_text() user.paste("({text})")`
percent that `text = edit.selected_text() user.paste("%{text}%")`
quote that `text = edit.selected_text() user.paste('"{text}"')`

text navigation

navigate [{user.arrow_key}] [{user.navigation_action}] [{user.navigation_target_name}] [{user.before_or_after}] [<user.ordinals>]

<user.navigation_target>
```
user.navigation(navigation_action or "GO", arrow_key or "RIGHT", navigation_target_name or "DEFAULT", before_or_after or "DEFAULT", navigation_target, ordinals or 1)
```
word neck [<number_small>]
```
user.navigation_by_name("SELECT", "RIGHT", "DEFAULT", "word", number_small or 1)
```
word pre [<number_small>]
```
user.navigation_by_name("SELECT", "LEFT", "DEFAULT", "word", number_small or 1)
```
small word neck [<number_small>]
```
user.navigation_by_name("SELECT", "RIGHT", "DEFAULT", "small", number_small or 1)
```
small word pre [<number_small>]
```
user.navigation_by_name("SELECT", "LEFT", "DEFAULT", "small", number_small or 1)
```
big word neck [<number_small>]
```
user.navigation_by_name("SELECT", "RIGHT", "DEFAULT", "big", number_small or 1)
```
big word pre [<number_small>]
```
user.navigation_by_name("SELECT", "LEFT", "DEFAULT", "big", number_small or 1)
```