Talon Cheat Sheet

Generated on: 2024-07-03 23:45:39.504888

LETTERS

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

FORMATTERS

Input	Result
all cap	EXAMPLE TEXT
all down	example text
camel	exampleText
dotted	example.text
dub string	"example text"
dunder	__example__text
hammer	ExampleText
kebab	example-text
packed	example::text
padded	[space]example text[space]
slasher	/example/text
smash	exampletext
snake	example_text
string	'example text'
sentence	Example text
title	Example Text

PUNCTUATIONS

Input	Result
,	,
L paren	(
R paren	)
`	`
ampersand	&
and sign	&
asterisk	*
at sign	@
back tick	`
colon	:
coma	,
comma	,
dollar sign	$
exclamation mark	!
exclamation point	!
forward slash	/
full stop	.
hash sign	#
hyphen	-
left paren	(
number sign	#
percent sign	%
pound sign	£
question mark	?
right paren	)
semicolon	;

SYMBOL_KEYS

Input	Result
,	,
L paren	(
L square	[
R angle	>
R paren	)
R square	]
`	`
amper	&
ampersand	&
and sign	&
angle	<
apostrophe	'
asterisk	*
at sign	@
back tick	`
backslash	\
bang	!
brace	{
brack	[
bracket	[
caret	^
colon	:
coma	,
comma	,
curly bracket	{
dash	-
dollar	$
dollar sign	$
dot	.
double quote	"
down score	_
dub quote	"
equals	=
exclamation mark	!
exclamation point	!
forward slash	/
full stop	.
grave	`
greater than	>
hash	#
hash sign	#
hyphen	-
left angle	<
left brace	{
left bracket	[
left curly bracket	{
left paren	(
left square	[
less than	<
minus	-
number sign	#
paren	(
percent	%

percent sign	%
pipe	|
plus	+
point	.
pound	£
pound sign	£
question	?
question mark	?
quote	'
r brace	}
r brack	]
r bracket	]
r curly bracket	}
rangle	>
right angle	>
right brace	}
right bracket	]
right curly bracket	}
right paren	)
right square	]
semicolon	;
slash	/
square	[
star	*
tilde	~
underscore	_

CURSORLESS

Input	Result
{user.cursorless_bring_move_action} <user.cursorless_bring_move_targets>	user.private_cursorless_bring_move(cursorless_bring_move_action, cursorless_bring_move_targets)
{user.cursorless_call_action} <user.cursorless_target> on <user.cursorless_target>	user.private_cursorless_call(cursorless_target_1, cursorless_target_2)
{user.cursorless_hide_scope_visualizer}	user.private_cursorless_hide_scope_visualizer()
{user.cursorless_homophone} settings	user.private_cursorless_show_settings_in_ide()
{user.cursorless_insert_snippet_action} {user.cursorless_insertion_snippet_single_phrase} <user.text> [{user.cursorless_phrase_terminator}]	user.private_cursorless_insert_snippet_with_phrase(cursorless_insertion_snippet_single_phrase, text)
{user.cursorless_insert_snippet_action} <user.cursorless_insertion_snippet>	user.private_cursorless_insert_snippet(cursorless_insertion_snippet)
{user.cursorless_paste_action} <user.cursorless_destination>	user.private_cursorless_paste(cursorless_destination)
{user.cursorless_reformat_action} <user.formatters> at <user.cursorless_target>	user.private_cursorless_reformat(cursorless_target, formatters)
{user.cursorless_show_scope_visualizer} <user.cursorless_scope_type> [{user.cursorless_visualization_type}]	user.private_cursorless_show_scope_visualizer(cursorless_scope_type, cursorless_visualization_type or "content")
{user.cursorless_swap_action} <user.cursorless_swap_targets>	user.private_cursorless_swap(cursorless_swap_targets)
{user.cursorless_wrapper_snippet} {user.cursorless_wrap_action} <user.cursorless_target>	user.private_cursorless_wrap_with_snippet(cursorless_wrap_action, cursorless_target, cursorless_wrapper_snippet)
<user.cursorless_action_or_ide_command> <user.cursorless_target>	user.private_cursorless_action_or_ide_command(cursorless_action_or_ide_command, cursorless_target)
<user.cursorless_wrapper_paired_delimiter> {user.cursorless_wrap_action} <user.cursorless_target>	user.private_cursorless_wrap_with_paired_delimiter(cursorless_wrap_action, cursorless_target, cursorless_wrapper_paired_delimiter)
bar {user.cursorless_homophone}	user.private_cursorless_show_sidebar()

CURSORLESS GLOBAL

Input	Result
{user.cursorless_homophone} (instructions | docks | help) | help {user.cursorless_homophone}	user.private_cursorless_open_instructions()
{user.cursorless_homophone} (reference | ref | cheatsheet | cheat sheet)	user.private_cursorless_cheat_sheet_show_html()

DIRECT CLICKING

Input	Result
<user.rango_target>	user.rango_command_with_target("directClickElement", rango_target)

RANGO

Input	Result
(go tab | slot) <user.rango_tab_marker>	user.rango_command_with_target("activateTab", rango_tab_marker)
address in title off	user.rango_command_without_target("disableUrlInTitle")
address in title on	user.rango_command_without_target("enableUrlInTitle")
blank <user.rango_target>	user.rango_command_with_target("openInNewTab", rango_target)
bottom <user.rango_target>	user.rango_command_with_target("scrollElementToBottom", rango_target)
center <user.rango_target>	user.rango_command_with_target("scrollElementToCenter", rango_target)
change <user.rango_target>	user.rango_clear_input(rango_target)
click <user.rango_target>	user.rango_command_with_target("clickElement", rango_target)
click mark <user.word>	user.rango_run_action_on_reference("clickElement", word)
copy [link] <user.rango_target>	user.rango_command_with_target("copyLink", rango_target)
copy mark <user.rango_target>	user.rango_command_with_target("copyMarkdownLink", rango_target)
copy mark address	user.rango_command_without_target("copyCurrentTabMarkdownUrl")
copy page {user.rango_page_location_property}	user.rango_command_without_target("copyLocationProperty", rango_page_location_property)
copy text <user.rango_target>	user.rango_command_with_target("copyElementTextContent", rango_target)
crown <user.rango_target>	user.rango_command_with_target("scrollElementToTop", rango_target)
custom hints reset	user.rango_command_without_target("resetCustomSelectors")
custom hints save	user.rango_command_without_target("confirmSelectorsCustomization")
dismiss	user.rango_command_without_target("unhoverAll")
down again	user.rango_command_without_target("scrollDownAtElement")
downer	user.rango_command_without_target("scrollDownPage")
downer <number>	user.rango_command_without_target("scrollDownPage", number)
downer <user.rango_target>	user.rango_command_with_target("scrollDownAtElement", rango_target)
downer all	user.rango_command_without_target("scrollDownPage", 9999)
downer left	user.rango_command_without_target("scrollDownLeftAside")
downer left all	user.rango_command_without_target("scrollDownLeftAside", 9999)
downer right	user.rango_command_without_target("scrollDownRightAside")
downer right all	user.rango_command_without_target("scrollDownRightAside", 9999)
enter <user.text> to <user.rango_target>	user.rango_insert_text_to_input(text, rango_target, 1)
exclude <user.rango_target>	user.rango_command_with_target("excludeExtraSelectors", rango_target)
exclude all	user.rango_command_without_target("excludeAllHints")
flick <user.rango_target>	user.rango_command_with_target("focusElement", rango_target) key(enter)
focus <user.rango_target>	user.rango_command_with_target("focusElement", rango_target)
focus mark <user.word>	user.rango_run_action_on_reference("focusElement", word)
go input	user.rango_command_without_target("focusFirstInput")
go root	user.rango_command_without_target("navigateToPageRoot")
hint bigger	user.rango_command_without_target("increaseHintSize")
hint exclude singles	user.rango_command_without_target("excludeSingleLetterHints")
hint extra	user.rango_command_without_target("displayExtraHints")
hint include singles	user.rango_command_without_target("includeSingleLetterHints")
hint less	user.rango_command_without_target("displayLessHints")
hint more	user.rango_command_without_target("displayExcludedHints")
hint smaller	user.rango_command_without_target("decreaseHintSize")
hints (toggle | switch)	user.rango_command_without_target("toggleHints")
hints off [{user.rango_hints_toggle_levels}]	user.rango_command_without_target("disableHints", rango_hints_toggle_levels or "global")
hints on [{user.rango_hints_toggle_levels}]	user.rango_command_without_target("enableHints", rango_hints_toggle_levels or "global")
hints refresh	user.rango_command_without_target("refreshHints")
hints reset {user.rango_hints_toggle_levels}	user.rango_command_without_target("resetToggleLevel", rango_hints_toggle_levels)

hover <user.rango_target>	user.rango_command_with_target("hoverElement", rango_target)
hover mark <user.word>	user.rango_run_action_on_reference("hoverElement", word)
include <user.rango_target>	user.rango_command_with_target("includeExtraSelectors", rango_target)
insert <user.text> to <user.rango_target>	user.rango_insert_text_to_input(text, rango_target, 0)
keyboard (toggle | switch)	user.rango_command_without_target("toggleKeyboardClicking")
left again	user.rango_command_without_target("scrollLeftAtElement")
mark <user.rango_target> as <user.word>	user.rango_command_with_target("saveReference", rango_target, word)
mark clear <user.word>	user.rango_command_without_target("removeReference", word)
mark show	user.rango_command_without_target("showReferences")
markers (toggle | switch)	user.rango_command_without_target("toggleTabMarkers")
page last	user.rango_command_without_target("navigateToPreviousPage")
page next	user.rango_command_without_target("navigateToNextPage")
paste to <user.rango_target>	user.rango_insert_text_to_input(clip.text(), rango_target, 0)
post <user.rango_target>	user.rango_command_with_target("setSelectionAfter", rango_target)
pre <user.rango_target>	user.rango_command_with_target("setSelectionBefore", rango_target)
rango direct	user.rango_force_direct_clicking()
rango explicit	user.rango_force_explicit_clicking()
rango open {user.rango_page}	user.rango_command_without_target("openPageInNewTab", rango_page)
rango settings	user.rango_command_without_target("openSettingsPage")
right again	user.rango_command_without_target("scrollRightAtElement")
scroll left	user.rango_command_without_target("scrollLeftPage")
scroll left <user.rango_target>	user.rango_command_with_target("scrollLeftAtElement", rango_target)
scroll left all	user.rango_command_without_target("scrollLeftPage", 9999)
scroll right	user.rango_command_without_target("scrollRightPage")
scroll right <user.rango_target>	user.rango_command_with_target("scrollRightAtElement", rango_target)
scroll right all	user.rango_command_without_target("scrollRightPage", 9999)
scroll save <user.word>	user.rango_command_without_target("storeScrollPosition", word)
scroll to <user.word>	user.rango_command_without_target("scrollToPosition", word)
show <user.rango_target>	user.rango_command_with_target("showLink", rango_target)
some less	user.rango_command_without_target("includeOrExcludeLessSelectors")
some more	user.rango_command_without_target("includeOrExcludeMoreSelectors")
stash <user.rango_target>	user.rango_command_with_target("openInBackgroundTab", rango_target)
tab ahead	user.rango_command_without_target("cycleTabsByText", 1)
tab back	user.rango_command_without_target("focusPreviousTab")
tab behind	user.rango_command_without_target("cycleTabsByText", -1)
tab clone	user.rango_command_without_target("cloneCurrentTab")
tab close final [<number_small>]	user.rango_command_without_target("closeTabsRightEndInWindow", number_small or 1)
tab close first [<number_small>]	user.rango_command_without_target("closeTabsLeftEndInWindow", number_small or 1)
tab close left	user.rango_command_without_target("closeTabsToTheLeftInWindow")
tab close next [<number_small>]	user.rango_command_without_target("closeNextTabsInWindow", number_small or 1)
tab close other	user.rango_command_without_target("closeOtherTabsInWindow")
tab close previous [<number_small>]	user.rango_command_without_target("closePreviousTabsInWindow", number_small or 1)
tab close right	user.rango_command_without_target("closeTabsToTheRightInWindow")
tab hunt <user.text>	user.rango_command_without_target("focusTabByText", text)
tab marker refresh	user.rango_command_without_target("refreshTabMarkers")
tab split	user.rango_command_without_target("moveCurrentTabToNewWindow")
tiny down	user.rango_command_without_target("scrollDownPage", 0.2)
tiny down <user.rango_target>	user.rango_command_with_target("scrollDownAtElement", rango_target, 0.2)
tiny left	user.rango_command_without_target("scrollLeftPage", 0.2)
tiny left <user.rango_target>	user.rango_command_with_target("scrollLeftAtElement", rango_target, 0.1)
tiny right	user.rango_command_without_target("scrollRightPage", 0.2)

tiny right <user.rango_target>	user.rango_command_with_target("scrollRightAtElement", rango_target, 0.1)
tiny up	user.rango_command_without_target("scrollUpPage", 0.2)
tiny up <user.rango_target>	user.rango_command_with_target("scrollUpAtElement", rango_target, 0.2)
toggle show	user.rango_command_without_target("displayTogglesStatus")
up again	user.rango_command_without_target("scrollUpAtElement")
upper	user.rango_command_without_target("scrollUpPage")
upper <number>	user.rango_command_without_target("scrollUpPage", number)
upper <user.rango_target>	user.rango_command_with_target("scrollUpAtElement", rango_target)
upper all	user.rango_command_without_target("scrollUpPage", 9999)
upper left	user.rango_command_without_target("scrollUpLeftAside")
upper left all	user.rango_command_without_target("scrollUpLeftAside", 9999)
upper right	user.rango_command_without_target("scrollUpRightAside")
upper right all	user.rango_command_without_target("scrollUpRightAside", 9999)
visit {user.website}	user.rango_command_without_target("focusOrCreateTabByUrl", website)

TALON HELPERS

Input	Result
click rango mark <user.word>	"user.rango_run_action_on_reference(\"clickElement\", \"{word}\")"
focus rango mark <user.word>	"user.rango_run_action_on_reference(\"focusElement\", \"{word}\")"
hover rango mark <user.word>	"user.rango_run_action_on_reference(\"hoverElement\", \"{word}\")"

BETA-CURSORLESS

Input	Result
model blend <user.cursorless_target> to <user.cursorless_target>	target_text = user.cursorless_get_text_list(cursorless_target_1) destination_text = user.cursorless_get_text(cursorless_target_2) default_destination = user.cursorless_create_destination(cursorless_target_2) result = user.gpt_blend_list(target_text, destination_text) user.cursorless_insert(default_destination, result)
model blend clip to <user.cursorless_target>	clipboard_text = clip.text() destination_text = user.cursorless_get_text(cursorless_target) default_destination = user.cursorless_create_destination(cursorless_target) result = user.gpt_blend(clipboard_text, destination_text) user.cursorless_insert(default_destination, result)

BETA-GPT

Input	Result
model blend clip	clipboard_text = clip.text() destination_text = edit.selected_text() result = user.gpt_blend(clipboard_text, destination_text) user.gpt_insert_response(result, "")
model find <user.text>	user.gpt_find_talon_commands(user.text)

CONTINUE

Input	Result
bar tin you	user.vscode("continue.continueGUIView.focus")
tin you (accept | yes)	user.vscode("continue.acceptDiff")
tin you add <user.cursorless_target>	user.cursorless_ide_command("continue.focusContinueInput", cursorless_target)
tin you cancel	key("escape")
tin you comment <user.cursorless_target>	user.cursorless_ide_command("continue.writeCommentsForCode", cursorless_target)
tin you debug terminal	user.vscode("continue.debugTerminal")
tin you dock <user.cursorless_target>	user.cursorless_ide_command("continue.writeDocstringForCode", cursorless_target)
tin you edit <user.cursorless_target>	user.cursorless_ide_command("continue.quickEdit", cursorless_target)
tin you file select	user.vscode("continue.selectFilesAsContext")
tin you fix code <user.cursorless_target>	user.cursorless_ide_command("continue.fixCode", cursorless_target)
tin you fix grammar <user.cursorless_target>	user.cursorless_ide_command("continue.fixGrammar", cursorless_target)
tin you history	user.vscode("continue.viewHistory")
tin you new	user.vscode("continue.newSession")
tin you optimize <user.cursorless_target>	user.cursorless_ide_command("continue.optimizeCode", cursorless_target)
tin you reject	user.vscode("continue.rejectDiff")
tin you select files	user.vscode("continue.selectFilesAsContext")
tin you share	user.vscode("continue.shareSession")
tin you toggle fullscreen	user.vscode("continue.toggleFullScreen")

GPT-CURSORLESS

Input	Result
model <user.modelPrompt> <user.cursorless_target> [<user.cursorless_destination>]	text = user.cursorless_get_text_list(cursorless_target) result = user.gpt_apply_prompt(user.modelPrompt, text) default_destination = user.cursorless_create_destination(cursorless_target) user.cursorless_insert(cursorless_destination or default_destination, result)

GPT-SHELL

Input	Result
copy model output	user.copy_model_confirmation_gui()
deny model output	user.close_model_confirmation_gui()
paste model output	user.paste_model_confirmation_gui()
model (sequel | sql) <user.text>	result = user.gpt_generate_sql(user.text) user.add_to_confirmation_gui(result)
model shell <user.text>	result = user.gpt_generate_shell(user.text) user.add_to_confirmation_gui(result)

GPT

Input	Result
model [nope] that was <user.text>	result = user.gpt_reformat_last(text) user.paste(result)
model <user.modelPrompt> [{user.modelSource}] [{user.modelDestination}]	text = user.gpt_get_source_text(modelSource or "") result = user.gpt_apply_prompt(modelPrompt, text) user.gpt_insert_response(result, modelDestination or "")
model apply [from] clip	prompt = clip.text() text = edit.selected_text() result = user.gpt_apply_prompt(prompt, text) user.paste(result)
model help	user.gpt_help()
model replace as in <user.text>	preprompt = "Your job is to Fix the following text. Do not change the original structure of the text. There is a word in the following content that has been mispronounced. Please replace the incorrect homophone based on this description:" prompt = user.text postprompt = ". The content is" txt = edit.selected_text() result = user.gpt_apply_prompt("{preprompt} {prompt} {postprompt}", txt) user.paste(result)
model replace with <user.text>	preprompt = "Your job is to Fix incorrect homophones that has been mispronounced. The incorrect homophones is/are" prompt = edit.selected_text() postprompt = "Please return the correct homophone based on the following description " result = user.gpt_apply_prompt("{preprompt} {prompt} {postprompt}", user.text) user.paste(result)
model take response	user.gpt_select_last()

AI-IMAGES

Input	Result
image describe [{user.descriptionPrompt}]	result = user.image_describe_clipboard(descriptionPrompt or "") user.paste(result)
image describe screen [{user.descriptionPrompt}]	user.screenshot_clipboard() result = user.image_describe_clipboard(descriptionPrompt or "") user.paste(result)
image describe window [{user.descriptionPrompt}]	user.screenshot_window_clipboard() result = user.image_describe_clipboard(descriptionPrompt or "") user.paste(result)
image generate <user.text>	user.image_generate(text)

GPT-TTS

Input	Result
echo <user.modelPrompt> [this]	text = edit.selected_text() result = user.gpt_apply_prompt(user.modelPrompt, text) user.tts(result)
echo ask <user.text>	result = user.gpt_answer_question(text) user.tts(result)

IMAGE-TTS

Input	Result
echo (describe | image) (image | describe)	result = user.image_describe_clipboard("") user.tts(result)
echo describe [{user.descriptionPrompt}]	result = user.image_describe_clipboard(descriptionPrompt or "") user.tts(result)
echo describe screen [{user.descriptionPrompt}]	user.screenshot_clipboard() result = user.image_describe_clipboard(descriptionPrompt or "") user.tts(result)
echo describe window [{user.descriptionPrompt}]	user.screenshot_window_clipboard() result = user.image_describe_clipboard(descriptionPrompt or "") user.tts(result)

CODEIUM

Input	Result
pilot (previous | last)	user.vscode("editor.action.inlineSuggest.showPrevious")
pilot cancel	user.vscode("editor.action.inlineSuggest.hide")
pilot chat [<user.prose>]	user.vscode("codeium.openChatView") sleep(2) user.paste(user.prose or "")
pilot debug	user.vscode("codeium.explainProblem")
pilot editor	user.vscode("codeium.openChatInPane")
pilot explain	user.vscode("codeium.explainCodeBlock")
pilot make [<user.prose>]	user.vscode("codeium.openCodeiumCommand") sleep(0.7) user.paste(user.prose or "")
pilot next	user.vscode("editor.action.inlineSuggest.showNext")
pilot nope	user.vscode("editor.action.inlineSuggest.undo")
pilot refactor	user.vscode("codeium.refactorCodeBlock")
pilot search	user.vscode("codeium.openSearchView")
pilot submit	key(ctrl-shift-enter)
pilot toggle	user.vscode("codeium.toggleEnabledForCurrentLanguage")
pilot yes	user.vscode("editor.action.inlineSuggest.commit")

COPILOT

Input	Result
pilot {user.copilot_slash_command} <user.cursorless_target> [to <user.prose>]	user.cursorless_command("setSelection", cursorless_target) user.copilot_inline_chat(copilot_slash_command or "", prose or "")
pilot (previous | last)	user.vscode("editor.action.inlineSuggest.showPrevious")
pilot block last	user.vscode("workbench.action.chat.previousCodeBlock")
pilot block next	user.vscode("workbench.action.chat.nextCodeBlock")
pilot bring <user.ordinal_or_last>	user.copilot_bring_code_block(ordinal_or_last)
pilot bring <user.ordinal_or_last> {user.makeshift_destination} <user.cursorless_target>	user.cursorless_command(makeshift_destination, cursorless_target) user.copilot_bring_code_block(ordinal_or_last)
pilot cancel	user.vscode("editor.action.inlineSuggest.hide")
pilot chat [<user.prose>]	user.copilot_chat(prose or "")
pilot copy <user.ordinal_or_last>	user.copilot_focus_code_block(ordinal_or_last) edit.copy()
pilot jest	user.vscode("editor.action.inlineSuggest.trigger")
pilot make [<user.prose>]	user.copilot_inline_chat("", prose or "")
pilot new file <user.ordinal_or_last>	user.copilot_focus_code_block(ordinal_or_last) user.vscode("workbench.action.chat.insertIntoNewFile")
pilot next	user.vscode("editor.action.inlineSuggest.showNext")
pilot nope	user.vscode("editor.action.inlineSuggest.undo")
pilot yes	user.vscode("editor.action.inlineSuggest.commit")
pilot yes word	user.vscode("editor.action.inlineSuggest.acceptNextWord")

CHEATSHEET

Input	Result
prince cheat sheet	user.cheatsheet()
prince cheatsheet	user.cheatsheet()
print cheat sheet	user.cheatsheet()
print cheatsheet	user.cheatsheet()

APPLE NOTES

Input	Result
apply body	key(shift-cmd-b)
apply bullet	key(shift-cmd-7)
apply checklist	key(shift-cmd-l)
apply dash	key(shift-cmd-8)
apply heading	key(shift-cmd-h)
apply mono	key(shift-cmd-m)
apply number	key(shift-cmd-9)
apply subheading	key(shift-cmd-j)
apply title	key(shift-cmd-t)
attach file	key(shift-cmd-a)
create link	key(cmd-k)
decrease font	key(cmd--)
drag [line] down	key('ctrl-cmd-down')
drag [line] up	key('ctrl-cmd-up')
duplicate note	key(cmd-d)
find all	key(alt-cmd-f)
gallery view	key(cmd-2)
increase font	key(cmd-+)
insert table	key(alt-cmd-t)
line break	key(ctrl-enter)
list view	key(cmd-1)
mark	key(shift-cmd-u)
new folder	key(shift-cmd-n)
new note	key(cmd-n)
print note	key(cmd-p)
show main	key(cmd-0)
toggle attachments	key(cmd-3)
toggle folders	key(alt-cmd-s)

APPLE TERMINAL

Input	Result
resume	insert("fg") key(enter)
suspend	key(ctrl-z)

CHROME

Input	Result
profile switch	user.chrome_mod("shift-m")
tab search	user.chrome_mod("shift-a")
tab search <user.text>	user.chrome_mod("shift-a") sleep(200ms) insert("{text}") key(down)

DISCORD

Input	Result
{user.discord_destination} [<user.text>]	user.discord_quick_switcher(user.discord_destination, user.text or "")
(mute | unmute) and sleep	user.discord_mute() speech.disable()
[channel] mentions last	user.discord_mentions_last()
[channel] mentions next	user.discord_mentions_next()
[toggle] (deafen | undeafen)	user.discord_deafen()
[toggle] (mute | unmute)	user.discord_mute()
answer call	user.discord_answer_call()
current call	user.discord_go_current_call()
decline call	user.discord_decline_call()
mark inbox channel read	user.discord_mark_inbox_read()
oldest unread	user.discord_oldest_unread()
pick (jif | gif | gift)	user.discord_gif_picker()
pick emoji	user.discord_emoji_picker()
pick sticker	user.discord_sticker_picker()
switcher	user.discord_quick_switcher("", "")
toggle (dee ems | dims)	user.discord_toggle_dms()
toggle (members | member list)	user.discord_toggle_members()
toggle inbox	user.discord_toggle_inbox()
toggle pins	user.discord_toggle_pins()

FINDER

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

FIREFOX

Input	Result
(sidebar | panel) bookmarks	user.firefox_bookmarks_sidebar()
(sidebar | panel) history	user.firefox_history_sidebar()
tab search	browser.focus_address() insert("% ")
tab search <user.text>	browser.focus_address() insert("% {text}") key(down)

GIT

Input	Result
git {user.git_command} [<user.git_arguments>]	args = git_arguments or "" "git {git_command}{args} "
git add clipboard	insert("git add ") edit.paste() key(enter)
git add highlighted	edit.copy() insert("git add ") edit.paste() key(enter)
git add patch	"git add --patch\n"
git clone clipboard	insert("git clone ") edit.paste() key(enter)
git commit [<user.git_arguments>] message [<user.prose>]	args = git_arguments or "" message = prose or "" user.insert_between('git commit{args} --message "{message}', "")
git commit highlighted	edit.copy() insert("git add ") edit.paste() insert("\ngit commit\n")
git diff (cached | cashed)	"git diff --cached\n"
git diff clipboard	insert("git diff ") edit.paste() key(enter)
git diff highlighted	edit.copy() insert("git diff ") edit.paste() key(enter)
git diff	"git diff\n"
git show head	"git show HEAD\n"
git stash [push] [<user.git_arguments>] message [<user.prose>]	args = git_arguments or "" message = prose or "" user.insert_between('git stash push{args} --message "{message}', "")
git status	"git status\n"

GIT ADD PATCH

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

GITHUB WEB

Input	Result
((branch | tag) switch | switch (branch | tag))	key(w)
(comment preview | preview comment)	key(ctrl-shift-p)
(comment submit | submit comment)	key(ctrl-enter)
(file find | find file)	key(t)
(filter by | edit) assignee	key(a)
(filter by | edit) labels	key(l)
(filter by | edit) milestones	key(m)
(form close | close form)	key(escape)
(issue create | create [an] issue)	key(c)
(issue create | create issue)	key(c)
(issue open | open issue)	key(o)
(keyboard shortcuts show | show keyboard shortcuts)	key(?)
(selection move down | move selection down)	key(j)
(selection move up | move selection up)	key(k)
(selection open | open selection)	key(o)
(selection toggle | toggle selection)	key(x)
(show | hide) annotations	key(a)
(show | hide) comments	key(i)
(thread mute | mute thread)	key(shift-m)
(url expand | expand url)	key(y)
[web] editor open	key(.)
assignee set	key(a)
blame view open	key(b)
focus search	key(s)
fullscreen toggle	key(shift-f)
git hub full screen	key(ctrl-shift-l)
go to actions	insert("ga")
go to code	insert("gc")
go to dashboard	insert("gd")
go to discussions	insert("gg")
go to issues	insert("gi")
go to notifications	insert("gn")
go to projects	insert("gb")
go to pull requests	insert("gp")
go to wiki	insert("gw")
go to workflow	insert("gf")
jump to line	key(l)
label set	key(l)
mark as read	key(y)
milestone set	key(m)
other parent commit	key(o)
parent commit	key(p)
reply	key(r)
reviewer request	key(q)
search (issues | [pull] requests)	key(/)
timestamps toggle	key(shift-t)

I3WM

Input	Result
(full screen | scuba)	user.i3wm_fullscreen()
(launch shell | koopa)	user.i3wm_shell()
(port flip | flipper)	user.i3wm_switch_to_workspace("back_and_forth")
(shuffle | move (win | window) [to] last port)	user.i3wm_move_to_workspace("back_and_forth")
(shuffle | move (win | window) [to] port) <number_small>	user.i3wm_move_to_workspace(number_small)
(shuffle | move (win | window) down)	user.i3wm_move("down")
(shuffle | move (win | window) left)	user.i3wm_move("left")
(shuffle | move (win | window) right)	user.i3wm_move("right")
(shuffle | move (win | window) up)	user.i3wm_move("up")
(shuffle | move) flipper	user.i3wm_move_to_workspace("back_and_forth")
(win | window) default	user.i3wm_layout()
(win | window) down	user.i3wm_focus("down")
(win | window) horizontal	user.i3wm_split("h")
(win | window) kill	app.window_close()
(win | window) left	user.i3wm_focus("left")
(win | window) right	user.i3wm_focus("right")
(win | window) stacking	user.i3wm_layout("stacking")
(win | window) tabbed	user.i3wm_layout("tabbed")
(win | window) up	user.i3wm_focus("up")
(win | window) vertical	user.i3wm_split("v")
[(show | hide)] scratch	user.i3wm_show_scratchpad()
center window	user.i3wm_move_position("center")
focus child	user.i3wm_focus("child")
focus floating	user.i3wm_focus("mode_toggle")
focus parent	user.i3wm_focus("parent")
grow window	user.i3wm_mode("resize") key(right:10) key(down:10)
horizontal (shell | terminal)	user.i3wm_split("h") user.i3wm_shell()
launch	user.i3wm_launch()
launch <user.text>	user.i3wm_launch() sleep(100ms) insert("{text}")
lock screen	user.i3wm_lock()
make scratch	user.i3wm_move("scratchpad")
murder	user.deprecate_command("2023-02-04", "murder", "win kill") app.window_close()
new scratch (shell | window)	user.i3wm_shell() sleep(200ms) user.i3wm_move("scratchpad") user.i3wm_show_scratchpad()
next scratch	user.i3wm_show_scratchpad() user.i3wm_show_scratchpad()
port <number_small>	user.i3wm_switch_to_workspace(number_small)
port left	user.i3wm_switch_to_workspace("prev")
port right	user.i3wm_switch_to_workspace("next")
reload i three config	user.i3wm_reload()
resize mode	user.i3wm_mode("resize")
restart i three	user.i3wm_restart()
shrink window	user.i3wm_mode("resize") key(left:10) key(up:10)
toggle floating	user.i3wm_float()
vertical (shell | terminal)	user.i3wm_split("v") user.i3wm_shell()

JETBRAINS

Input	Result
(action | please)	user.idea("action GotoAction")
(action | please) <user.text>	user.idea("action GotoAction") insert(text)
(done | finish)	user.idea("action EditorCompleteStatement")
(go declaration | follow)	user.idea("action GotoDeclaration")
(grow | shrink) window down	user.idea("action ResizeToolWindowDown")
(grow | shrink) window left	user.idea("action ResizeToolWindowLeft")
(grow | shrink) window right	user.idea("action ResizeToolWindowRight")
(grow | shrink) window up	user.idea("action ResizeToolWindowUp")
(pop deaf | toggle definition)	user.idea("action QuickImplementations")
(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")
(toggle | pop) (doc | documentation)	user.idea("action QuickJavaDoc")
blacken	user.idea("action BLACKReformatCode")
change (recording | recordings)	user.idea("action EditMacros")
change scheme	user.idea("action QuickChangeScheme")
clear last <user.text> [over]	user.idea("find prev {text}, action EditorBackSpace")
clear next <user.text> [over]	user.idea("find next {text}, action EditorBackSpace")
clear task	user.idea("action tasks.close")
clippings	user.idea("action PasteMultiple")
clone <number>	user.line_clone(number)
collapse <number> until <number>	user.select_range(number_1, number_2) user.idea("action CollapseRegion")
collapse all	user.idea("action CollapseAllRegions")
collapse deep	user.idea("action CollapseRegionRecursively")
comment last <user.text> [over]	user.idea("find prev {text}, action CommentByLineComment")
comment next <user.text> [over]	user.idea("find next {text}, action CommentByLineComment")
complete	user.idea("action CodeCompletion")
configure servers	user.idea("action tasks.configure.servers")
continue	user.idea("action Resume")
copilot chat	user.chrome_mod("shift-c")
copy path	user.idea("action CopyPaths")
copy pretty	user.idea("action CopyAsRichText")
copy reference	user.idea("action CopyReference")
create (template | snippet)	user.idea("action SaveAsTemplate")
create file	user.idea("action NewElement")
create file <user.text> [over]	user.idea("action NewElement") sleep(500ms) insert(text)
create sibling	user.idea("action NewElementSamePlace")
create sibling <user.text> [over]	user.idea("action NewElementSamePlace") sleep(500ms) insert(text)
debug test	user.idea("action DebugClass")
expand <number> until <number>	user.select_range(number_1, number_2) user.idea("action ExpandRegion")
expand all	user.idea("action ExpandAllRegions")
expand deep	user.idea("action ExpandRegionRecursively")
extract constant	user.idea("action IntroduceConstant")
extract field	user.idea("action IntroduceField")
extract interface	user.idea("action ExtractInterface")

extract method	user.idea("action ExtractMethod")
extract parameter	user.idea("action IntroduceParameter")
extract variable	user.idea("action IntroduceVariable")
find (everywhere | all)	user.idea("action SearchEverywhere")
find (everywhere | all) <user.text> [over]	user.idea("action SearchEverywhere") sleep(500ms) insert(text)
fix (format | formatting)	user.idea("action ReformatCode")
fix imports	user.idea("action OptimizeImports")
fix last (error | air)	user.idea("action GotoPreviousError") user.idea("action ShowIntentionActions")
fix next (error | air)	user.idea("action GotoNextError") user.idea("action ShowIntentionActions")
git (annotate | blame)	user.idea("action Annotate")
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 browse	user.idea("action Github.Open.In.Browser")
git commit	user.idea("action CheckinProject")
git log	user.idea("action Vcs.ShowTabbedFileHistory")
git menu	user.idea("action Vcs.QuickListPopupAction")
git pull	user.idea("action Vcs.UpdateProject")
git push	user.idea("action CheckinProject")
go back	user.idea("action Back")
go breakpoints	user.idea("action ViewBreakpoints")
go browser task	user.idea("action tasks.open.in.browser")
go camel left	user.camel_left()
go camel right	user.camel_right()
go forward	user.idea("action Forward")
go implementation	user.idea("action GotoImplementation")
go last (error | air)	user.idea("action GotoPreviousError")
go last (method | function)	user.idea("action MethodUp")
go last <user.text> [over]	user.idea("find prev {text}, action EditorRight")
go last mark	user.idea("action GotoPreviousBookmark")
go mark	user.idea("action ShowBookmarks")
go mark <number>	user.idea("action GotoBookmark{number}")
go next (error | air)	user.idea("action GotoNextError")
go next (method | function)	user.idea("action MethodDown")
go next <user.text> [over]	user.idea("find next {text}, action EditorRight")
go next mark	user.idea("action GotoNextBookmark")
go task	user.idea("action tasks.goto")
go test	user.idea("action GotoTest")
go type	user.idea("action GotoTypeDeclaration")
go usage	user.idea("action FindUsages")
grab <number>	user.idea_grab(number)
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)
paste <number> until <number>	user.select_range(number_1, number_2) user.idea("action EditorPaste")
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")
perfect	user.idea("action CodeCompletion,action CodeCompletion")
play recording	user.idea("action PlaybackLastMacro")
play recording <user.text> [over]	idea("action PlaySavedMacrosAction") insert(text)

	sleep(500ms) Key("enter")
pop parameters	user.idea("action ParameterInfo")
pop type	user.idea("action ExpressionTypeInfo")
recent	user.idea("action RecentFiles")
refactor	user.idea("action Refactorings.QuickListPopupAction")
refactor <number> until <number>	user.select_range(number_1, number_2) user.idea("action Refactorings.QuickListPopupAction")
refactor <user.text>	user.idea("action Refactorings.QuickListPopupAction") insert(text)
refactor in line	user.idea("action Inline")
refactor last <user.text> [over]	user.idea("find prev {text}, action Refactorings.QuickListPopupAction")
refactor move	user.idea("action Move")
refactor next <user.text> [over]	user.idea("find next {text}, action Refactorings.QuickListPopupAction")
refactor rename	user.idea("action RenameElement")
rename file	user.idea("action RenameFile")
replace last <user.text> [over]	user.idea("find prev {text}, action EditorPaste")
replace next <user.text> [over]	user.idea("find next {text}, action EditorPaste")
run menu	user.idea("action ChooseRunConfiguration")
run test	user.idea("action RunClass")
run test again	user.idea("action Rerun")
select (more | this)	user.idea("action EditorSelectWord")
select camel left	user.extend_camel_left()
select camel right	user.extend_camel_right()
select last <user.text> [over]	user.idea("find prev {text}")
select less	user.idea("action EditorUnSelectWord")
select next <user.text> [over]	user.idea("find next {text}")
smart	user.idea("action SmartTypeCompletion")
step into	user.idea("action StepInto")
step over	user.idea("action StepOver")
step smart	user.idea("action SmartStepInto")
step to line	user.idea("action RunToCursor")
surround [this] with <user.text> [over]	idea("action SurroundWith") sleep(500ms) insert(text)
switch task	user.idea("action tasks.switch")
toggle (bread crumbs | breadcrumbs)	user.idea("action EditorToggleShowBreadcrumbs")
toggle [line] breakpoint	user.idea("action ToggleLineBreakpoint")
toggle comment	code.toggle_comment()
toggle database	user.idea("action ActivateDatabaseToolWindow")
toggle database changes	user.idea("action ActivateDatabaseChangesToolWindow")
toggle debug	user.idea("action ActivateDebugToolWindow")
toggle distraction [free mode]	user.idea("action ToggleDistractionFreeMode")
toggle docked	user.idea("action ToggleDockMode")
toggle docker	user.idea("action ActivateDockerToolWindow")
toggle events	user.idea("action ActivateEventLogToolWindow")
toggle favorites	user.idea("action ActivateFavoritesToolWindow")
toggle find	user.idea("action ActivateFindToolWindow")
toggle floating	user.idea("action ToggleFloatingMode")
toggle fullscreen	user.idea("action ToggleFullScreen")
toggle git	user.idea("action ActivateVersionControlToolWindow")
toggle gutter icons	user.idea("action EditorToggleShowGutterIcons")
toggle indents	user.idea("action EditorToggleShowIndentLines")
toggle last	user.idea("action JumpToLastWindow")
toggle line numbers	user.idea("action EditorToggleShowLineNumbers")
toggle make	user.idea("action ActivatemakeToolWindow")
toggle mark	user.idea("action ToggleBookmark")
toggle mark <number>	user.idea("action ToggleBookmark{number}")
toggle method breakpoint	user.idea("action ToggleMethodBreakpoint")
toggle navigation [bar]	user.idea("action ViewNavigationBar")

toggle parameters	user.idea("action ToggleInlineHintsAction")
toggle pinned	user.idea("action TogglePinnedMode")
toggle power save	user.idea("action TogglePowerSave")
toggle presentation [mode]	user.idea("action TogglePresentationMode")
toggle project	user.idea("action ActivateProjectToolWindow")
toggle recording	user.idea("action StartStopMacroRecording")
toggle run	user.idea("action ActivateRunToolWindow")
toggle split	user.idea("action ToggleSideMode")
toggle status [bar]	user.idea("action ViewStatusBar")
toggle structure	user.idea("action ActivateStructureToolWindow")
toggle terminal	user.idea("action ActivateTerminalToolWindow")
toggle to do	user.idea("action ActivateTODOToolWindow")
toggle tool buttons	user.idea("action ViewToolButtons")
toggle toolbar	user.idea("action ViewToolBar")
toggle whitespace	user.idea("action EditorToggleShowWhitespaces")
toggle windowed	user.idea("action ToggleWindowedMode")
toggle wrap	user.idea("action EditorToggleUseSoftWraps")

KUBECTL

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

OUTLOOK WEB

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

OUTLOOK WIN

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

SLACK MAC

Input	Result
([toggle] mute | unmute)	key(m)
(element | bit) (previous | last)	key(shift-tab)
(element | bit) [next]	key(tab)
(go | undo | toggle) full	key(ctrl-cmd-f)
(insert command | commandify)	key(cmd-shift-c)
(italic | italicize)	key(cmd-i)
(move | next) focus	app.notify("please use the voice command 'focus next' instead of 'next focus'") key(ctrl-`)
(previous | last) (element | bit)	app.notify("please use the voice command 'element last' instead of 'last element'") key(shift-tab)
(previous | last) (section | zone)	app.notify("please use the voice command 'section last' instead of 'last section'") key(shift-f6)
(section | zone) (previous | last)	key(shift-f6)
(section | zone) [next]	key(f6)
(slack | lack) ([toggle] video)	key(v)
(slack | lack) (bull | bullet | bulleted) [list]	key(cmd-shift-8)
(slack | lack) (history [next] | back | backward)	key(cmd-[)
(slack | lack) (my stuff | activity)	key(cmd-shift-m)
(slack | lack) (number | numbered) [list]	key(cmd-shift-7)
(slack | lack) (quotes | quotation)	key(cmd-shift->)
(slack | lack) (react | reaction)	key(cmd-shift-\)
(slack | lack) (slap | slaw | slapper)	key(cmd-right shift-enter)
(slack | lack) (starred [items] | stars)	key(cmd-shift-s)
(slack | lack) [channel] info	key(cmd-shift-i)
(slack | lack) [direct] messages	key(cmd-shift-k)
(slack | lack) directory	key(cmd-shift-e)
(slack | lack) forward	key(cmd-])
(slack | lack) huddle	key(cmd-shift-h)
(slack | lack) invite	key(a)
(slack | lack) shortcuts	key(cmd-/)
(slack | lack) snippet	key(cmd-shift-enter)
(slack | lack) threads	key(cmd-shift-t)
(slack | lack) unread [messages]	key(cmd-shift-a)
(strike | strikethrough)	key(cmd-shift-x)
[next] (element | bit)	app.notify("please use the voice command 'element next' instead of 'next element'") key(tab)
[next] (section | zone)	app.notify("please use the voice command 'section next' instead of 'next section'") key(f6)
add line	key(shift-enter)
bold	key(cmd-b)
emote <user.text>	"{text}"
focus (move | next)	key(ctrl-`)
grab left	key(shift-up)
grab right	key(shift-down)
insert code	key(cmd-shift-alt-c)
insert link	key(cmd-shift-u)
toggle left sidebar	key(cmd-shift-d)
toggle right sidebar	key(cmd-.)
workspace <number>	key("cmd-{number}")

SLACK WIN

Input	Result
([toggle] mute | unmute)	key(m)
(element | bit) (previous | last)	key(shift-tab)
(element | bit) [next]	key(tab)
(insert command | commandify)	key(ctrl-shift-c)
(italic | italicize)	key(ctrl-i)
(move | next) focus	app.notify("please use the voice command 'focus next' instead of 'next focus'") key(ctrl-`)
(previous | last) (element | bit)	app.notify("please use the voice command 'element last' instead of 'last element'") key(shift-tab)
(previous | last) (section | zone)	app.notify("please use the voice command 'section last' instead of 'last section'") key(shift-f6)
(section | zone) (previous | last)	key(shift-f6)
(section | zone) [next]	key(f6)
(slack | lack) ([toggle] video)	key(v)
(slack | lack) (bull | bullet | bulleted) [list]	key(ctrl-shift-8)
(slack | lack) (history [next] | back | backward)	key(alt-left)
(slack | lack) (my stuff | activity)	key(ctrl-shift-m)
(slack | lack) (number | numbered) [list]	key(ctrl-shift-7)
(slack | lack) (quotes | quotation)	key(ctrl-shift-9)
(slack | lack) (react | reaction)	key(ctrl-shift-\)
(slack | lack) (starred [items] | stars)	key(ctrl-shift-s)
(slack | lack) [channel] info	key(ctrl-shift-i)
(slack | lack) [direct] messages	key(ctrl-shift-k)
(slack | lack) directory	key(ctrl-shift-e)
(slack | lack) forward	key(alt-right)
(slack | lack) invite	key(a)
(slack | lack) shortcuts	key(ctrl-/)
(slack | lack) snippet	key(ctrl-shift-enter)
(slack | lack) threads	key(ctrl-shift-t)
(slack | lack) unread [messages]	key(ctrl-shift-a)
(strike | strikethrough)	key(ctrl-shift-x)
[next] (element | bit)	app.notify("please use the voice command 'element next' instead of 'next element'") key(tab)
[next] (section | zone)	app.notify("please use the voice command 'section next' instead of 'next section'") key(f6)
add line	key(shift-enter)
bold	key(ctrl-b)
emote <user.text>	"{text}"
focus (move | next)	key(ctrl-`)
grab left	key(shift-up)
grab right	key(shift-down)
insert code	insert("```")
toggle left sidebar	key(ctrl-shift-d)
toggle right sidebar	key(ctrl-.)
workspace <number>	key("ctrl-{number}")

TALON REPL

Input	Result
debug {user.running} windows	insert("actions.user.talon_debug_app_windows('{running}')") key(enter)
debug action {user.talon_actions}	insert("actions.find('{user.talon_actions}')") key(enter)
debug all windows	insert("actions.user.talon_pretty_print(ui.windows())") key(enter)
debug list {user.talon_lists}	insert("actions.user.talon_pretty_print(registry.lists['{talon_lists}'])") key(enter)
debug modes	insert("actions.user.talon_pretty_print(scope.get('mode'))") key(enter)
debug running apps	insert("actions.user.talon_pretty_print(ui.apps(background=False))") key(enter)
debug scope {user.talon_scopes}	insert("actions.user.talon_pretty_print(scope.get('{talon_scopes}'))") key(enter)
debug settings	insert("actions.user.talon_pretty_print(registry.settings)") key(enter)
debug tags	insert("actions.user.talon_pretty_print(registry.tags)") key(enter)
test <phrase>	insert("sim('{phrase}')") key(enter)
test last	phrase = user.history_get(1) command = "sim('{phrase}')" insert(command) key(enter)
test numb <number_small>	phrase = user.history_get(number_small) command = "sim('{phrase}')"

TEAMS

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

TEAMS MAC

Input	Result
(raise | lower) hand	key(super-shift-k)
(save | send) meeting request	key(super-s)
[expand] compose [box]	key(super-shift-x)
[go] [to] (prev | previous) [list] item	key(alt-up)
[go] [to] (prev | previous) section	key(super-shift-f6)
[go] [to] compose [box]	key(shift-alt-c)
[go] [to] next [list] item	key(alt-down)
[go] [to] next section	key(super-f6)
[go] [to] search	key(super-e)
[go] [to] sharing toolbar	key(super-shift-space)
[start] new chat	key(super-n)
[start] new line	key(shift-enter)
accept audio call	key(super-shift-s)
accept screen share	key(super-shift-a)
accept video call	key(super-shift-a)
attach file	key(shift-alt-o)
attach local file	key(shift-alt-o) sleep(100ms) key(down) key(space)
close	key(escape)
decline call	key(super-shift-d)
decline screen share	key(super-shift-d)
go to	key(super-g)
go to (prev | previous) (day | week)	key(super-alt-left)
go to current time	key(alt-.)
go to next (day | week)	key(super-alt-right)
go to suggested time	key(alt-shift-s)
join [from] meeting [details]	key(alt-shift-j)
leave team meeting	key(super-shift-h)
move [selected] team down	key(super-shift-down)
move [selected] team up	key(super-shift-up)
open (act | activity)	key(super-1)
open (apps | applications)	key(ctrl-`)
open calendar	key(super-4)
open calls	key(super-5)
open chat	key(super-2)
open files	key(super-6)
open filter	key(super-shift-f)
open help	key(f1)
open history	key(super-shift-h)
open settings	key(super-,)
open teams	key(super-3)
reply [to] [thread]	key(shift-alt-r)
reset	key(escape) key(escape) key(escape) key(escape)
schedule [a] meeting	key(alt-shift-n)
send message	key(super-enter)
show commands	key(super-/)
show shortcuts	key(super-.)
starch screen share session	key(super-shift-e)
start audio call	key(super-shift-c)
start video call	key(super-shift-u)

toggle mute	key(super-shift-m)
toggle video	key(super-shift-o)
view day	key(super-alt-1)
view shortcuts	key(super-.)
view week	key(super-alt-3)
view work week	key(super-alt-2)
zoom reset	key(super-0)

TERRAFORM

Input	Result
terraform	"terraform "
terraform apply	"terraform apply "
terraform destroy	"terraform destroy "
terraform format	"terraform fmt\n"
terraform format recursive	"terraform fmt -recursive\n"
terraform help	"terraform -help"
terraform init	"terraform init\n"
terraform init upgrade	"terraform init -upgrade\n"
terraform plan	"terraform plan\n"
terraform state move	"terraform state mv "
terraform validate	"terraform validate\n"

THUNDERBIRD

Input	Result
(open address [book] | address book | open contacts)	user.thunderbird_mod("shift-b")
dev tools	user.thunderbird_mod("shift-i")
go (calendar | lightning)	user.thunderbird_mod("shift-c")
go (mails | messages | inbox)	user.tab_jump(1)
go tasks	user.thunderbird_mod("shift-d")

THUNDERBIRD CALENDAR

Input	Result
(task | event) delete	key(delete)
event new	user.thunderbird_mod("i")
task new	user.thunderbird_mod("d")
toggle today	key(f11)
view <number_small>	user.thunderbird_calendar_view(number_small)
view day	user.thunderbird_calendar_view(1)
view month	user.thunderbird_calendar_view(4)
view multi [week]	user.thunderbird_calendar_view(3)
view week	user.thunderbird_calendar_view(2)

THUNDERBIRD COMPOSER

Input	Result
(draft | mail | message) print	user.thunderbird_mod("p")
(draft | mail | message) save	user.thunderbird_mod("s")
(draft | mail | message) send	user.thunderbird_mod("enter")
(unformatted | raw) paste	user.thunderbird_mod("shift-v")
cite paste	user.thunderbird_mod("shift-o")
go (inbox | thunderbird | main)	user.thunderbird_mod("1")
link delete	user.thunderbird_mod("shift-k")
link new	user.thunderbird_mod("k")
toggle contacts	key(f9)

THUNDERBIRD CONTACTS

Input	Result
contact delete	key(delete)
contact down	key(down)
contact edit	user.thunderbird_mod("i")
contact message	user.thunderbird_mod("m")
contact new	user.thunderbird_mod("n")
contact print	user.thunderbird_mod("p")
contact up	key(up)

THUNDERBIRD INBOX

Input	Result
(mail | message) (down | next)	key(f)
(mail | message) (favorite | unfavorite)	key(s)
(mail | message) (ignore | unignore)	key(k)
(mail | message) (read | unread)	key(m)
(mail | message) (up | last)	key(b)
(mail | message) (watch | unwatch)	key(w)
(mail | message) archive	key(a)
(mail | message) delete	key(delete)
(mail | message) edit	user.thunderbird_mod("e")
(mail | message) forward	user.thunderbird_mod("l")
(mail | message) new	user.thunderbird_mod("n")
(mail | message) open	key(enter)
(mail | message) print	user.thunderbird_mod("p")
(mail | message) reply all	user.thunderbird_mod("shift-r")
(mail | message) reply list	user.thunderbird_mod("shift-l")
(mail | message) reply sender	user.thunderbird_mod("r")
(mail | message) save	user.thunderbird_mod("s")
(mail | message) spam	key(j)
(mail | message) suspend	key(c)
go home	key(alt-home)
toggle (mail | message) [pane]	key(f8)
unread [mail | message] (down | next)	key(n)
unread [mail | message] (up | last)	key(p)

THUNDERBIRD TASKS

Input	Result
(task | event) delete	key(delete)
event new	user.thunderbird_mod("i")
task new	user.thunderbird_mod("d")
toggle today	key(f11)

TWITTER

Input	Result
(show shortcuts | shortcuts help)	key(?)
block account	key(x)
bookmark	key(b)
display settings	insert("gd")
expand photo	key(o)
go book marks	insert("gb")
go direct messages	insert("gm")
go explore	insert("ge")
go home	insert("gh")
go likes	insert("gl")
go lists	insert("gi")
go mentions	insert("gr")
go notifications	insert("gn")
go profile	insert("gp")
go settings	insert("gs")
go to user	insert("gu")
like message	key(l)
load new tweet	key(.)
mute account	key(urge)
new direct message	key(m)
new tweet	key(n)
next tweet	key(j)
open details	key(enter)
page down	key(space)
previous tweet	key(k)
re tweet [message]	key(t)
reply message	key(r)
search	key(/)
send tweet	key(ctrl-enter)
share tweet	key(s)

VSCODE

Input	Result
(go declaration | follow)	user.vscode("editor.action.revealDefinition")
bar explore	user.vscode("workbench.view.explorer")
bar extensions	user.vscode("workbench.view.extensions")
bar marks	user.vscode("workbench.view.extension.bookmarks")
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.toggleSidebarVisibility")
bar test	user.vscode("workbench.view.testing.focus")
break point	user.vscode("editor.debug.action.toggleBreakpoint")
cell last	user.vscode("notebook.focusPreviousEditor")
cell next	user.vscode("notebook.focusNextEditor")
cell run	user.vscode("notebook.cell.execute")
cell run above	user.vscode("notebook.cell.executeCellsAbove")
centered switch	user.vscode("workbench.action.toggleCenteredLayout")
change last	key(shift-alt-f5)
change next	key(alt-f5)
close all tabs	user.vscode("workbench.action.closeAllEditors")
close other tabs	user.vscode("workbench.action.closeOtherEditors")
close tabs way left	user.vscode("workbench.action.closeEditorsToTheLeft")
close tabs way right	user.vscode("workbench.action.closeEditorsToTheRight")
copy line down	user.vscode("editor.action.copyLinesDownAction")
copy line up	user.vscode("editor.action.copyLinesUpAction")
curse redo	user.vscode("cursorRedo")
curse undo	user.vscode("cursorUndo")
debug clean	user.vscode("workbench.debug.panel.action.clearReplAction")
debug console	user.vscode("workbench.debug.action.toggleRepl")
debug continue	user.vscode("workbench.action.debug.continue")
debug pause	user.vscode("workbench.action.debug.pause")
debug restart	user.vscode("workbench.action.debug.restart")
debug start	user.vscode("workbench.action.debug.start")
debug step into	user.vscode("workbench.action.debug.stepInto")
debug step out [of]	user.vscode("workbench.action.debug.stepOut")
debug stopper	user.vscode("workbench.action.debug.stop")
definition peek	user.vscode("editor.action.peekDefinition")
definition show	user.vscode("editor.action.revealDefinition")
definition side	user.vscode("editor.action.revealDefinitionAside")
file clone	user.vscode("fileutils.duplicateFile") sleep(150ms)
file copy local [path]	user.vscode("copyRelativeFilePath")
file copy name	user.vscode("fileutils.copyFileName")
file copy path	user.vscode("copyFilePath")
file create	user.vscode("workbench.action.files.newUntitledFile")
file create relative	user.vscode("fileutils.newFile")
file create root	user.vscode("fileutils.newFileAtRoot")
file create sibling	user.vscode_and_wait("explorer.newFile")
file delete	user.vscode("fileutils.removeFile") sleep(150ms)
file hunt (pace | paste)	user.vscode("workbench.action.quickOpen") sleep(50ms) edit.paste()
file hunt [<user.text>]	user.vscode("workbench.action.quickOpen") sleep(50ms) insert(text or "")

file move	user.vscode("fileutils.moveFile") sleep(150ms)
file open folder	user.vscode("revealFileInOS")
file rename	user.vscode("fileutils.renameFile") sleep(150ms)
file reveal	user.vscode("workbench.files.action.showActiveFileInExplorer")
focus editor	user.vscode("workbench.action.focusActiveEditorGroup")
fold all	user.vscode("editor.foldAll")
fold comments	user.vscode("editor.foldAllBlockComments")
fold five	user.vscode("editor.foldLevel5")
fold four	user.vscode("editor.foldLevel4")
fold one	user.vscode("editor.foldLevel1")
fold seven	user.vscode("editor.foldLevel7")
fold six	user.vscode("editor.foldLevel6")
fold that	user.vscode("editor.fold")
fold those	user.vscode("editor.foldAllMarkerRegions")
fold three	user.vscode("editor.foldLevel3")
fold two	user.vscode("editor.foldLevel2")
format selection	user.vscode("editor.action.formatSelection")
format that	user.vscode("editor.action.formatDocument")
full screen	user.vscode("workbench.action.toggleFullScreen")
fullscreen switch	user.vscode("workbench.action.toggleFullScreen")
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 amend	user.vscode("git.commitStagedAmend")
git commit undo	user.vscode("git.undoCommit")
git diff	user.vscode("git.openChange")
git fetch	user.vscode("git.fetch")
git fetch all	user.vscode("git.fetchAll")
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 force	user.vscode("git.pushForce")
git rebase abort	user.vscode("git.rebaseAbort")
git reveal	user.vscode("git.revealInExplorer")
git revert	user.vscode("git.revertChange")
git stage	user.vscode("git.stage")
git stage all	user.vscode("git.stageAll")
git stash	user.vscode("git.stash")
git stash pop	user.vscode("git.stashPop")
git status	user.vscode("workbench.scm.focus")
git sync	user.vscode("git.sync")
git unstage	user.vscode("git.unstage")
git unstage all	user.vscode("git.unstageAll")
go back	user.vscode("workbench.action.navigateBack")
go edit	user.vscode("workbench.action.navigateToLastEditLocation")
go forward	user.vscode("workbench.action.navigateForward")
go implementation	user.vscode("editor.action.goToImplementation")
go last mark	user.vscode("bookmarks.jumpToPrevious")
go marks	user.deprecate_command("2023-06-06", "go marks", "bar marks") user.vscode("workbench.view.extension.bookmarks")

go next mark	user.vscode("bookmarks.jumpToNext")
go recent [<user.text>]	user.vscode("workbench.action.openRecent") sleep(50ms) insert(text or "") sleep(250ms)
go type	user.vscode("editor.action.goToTypeDefinition")
go usage	user.vscode("references-view.find")
hierarchy peek	user.vscode("editor.showCallHierarchy")
hint show	user.vscode("editor.action.triggerParameterHints")
hover show	user.vscode("editor.action.showHover")
imports fix	user.vscode("editor.action.organizeImports")
install local	user.vscode("workbench.extensions.action.installVSIX")
join lines	user.vscode("editor.action.joinLines")
language switch	user.vscode("workbench.action.editor.changeLanguageMode")
maximize	user.vscode("workbench.action.minimizeOtherEditors")
minimap	user.vscode("editor.action.toggleMinimap")
panel control	user.vscode("workbench.panel.repl.view.focus")
panel output	user.vscode("workbench.panel.output.focus")
panel problems	user.vscode("workbench.panel.markers.view.focus")
panel switch	user.vscode("workbench.action.togglePanel")
panel terminal	user.vscode("workbench.action.terminal.focus")
please [<user.text>]	user.vscode("workbench.action.showCommands") insert(user.text or "")
preview markdown	user.vscode("markdown.showPreview")
problem fix	user.vscode("problems.action.showQuickFixes")
problem last	user.vscode("editor.action.marker.prevInFiles")
problem next	user.vscode("editor.action.marker.nextInFiles")
pull request	user.vscode("pr.create")
refactor rename	user.vscode("editor.action.rename")
refactor that	user.vscode("editor.action.refactor")
refactor this	user.vscode("editor.action.refactor")
references find	user.vscode("references-view.find")
references show	user.vscode("editor.action.goToReferences")
rename that	user.vscode("editor.action.rename")
replace here	user.replace("") key(cmd-alt-l)
restore	user.vscode("workbench.action.evenEditorWidths")
save ugly	user.vscode("workbench.action.files.saveWithoutFormatting")
select (more | this)	user.vscode("editor.action.smartSelect.expand")
select breadcrumb	user.vscode("breadcrumbs.focusAndSelect")
select less	user.vscode("editor.action.smartSelect.shrink")
select word	user.vscode("editor.action.addSelectionToNextFindMatch")
show settings	user.vscode("workbench.action.openGlobalSettings")
show settings folder	user.vscode("workbench.action.openFolderSettings")
show settings folder json	user.vscode("workbench.action.openFolderSettingsFile")
show settings json	user.vscode("workbench.action.openSettingsJson")
show settings workspace	user.vscode("workbench.action.openWorkspaceSettings")
show settings workspace json	user.vscode("workbench.action.openWorkspaceSettingsFile")
show shortcuts	user.vscode("workbench.action.openGlobalKeybindings")
show shortcuts json	user.vscode("workbench.action.openGlobalKeybindingsFile")
show snippets	user.vscode("workbench.action.openSnippets")
skip word	user.vscode("editor.action.moveSelectionToNextFindMatch")
snip (last | previous)	user.vscode("jumpToPrevSnippetPlaceholder")
snip next	user.vscode("jumpToNextSnippetPlaceholder")
step over	user.vscode("workbench.action.debug.stepOver")
suggest show	user.vscode("editor.action.triggerSuggest")

symbol hunt [<user.text>]	user.vscode("workbench.action.gotoSymbol") sleep(50ms) insert(text or "")
symbol hunt all [<user.text>]	user.vscode("workbench.action.showAllSymbols") sleep(50ms) insert(text or "")
terminal <number_small>	user.vscode_terminal(number_small)
terminal external	user.vscode("workbench.action.terminal.openNativeConsole")
terminal last	user.vscode("workbench.action.terminal.focusPrevious")
terminal new	user.vscode("workbench.action.terminal.new")
terminal next	user.vscode("workbench.action.terminal.focusNext")
terminal scroll down	user.vscode("workbench.action.terminal.scrollDown")
terminal scroll up	user.vscode("workbench.action.terminal.scrollUp")
terminal split	user.vscode("workbench.action.terminal.split")
terminal toggle	user.vscode_and_wait("workbench.action.terminal.toggleTerminal")
terminal trash	user.vscode("workbench.action.terminal.kill")
terminal zoom	user.vscode("workbench.action.toggleMaximizedPanel")
test cancel	user.vscode("testing.cancelRun")
test debug	user.vscode("testing.debugAtCursor")
test debug all	user.vscode("testing.debugAll")
test debug failed	user.vscode("testing.debugFailTests")
test debug file	user.vscode("testing.debugCurrentFile")
test debug last	user.vscode("testing.debugLastRun")
test run	user.vscode("testing.runAtCursor")
test run all	user.vscode("testing.runAll")
test run failed	user.vscode("testing.reRunFailTests")
test run file	user.vscode("testing.runCurrentFile")
test run last	user.vscode("testing.reRunLastRun")
theme switch	user.vscode("workbench.action.selectTheme")
toggle mark	user.vscode("bookmarks.toggle")
unfold all	user.vscode("editor.unfoldAll")
unfold that	user.vscode("editor.unfold")
unfold those	user.vscode("editor.unfoldRecursively")
whitespace trim	user.vscode("editor.action.trimTrailingWhitespace")
window close	user.vscode("workbench.action.closeWindow")
window reload	user.vscode("workbench.action.reloadWindow")
wrap switch	user.vscode("editor.action.toggleWordWrap")
zen switch	user.vscode("workbench.action.toggleZenMode")

WINDOWS EXPLORER

Input	Result
go <user.letter>	user.file_manager_open_volume("{letter}:")
go app data	user.file_manager_open_directory("%AppData%")
go program files	user.file_manager_open_directory("%programfiles%")

WINDOWS TERMINAL

Input	Result
find it	edit.find()
find it <phrase>	
focus down	key(ctrl-alt-shift-down)
focus left	key(ctrl-alt-shift-left)
focus right	key(ctrl-alt-shift-right)
focus up	key(ctrl-alt-shift-up)
settings open	key(ctrl-,)
term menu	key(ctrl-shift-f1)

WSL

Input	Result
go <user.letter>	user.file_manager_open_volume("/mnt/{letter}")
(wsl | weasel) reset path detection	user.wsl_reset_path_detection()
(wsl | weasel) speak	user.wsl_speak()

ABBREVIATE

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

EDIT

Input	Result
(indent less | out dent)	edit.indent_less()
(pace | paste) all	user.paste_all()
(pace | paste) enter	edit.paste() key(enter)
(pace | paste) line	user.paste_line()
(pace | paste) line end	user.paste_line_end()
(pace | paste) line start	user.paste_line_start()
(pace | paste) that	edit.paste()
(pace | paste) word	user.paste_word()
(pad | padding)	user.insert_between(" ", " ")
(pad | padding) <user.symbol_key>+	insert(" ") user.insert_many(symbol_key_list) insert(" ")
clear all	user.delete_all()
clear down	edit.extend_line_down() edit.delete()
clear left	edit.delete()
clear line	edit.delete_line()
clear line end	user.delete_line_end()
clear line start	user.delete_line_start()
clear right	user.delete_right()
clear up	edit.extend_line_up() edit.delete()
clear way down	edit.extend_file_end() 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 word	edit.delete_word()
clear word left	edit.extend_word_left() edit.delete()
clear word right	edit.extend_word_right() edit.delete()
clone line	edit.line_clone()
clone that	edit.selection_clone()
copy all	user.copy_all()
copy line	user.copy_line()
copy line end	user.copy_line_end()
copy line start	user.copy_line_start()
copy that	edit.copy()
copy word	user.copy_word()
copy word left	user.copy_word_left()
copy word right	user.copy_word_right()
cut all	user.cut_all()
cut line	user.cut_line()
cut line end	user.cut_line_end()
cut line start	user.cut_line_start()
cut that	edit.cut()
cut word	user.cut_word()
cut word left	user.cut_word_left()
cut word right	user.cut_word_right()
file save	edit.save()
file save all	edit.save_all()

find it	edit.find()
go bottom	edit.file_end()
go down	edit.down()
go left	edit.left()
go line end | tail	edit.line_end()
go line start | head	edit.line_start()
go page down	edit.page_down()
go page up	edit.page_up()
go right	edit.right()
go top	edit.file_start()
go up	edit.up()
go way down	edit.file_end()
go way left	edit.line_start() edit.line_start()
go way right	edit.line_end()
go way up	edit.file_start()
go word left	edit.word_left()
go word right	edit.word_right()
indent [more]	edit.indent_more()
new line above	edit.line_insert_up()
new line below | slap	edit.line_insert_down()
next one	edit.find_next()
paste match	edit.paste_match_style()
redo that	edit.redo()
scroll down	edit.page_down()
scroll up	edit.page_up()
select all	edit.select_all()
select down	edit.extend_line_down()
select left	edit.extend_left()
select line	edit.select_line()
select line end	user.select_line_end()
select line start	user.select_line_start()
select right	edit.extend_right()
select up	edit.extend_line_up()
select way down	edit.extend_file_end()
select way left	edit.extend_line_start()
select way right	edit.extend_line_end()
select way up	edit.extend_file_start()
select word	edit.select_word()
select word left	edit.extend_word_left()
select word right	edit.extend_word_right()
undo that	edit.undo()
zoom in	edit.zoom_in()
zoom out	edit.zoom_out()
zoom reset	edit.zoom_reset()

EDIT SETTINGS

Input	Result
customize {user.talon_settings_csv}	user.edit_text_file(talon_settings_csv) sleep(500ms) edit.file_end()

FILE EXTENSION

Input	Result
{user.file_extension}	"{file_extension}"

HELP

Input	Result
(help formatters | help format | format help)	user.help_formatters(user.get_formatters_words())
help active	user.help_context_enabled()
help alphabet	user.help_list("user.letter")
help arrows	user.help_list("user.arrow_key")
help context {user.help_contexts}	user.help_selected_context(help_contexts)
help context	user.help_context()
help function keys	user.help_list("user.function_key")
help help	user.help_search("help")
help modifier	user.help_list("user.modifier_key")
help numbers	user.help_list("user.number_key")
help punctuation	user.help_list("user.punctuation")
help scope	user.help_scope_toggle()
help search <user.text>	user.help_search(text)
help special keys	user.help_list("user.special_key")
help symbols	user.help_list("user.symbol_key")

HELP OPEN

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

HELP SCOPE OPEN

Input	Result
scope (hide | close)	user.help_scope_toggle()

HOMOPHONES

Input	Result
phones [<user.ordinals>] word left	n = ordinals or 1 user.words_left(n - 1) edit.extend_word_left() user.homophones_show_selection()
phones [<user.ordinals>] word right	n = ordinals or 1 user.words_right(n - 1) edit.extend_word_right() user.homophones_show_selection()
phones <user.homophones_canonical>	user.homophones_show(homophones_canonical)
phones force	user.homophones_force_show_selection()
phones force <user.homophones_canonical>	user.homophones_force_show(homophones_canonical)
phones hide	user.homophones_hide()
phones that	user.homophones_show_auto()
phones word	edit.select_word() user.homophones_show_selection()

HOMOPHONES OPEN

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

KEYS

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

DICTATION MODE

Input	Result
(no cap | no-caps) that	user.dictation_reformat_no_cap()
(no space | no-space) that	user.dictation_reformat_no_space()
cap that	user.dictation_reformat_cap()
escape <user.text>	user.dictation_insert(user.text)
format selection <user.formatters>	user.formatters_reformat_selection(formatters)
press <user.keys>	key(keys)
press <user.modifiers>	key(modifiers)
(no cap | no-caps)	user.dictation_format_no_cap()
(no space | no-space)	user.dictation_format_no_space()
(nope | scratch) selection	edit.delete()
<user.raw_prose>	user.dictation_insert(raw_prose)
cap	user.dictation_format_cap()
clear left <number_small> (character | characters)	edit.extend_left() repeat(number_small - 1) edit.delete()
clear left <number_small> (word | words)	edit.extend_word_left() repeat(number_small - 1) edit.delete()
clear right <number_small> (character | characters)	edit.extend_right() repeat(number_small - 1) edit.delete()
clear right <number_small> (word | words)	edit.extend_word_right() repeat(number_small - 1) edit.delete()
formatted <user.format_text>	user.dictation_insert_raw(format_text)
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 line end	edit.line_end()
go line start	edit.line_start()
go right <number_small> (word | words)	edit.word_right() repeat(number_small - 1)
go up <number_small> (line | lines)	edit.up() repeat(number_small - 1)
nope that | scratch that	user.clear_last_phrase()
select left <number_small> (character | characters)	edit.extend_left() repeat(number_small - 1)
select left <number_small> (word | words)	edit.extend_word_left() repeat(number_small - 1)
select right <number_small> (character | characters)	edit.extend_right() repeat(number_small - 1)
select right <number_small> (word | words)	edit.extend_word_right() repeat(number_small - 1)
select that	user.select_last_phrase()
spell that <user.formatters> <user.letters>	result = user.formatted_text(letters, formatters) user.dictation_insert_raw(result)
spell that <user.letters>	user.dictation_insert(letters)

DRAGON MODE

Input	Result
dragon mode	user.dragon_mode()
talon mode	user.talon_mode()

LANGUAGE MODES

Input	Result
clear language modes	user.code_clear_language_mode()
force {user.language_mode}	user.code_set_language_mode(language_mode)

MODES

Input	Result
command mode	mode.disable("sleep") mode.disable("dictation") mode.enable("command")
dictation mode	mode.disable("sleep") mode.disable("command") mode.enable("dictation") user.code_clear_language_mode() user.gdb_disable()
mixed mode	mode.disable("sleep") mode.enable("dictation") mode.enable("command")

SLEEP MODE

Input	Result
(talon wake)+	speech.enable()
(welcome back)+	user.mouse_wake() user.history_enable() user.talon_mode()

SLEEP MODE NOT DRAGON

Input	Result
(wake up)+	speech.enable()

SLEEP MODE WAV2LETTER

Input	Result
<phrase>	skip()

TO SLEEP MODE

Input	Result
sleep all [<phrase>]	user.switcher_hide_running() user.history_disable() user.homophones_hide() user.help_hide() user.mouse_sleep() speech.disable() user.engine_sleep()
talon sleep [<phrase>]	speech.disable()

TO SLEEP MODE NOT DRAGON

Input	Result
go to sleep [<phrase>]	speech.disable()

MOUSE GRID

Input	Result
M grid	app.notify("please use the voice command 'mouse grid' instead of 'm grid'") user.grid_select_screen(1) user.grid_activate()

MOUSE GRID ALWAYS

Input	Result
grid <user.number_key>+	user.grid_activate() user.grid_narrow_list(number_key_list)
grid screen [<number>]	user.grid_select_screen(number or 1) user.grid_activate()
grid win	user.grid_place_window() user.grid_activate()
mouse grid	user.grid_select_screen(1) user.grid_activate()

MOUSE GRID OPEN

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

NUMBERS

Input	Result
<user.number_string>	"{number_string}"

SCREENS

Input	Result
screen numbers	user.screens_show_numbering()

SNIPPETS

Input	Result
snip {user.snippet_with_phrase} <user.text>	user.insert_snippet_by_name_with_phrase(snippet_with_phrase, text)
snip {user.snippet}	user.insert_snippet_by_name(snippet)

TEXT

Input	Result
{user.prose_formatter} <user.prose> over	user.insert_formatted(prose, prose_formatter)
{user.prose_formatter} <user.prose>	user.insert_formatted(prose, prose_formatter)
<user.format_code>+ over	user.insert_many(format_code_list)
<user.format_code>+	user.insert_many(format_code_list)
<user.formatters> that	user.formatters_reformat_selection(user.formatters)
before that	user.before_last_phrase()
nope that | scratch that	user.clear_last_phrase()
nope that was <user.formatters>	user.formatters_reformat_last(formatters)
phrase <user.text> over	user.add_phrase_to_history(text) insert(text)
phrase <user.text>	user.add_phrase_to_history(text) insert(text)
proud <user.word>	user.insert_formatted(word, "CAPITALIZE_FIRST_WORD")
recent close	user.phrase_history_hide()
recent copy <number_small>	clip.set_text(user.get_recent_phrase(number_small))
recent list	user.toggle_phrase_history()
recent repeat <number_small>	recent_phrase = user.get_recent_phrase(number_small) user.add_phrase_to_history(recent_phrase) insert(recent_phrase)
select that	user.select_last_phrase()
word <user.word>	user.add_phrase_to_history(word) insert(word)

EDIT VOCABULARY

Input	Result
copy name to replacements as <phrase>	user.add_selection_to_words_to_replace(phrase, "name")
copy name to vocab [as <phrase>]	user.add_selection_to_vocabulary(phrase or "", "name")
copy noun to replacements as <phrase>	user.add_selection_to_words_to_replace(phrase, "noun")
copy noun to vocab [as <phrase>]	user.add_selection_to_vocabulary(phrase or "", "noun")
copy to replacements as <phrase>	user.add_selection_to_words_to_replace(phrase)
copy to vocab [as <phrase>]	user.add_selection_to_vocabulary(phrase or "")

WEBSITES AND SEARCH ENGINES

Input	Result
{user.search_engine} (that | this)	text = edit.selected_text() user.search_with_search_engine(search_engine, text)
{user.search_engine} hunt <user.text>	user.search_with_search_engine(search_engine, user.text)
{user.search_engine} paste	user.search_with_search_engine(search_engine, clip.text())
open {user.website}	user.open_url(website)
open paste	user.open_url(clip.text())
open that	user.open_url(edit.selected_text())

TABS

Input	Result
go tab <number>	user.tab_jump(number)
go tab final	user.tab_final()
tab (last | previous)	app.tab_previous()
tab (open | new)	app.tab_open()
tab (reopen | restore)	app.tab_reopen()
tab close	user.tab_close_wrapper()
tab duplicate	user.tab_duplicate()
tab next	app.tab_next()

WINDOW MANAGEMENT

Input	Result
focus <user.running_applications>	user.switcher_focus(running_applications)
focus last	user.switcher_focus_last()
focus	user.switcher_menu()
launch <user.launch_applications>	user.switcher_launch(launch_applications)
running close	user.switcher_hide_running()
running list	user.switcher_toggle_running()
snap <user.running_applications> [screen] <number>	user.move_app_to_screen(running_applications, number)
snap <user.running_applications> <user.window_snap_position>	user.snap_app(running_applications, window_snap_position)
snap <user.window_snap_position>	user.snap_window(window_snap_position)
snap last [screen]	user.move_window_previous_screen()
snap next [screen]	user.move_window_next_screen()
snap screen <number>	user.move_window_to_screen(number)
window (new | open)	app.window_open()
window close	app.window_close()
window hide	app.window_hide()
window last	app.window_previous()
window next	app.window_next()

SETTINGS OVERIDE

Input	Result
clipboard disable	user.clipboard_disable()
clipboard enable	user.clipboard_enable()

SIMPLE TEST

Input	Result
select everything	key(ctrl-a)

BATCH

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

C

Input	Result
[state] (undefine | undeaf) <user.text>	"
[state] define <user.text>	"
[state] if (define | deaf) <user.text>	"
<user.c_keywords>	"{c_keywords}"
<user.c_pointers>	"{c_pointers}"
<user.c_signed>	"{c_signed}"
<user.c_types>	"{c_types}"
<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}"
include <user.code_libraries>	user.code_insert_library(code_libraries, "") key(end enter)
int main	user.insert_between("int main(", ")")
push braces	edit.line_end()
standard <user.stdint_types>	"{stdint_types}"
standard cast to <user.stdint_cast>	"{stdint_cast}"
state (undefine | undeaf)	"#undef "
state default	"default:\nbreak;"
state define	"#define "
state error	"#error "
state if (define | deaf)	"#ifdef "
state include	insert("#include ")
state include local	user.insert_between('
state include system	user.insert_between("
state pragma	"#pragma "
state pre else if	"#elif "
state pre end	"#endif "
state pre if	"#if "
state type deaf	insert("typedef ")
state type deaf struct	insert("typedef struct") insert("{\n\n}") edit.up() key('tab')
toggle includes	user.code_toggle_libraries()

CSS

Input	Result
(value | state) {user.css_global_value}	"{css_global_value}"
[value] current color	"currentColor"
at {user.css_at_rule}	"@{css_at_rule} "
attribute [<user.text>]	name = user.formatted_text(text or "", "DASH_SEPARATED") user.insert_between("[{name}", "]")
block	user.code_block()
op important	" !important"
op var	user.insert_between("var(--", ")")
prop <user.text>	name = user.formatted_text(text, "DASH_SEPARATED") user.insert_between("{name}: ", ";")
rule <user.text>	name = user.formatted_text(text, "DASH_SEPARATED") insert("{name}: ")
unit {user.css_unit}	insert(css_unit)
value <user.number_string> [{user.css_unit}]	"{number_string}{css_unit or "}"
value <user.number_string> point <digit_string> [{user.css_unit}]	"{number_string}.{digit_string}{css_unit or "}"
value <user.text>	user.insert_formatted(text, "DASH_SEPARATED")
variable <user.text>	name = user.formatted_text(text, "DASH_SEPARATED") insert("var(--{name})")

DART

Input	Result
[state] {user.java_modifier}	insert(user.java_modifier + " ")
boxed [type] {user.java_boxed_type}	insert(user.java_boxed_type + " ")
generic [type] {user.java_generic_data_structure}	user.insert_between(java_generic_data_structure + "<", ">")
op array	user.code_operator_subscript()
op new	insert("new ")
state var	"var "
type {user.code_type} array	insert(user.code_type) user.code_operator_subscript()

JAVA

Input	Result
[state] {user.java_modifier}	insert(user.java_modifier + " ")
boxed [type] {user.java_boxed_type}	insert(user.java_boxed_type + " ")
generic [type] {user.java_generic_data_structure}	user.insert_between(java_generic_data_structure + "<", ">")
op array	user.code_operator_subscript()
op new	insert("new ")
state var	"var "
type {user.code_type} array	insert(user.code_type) user.code_operator_subscript()

JAVASCRIPT

Input	Result
(op | is) strict equal	" === "
(op | is) strict not equal	" !== "
dot {user.code_common_member_function}	user.insert_between(".{code_common_member_function}(", ")")
from import	user.insert_between(' from "', "")
op null else	" ?? "
state async	"async "
state await	"await "
state const	"const "
state export	"export "
state filter	app.notify('ERROR: Command deprecated; please use "dot filter"')
state let	"let "
state map	app.notify('ERROR: Command deprecated; please use "dot map"')
state reduce	app.notify('ERROR: Command deprecated; please use "dot reduce"')
state spread	"..."
state var	"var "

LUA

Input	Result
index (var | variable) <user.text>	var = user.formatted_text(text, "snake") insert("[{var}]")
index <user.word>	'["{word}"]'
method <user.text>	insert(":") user.code_public_function_formatter(text) insert("()") edit.left()
require <user.code_libraries>	user.code_insert_library("", code_libraries) key(end enter)
self dot	"self."
state (variable | var) [<user.text>] [over]	user.code_public_variable_formatter(text)
state append string	" .. "
state end	"end"
state label <user.text>	insert("::") user.insert_formatted(text, "snake") insert("::")
state local	"local"
state local (variable | var) [<user.text>] [over]	insert("local ") user.code_private_variable_formatter(text)
state repeat	"repeat"
state return (null | nil)	"return nil"
state return dick	user.insert_between("return {", "}")
state return false	"return false"
state return table	user.insert_between("return {", "}")
state return true	"return true"
state then	"then"
state until	"until"

STYLUA

Input	Result
lint ignore	"-- stylua: ignore"
lint ignore end	"-- stylua: ignore end"
lint ignore start	"-- stylua: ignore start"

MARKDOWN

Input	Result
{user.markdown_code_block_language} block	user.insert_snippet("```{markdown_code_block_language}\n$0\n```")
(level | heading | header) five	edit.line_start() "
(level | heading | header) four	edit.line_start() "
(level | heading | header) one	edit.line_start() "
(level | heading | header) six	edit.line_start() "
(level | heading | header) three	edit.line_start() "
(level | heading | header) two	edit.line_start() "
link	"[]()" key(left:3)
list [one]	edit.line_start() "- "
list five	edit.line_start() " - "
list four	edit.line_start() " - "
list six	edit.line_start() " - "
list three	edit.line_start() " - "
list two	edit.line_start() " - "

PHP

Input	Result
(op | is) loosely equal	" == "
(op | is) loosely not equal	" != "
state catch	"catch (\\Throwable $exception) {\n"
state try	"try {\n"
var <phrase> [over]	insert("$") insert(user.formatted_text(phrase, "PRIVATE_CAMEL_CASE"))

PROTO

Input	Result
block	user.code_block()
op equals	" = "
repeated type {user.code_type}	"repeated {code_type}"
state enum	"enum "
state import	"import "
state import public	"import public "
state message	"message "
state option	"option "
state package	"package "
state repeated	"repeated "
state reserved	"reserved "
type {user.code_type}	"{code_type}"

PYTHON

Input	Result
[state] except {user.python_exception}	"except {python_exception}:"
[state] raise {user.python_exception}	user.insert_between("raise {python_exception}(", ")")
dock {user.python_docstring_fields}	insert("{python_docstring_fields}") edit.left()
dock returns type {user.code_type}	user.insert_between(":rtype ", ": {code_type}")
dock string	user.code_comment_documentation()
dock type {user.code_type}	user.insert_between(":type ", ": {code_type}")
dunder in it	"__init__"
from import	user.insert_between("from ", " import ")
import <user.code_libraries>	user.code_insert_library(code_libraries, "") key(end enter)
pie test	"pytest"
self taught	"self."
state (def | deaf | deft)	"def "
state except	"except "
state past	"pass"
state raise	"raise "
state try	"try:\n"
toggle imports	user.code_toggle_libraries()

R

Input	Result
function define <user.text>	user.code_private_function(text)
(chain | pipe that)	key(end) " %>%" key(enter)
(op | is) in	" %in% "
library <user.code_libraries>	user.code_insert_library(code_libraries, "") key(end enter)
named arg {user.code_parameter_name}	user.code_insert_named_argument(code_parameter_name)
state na	insert("NA")
toggle library	user.code_toggle_libraries()

RUBY

Input	Result
instance <user.text>	insert("@") user.code_public_variable_formatter(text)
args pipe	user.insert_between("|", "|")
dock string	user.code_comment_documentation()
state begin	"begin"
state end	"end"
state module	"module "
state rescue	"rescue "

RUST

Input	Result
block dock comment	user.code_comment_documentation_block()
borrow	"&"
borrow mutable	"&mut "
dynamic trait {user.code_trait}	insert("dyn {code_trait}")
implement (struct | structure)	user.code_state_implements()
implemented trait {user.code_trait}	insert("impl {code_trait}")
inner block dock comment	user.code_comment_documentation_block_inner()
inner dock comment	user.code_comment_documentation_inner()
is implemented trait	": impl "
is implemented trait {user.code_trait}	": impl {code_trait}"
is some	user.code_insert_is_not_null()
macro {user.code_macros}	user.code_insert_macro(code_macros, "")
macro wrap {user.code_macros}	user.code_insert_macro(code_macros, edit.selected_text())
returns implemented trait	" -> impl "
returns implemented trait {user.code_trait}	" -> impl {code_trait}"
self taught	"self."
state (a sink | async | asynchronous)	"async "
state (dyn | dynamic)	"dyn "
state (funk | func | function)	"fn "
state (imp | implements)	"impl "
state (mod | module)	"mod "
state (mute | mutable)	"mut "
state (pub | public)	"pub "
state (pub | public) crate	"pub(crate) "
state (some | sum)	"Some"
state (struct | structure) <user.text>	insert("struct ") insert(user.formatted_text(text, "PUBLIC_CAMEL_CASE"))
state constant	"const "
state enum <user.text>	insert("enum ") insert(user.formatted_text(text, "PUBLIC_CAMEL_CASE"))
state if let (ok | okay)	user.insert_between("if let Ok(", ")")
state if let error	user.insert_between("if let Err(", ")")
state if let some	user.insert_between("if let Some(", ")")
state let	"let "
state let mute	"let mut "
state match	user.code_state_switch()
state ref	"ref "
state ref (mute | mutable)	"ref mut "
state static	"static "
state trait	"trait "
state unsafe	"unsafe "
state use	user.code_import()
toggle use	user.code_toggle_libraries()
trait {user.code_trait}	insert("{code_trait}")
unsafe block	user.code_state_unsafe()
use <user.code_libraries>	user.code_insert_library(code_libraries, "") key(; enter)

SCALA

Input	Result
block string	insert("""""") key("left left left")
op left arrow	" <- "
op plus plus	" ++ "
op right arrow	" -> "
op subtype	" <: "
state {user.scala_keyword}	insert("{scala_keyword} ")
state {user.scala_modifier}	insert("{user.scala_modifier} ")
state match	user.code_state_switch()

SQL

Input	Result
ascending	" ASC"
column	key(return) ", "
count	user.code_insert_function("Count", "")
date	user.insert_between("DATE '", "'")
descending	" DESC"
distinct	"DISTINCT "
dot i d	".id"
from	"FROM "
group by	"GROUP BY "
having	"HAVING "
inner join	user.insert_between("INNER JOIN ", " ON ")
inner join using	user.insert_between("INNER JOIN ", " USING ")
left outer join	user.insert_between("LEFT OUTER JOIN ", " ON ")
order by	"ORDER BY "
right outer join	user.insert_between("RIGHT OUTER JOIN ", " ON ")
select	"SELECT "
select star from	"SELECT *\nFROM "
where	"WHERE "
with	key(enter up) "WITH AS (" key(enter tab) "SELECT " key(enter shift-tab) edit.extend_line_end() edit.delete() ") " key(delete up:2 right:3)

COMMENT BLOCK

Input	Result
(line | inline) block comment <user.text> over	
(line | inline) block comment <user.text>	
block comment	user.code_comment_block()
block comment <user.text> over	
block comment <user.text>	
block comment line	
block comment line <user.text> over	
close block comment	user.code_comment_block_suffix()
open block comment	user.code_comment_block_prefix()

COMMENT DOCUMENTATION

Input	Result
dock comment	user.code_comment_documentation()

COMMENT LINE

Input	Result
(line | inline) comment <user.text> over	
(line | inline) comment <user.text>	
comment	user.code_comment_line_prefix()
comment <user.text> over	
comment <user.text>	
comment line	
comment line <user.text> over	

DATA BOOL

Input	Result
state false	user.code_insert_false()
state true	user.code_insert_true()

DATA NULL

Input	Result
is (none | null)	user.code_insert_is_null()
is not (none | null)	user.code_insert_is_not_null()
state (no | none | nil | null)	user.code_insert_null()

FUNCTIONS

Input	Result
{user.code_function_modifier}* funky <user.text>	user.code_modified_function(code_function_modifier_list or 0, text)
is type <user.code_type>	user.code_insert_type_annotation(code_type)
returns [type] <user.code_type>	user.code_insert_return_type(code_type)
type <user.code_type>	insert(code_type)

FUNCTIONS COMMON

Input	Result
funk <user.code_common_function>	user.code_insert_function(code_common_function, "")
funk cell <number>	user.code_select_function(number - 1, "")
funk wrap <number>	user.code_select_function(number - 1, edit.selected_text())
funk wrap <user.code_common_function>	user.code_insert_function(code_common_function, edit.selected_text())
toggle funk	user.code_toggle_functions()

FUNCTIONS COMMON GUI ACTIVE

Input	Result
toggle funk	user.code_toggle_functions()

IMPERATIVE

Input	Result
block	user.code_block()
state (continue | next)	user.code_next()
state break	user.code_break()
state case	user.code_state_case()
state do	user.code_state_do()
state else	user.code_state_else()
state else if	user.code_state_else_if()
state for	user.code_state_for()
state for in	user.code_state_for_each()
state goto	user.code_state_go_to()
state if	user.code_state_if()
state loop	user.code_state_infinite_loop()
state return	user.code_state_return()
state switch	user.code_state_switch()
state while	user.code_state_while()

KEYWORDS

Input	Result
keyword (<user.code_keyword>+)	user.code_keyword(code_keyword_list)

LIBRARIES

Input	Result
state import	user.code_import()

LIBRARY GUI OPEN

Input	Result
toggle libraries close	user.code_toggle_libraries()

OBJECT ORIENTED

Input	Result
self dot	user.code_self() user.code_operator_object_accessor()
state class	user.code_define_class()
state self	user.code_self()

OPERATORS ARRAY

Input	Result
op subscript	user.code_operator_subscript()

OPERATORS ASSIGNMENT

Input	Result
(op | logical | bitwise) (ex | exclusive) or equals	user.code_operator_bitwise_exclusive_or_assignment()
[(op | logical | bitwise)] (left shift | shift left) equals	user.code_operator_bitwise_left_shift_assignment()
[(op | logical | bitwise)] (right shift | shift right) equals	user.code_operator_bitwise_right_shift_assignment()
[op] bit [wise] and equals	user.code_operator_bitwise_and_assignment()
[op] bit [wise] or equals	user.code_operator_bitwise_or_assignment()
[op] increment	user.code_operator_increment()
op (equals | assign)	user.code_operator_assignment()
op (minus | subtract) equals	user.code_operator_subtraction_assignment()
op (plus | add) equals	user.code_operator_addition_assignment()
op (times | multiply) equals	user.code_operator_multiplication_assignment()
op divide equals	user.code_operator_division_assignment()
op mod equals	user.code_operator_modulo_assignment()
op or equals	user.code_or_operator_assignment()

OPERATORS BITWISE

Input	Result
(op | logical | bitwise) (ex | exclusive) or	user.code_operator_bitwise_exclusive_or()
(op | logical | bitwise) (left shift | shift left)	user.code_operator_bitwise_left_shift()
(op | logical | bitwise) (right shift | shift right)	user.code_operator_bitwise_right_shift()
[op] bitwise and	user.code_operator_bitwise_and()
[op] bitwise or	user.code_operator_bitwise_or()

OPERATORS LAMBDA

Input	Result
op lambda	user.code_operator_lambda()

OPERATORS MATH

Input	Result
(op (power | exponent) | to the power [of])	user.code_operator_exponent()
(op | is) (greater | more)	user.code_operator_greater_than()
(op | is) (less | below) [than]	user.code_operator_less_than()
(op | is) equal	user.code_operator_equal()
(op | is) greater [than] or equal	user.code_operator_greater_than_or_equal_to()
(op | is) in	user.code_operator_in()
(op | is) less [than] or equal	user.code_operator_less_than_or_equal_to()
(op | is) not equal	user.code_operator_not_equal()
(op | is) not in	user.code_operator_not_in()
(op | logical) and	user.code_operator_and()
(op | logical) or	user.code_operator_or()
(op | pad) colon	" : "
op (minus | subtract)	user.code_operator_subtraction()
op (plus | add)	user.code_operator_addition()
op (times | multiply)	user.code_operator_multiplication()
op divide	user.code_operator_division()
op mod	user.code_operator_modulo()

OPERATORS POINTER

Input	Result
op address of	user.code_operator_address_of()
op arrow	user.code_operator_structure_dereference()
op dereference	user.code_operator_indirection()

TALON

Input	Result
action block	user.insert_between("action(", "):")
application [require] [{user.talon_apps}]	app = talon_apps or "" user.paste("app: {app}")
capture {user.talon_captures}	"<{talon_captures}>"
dot talon	insert(".talon")
funk {user.talon_actions}	user.code_insert_function(talon_actions, edit.selected_text())
funk cell <number>	user.code_select_function(number - 1, "")
funk wrap <number>	user.code_select_function(number - 1, edit.selected_text())
funk wrap <user.code_common_function>	user.code_insert_function(code_common_function, edit.selected_text())
host require	hostname = user.talon_get_hostname() user.paste("hostname: {hostname}\n")
key <user.keys> over	"{keys}"
key <user.modifiers> over	"{modifiers}"
linux require	insert("os: linux\n")
list {user.talon_lists}	"{{{talon_lists}}}"
mac require	insert("os: mac\n")
mode require [{user.talon_modes}]	mode = talon_modes or "" user.paste("mode: {mode}")
setting {user.talon_settings}	user.paste("{talon_settings} = ")
setting block	insert("settings():\n\t")
tag require [{user.talon_tags}]	tag = talon_tags or "" user.paste("tag: {tag}")
tag set [{user.talon_tags}]	tag = talon_tags or "" user.paste("tag(): {tag}")
title require	insert("win.title: ")
win require	insert("os: windows\n")

TERRAFORM

Input	Result
[state] prop {user.terraform_common_property}	insert(user.terraform_common_property) user.code_operator_assignment()
block	user.code_block()
data [source] <user.text>	user.code_terraform_data_source(text)
resource <user.text>	user.code_terraform_resource(text)
state {user.terraform_module_block}	user.code_terraform_module_block(user.terraform_module_block)
type {user.code_type}	insert("{code_type}")

TYPESCRIPT

Input	Result
as const	" as const"
state type	user.insert_between("type ", " = ")
type intersect [<user.code_type>]	" & {code_type or "}"
type union [<user.code_type>]	" | {code_type or "}"

VIMSCRIPT

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

CANCEL

Input	Result
cancel cancel	skip()
ignore [<phrase>]	app.notify("Command ignored")

COMMAND HISTORY

Input	Result
command history	user.history_toggle()
command history clear	user.history_clear()
command history close	user.history_disable()
command history less	user.history_less()
command history more	user.history_more()

DATETIMEINSERT

Input	Result
date insert	insert(user.time_format("%Y-%m-%d"))
date insert UTC	insert(user.time_format_utc("%Y-%m-%d"))
timestamp insert	insert(user.time_format("%Y-%m-%d %H:%M:%S"))
timestamp insert UTC	insert(user.time_format_utc("%Y-%m-%d %H:%M:%S"))
timestamp insert UTC high resolution	insert(user.time_format_utc("%Y-%m-%d %H:%M:%S.%f"))
timestamp insert high resolution	insert(user.time_format("%Y-%m-%d %H:%M:%S.%f"))

DESKTOPS

Input	Result
desk <number_small>	user.desktop(number_small)
desk last	user.desktop_last()
desk next	user.desktop_next()
desk show	user.desktop_show()
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()

DRAFT EDITOR

Input	Result
draft all	edit.select_all() user.draft_editor_open()
draft bottom	edit.extend_file_end() user.draft_editor_open()
draft line	edit.select_line() user.draft_editor_open()
draft submit	user.draft_editor_paste_last()
draft this	user.draft_editor_open()
draft top	edit.extend_file_start() user.draft_editor_open()

DRAFT EDITOR OPEN

Input	Result
draft discard	user.draft_editor_discard()
draft submit	user.draft_editor_submit()

DROPDOWN

Input	Result
drop down <number_small>	key("down:{number_small-1} enter")
drop down up <number_small>	key("up:{number_small} enter")

MACRO

Input	Result
macro copy [{user.saved_macros}]	user.macro_copy(saved_macros or "")
macro copy as <user.text>	user.macro_copy(text)
macro list	user.macro_list()
macro list close	user.macro_list_close()
macro play [{user.saved_macros}]	user.macro_play(saved_macros or "")
macro record	user.macro_record()
macro save as <user.text>	user.macro_save(text)
macro stop	user.macro_stop()

MEDIA

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

MICROPHONE SELECTION

Input	Result
microphone close	user.microphone_selection_hide()
microphone pick <number_small>	user.microphone_select(number_small)
microphone show	user.microphone_selection_toggle()

MOUSE

Input	Result
(dub click | duke)	
(trip click | trip lick)	
<user.modifiers> righty	
<user.modifiers> touch	
camera overlay	tracking.control_debug_toggle()
control mouse	tracking.control_toggle()
control off	user.mouse_sleep()
copy mouse position	user.copy_mouse_position()
curse no	
end drag | drag end	user.mouse_drag_end()
left drag | drag | drag start	
mid click	
mouse hiss down	user.hiss_scroll_down()
mouse hiss up	user.hiss_scroll_up()
right drag | righty drag	
righty	
run calibration	tracking.calibrate()
touch	
wheel down	user.mouse_scroll_down()
wheel down here	user.mouse_move_center_active_window() user.mouse_scroll_down()
wheel downer	user.mouse_scroll_down_continuous()
wheel downer here	user.mouse_move_center_active_window() user.mouse_scroll_down_continuous()
wheel gaze	user.mouse_gaze_scroll()
wheel gaze here	user.mouse_move_center_active_window() user.mouse_gaze_scroll()
wheel left	user.mouse_scroll_left()
wheel left here	user.mouse_move_center_active_window() user.mouse_scroll_left()
wheel right	user.mouse_scroll_right()
wheel right here	user.mouse_move_center_active_window() user.mouse_scroll_right()
wheel stop	user.mouse_scroll_stop()
wheel stop here	user.mouse_move_center_active_window() user.mouse_scroll_stop()
wheel tiny [down]	user.mouse_scroll_down(0.2)
wheel tiny [down] here	user.mouse_move_center_active_window() user.mouse_scroll_down(0.2)
wheel tiny left	user.mouse_scroll_left(0.5)
wheel tiny left here	user.mouse_move_center_active_window() user.mouse_scroll_left(0.5)
wheel tiny right	user.mouse_scroll_right(0.5)
wheel tiny right here	user.mouse_move_center_active_window() user.mouse_scroll_right(0.5)
wheel tiny up	user.mouse_scroll_up(0.2)
wheel tiny up here	user.mouse_move_center_active_window() user.mouse_scroll_up(0.2)
wheel up	user.mouse_scroll_up()
wheel up here	user.mouse_move_center_active_window() user.mouse_scroll_up()
wheel upper	user.mouse_scroll_up_continuous()
wheel upper here	user.mouse_move_center_active_window() user.mouse_scroll_up_continuous()
zoom mouse	tracking.control_zoom_toggle()

MOUSE CURSOR

Input	Result
curse no	user.mouse_hide_cursor()
curse yes	user.mouse_show_cursor()

REPEATER

Input	Result
(repeat phrase | again) [<number_small> times]	core.repeat_partial_phrase(number_small or 1)
(repeat that | twice)	core.repeat_command(1)
<number_small> times	core.repeat_command(number_small - 1)
<user.ordinals>	core.repeat_command(ordinals - 1)
repeat that <number_small> [times]	core.repeat_command(number_small)

SCREENSHOT

Input	Result
grab screen <number_small> clip	user.screenshot_clipboard(number_small)
grab screen <number_small>	user.screenshot(number_small)
grab screen clip	user.screenshot_clipboard()
grab screen interactive	user.screenshot_interactive()
grab screen	user.screenshot()
grab selection clip	user.screenshot_selection_clip()
grab selection	user.screenshot_selection()
grab settings	user.screenshot_settings()
grab window clip	user.screenshot_window_clipboard()
grab window	user.screenshot_window()

SYMBOLS

Input	Result
(brace | curly bracket) that	text = edit.selected_text() user.paste("{{{text}}}")
(comma and | spamma)	", "
(dot dot | dotdot)	".."
(double quote | dub quote) that	text = edit.selected_text() user.paste('"{text}"')
(grave | back tick) that	text = edit.selected_text() user.paste("`{text}`")
(inside parens | args)	user.insert_between("(", ")")
(parens | args) that	text = edit.selected_text() user.paste("({text})")
(square | bracket | square bracket) that	text = edit.selected_text() user.paste("[{text}]")
(triple grave | triple back tick | gravy)	insert("```")
angle that	text = edit.selected_text() user.paste("<{text}>")
arrow	"->"
double dash	"--"
dub arrow	"=>"
ellipsis	"..."
empty dub string	user.insert_between('"', '"')
empty escaped (dub string | dub quotes)	user.insert_between('\\"', '\\"')
empty escaped string	user.insert_between("\\'", "\\'")
empty string	user.insert_between("'", "'")
inside (braces | curly brackets)	user.insert_between("{", "}")
inside (double quotes | dub quotes)	user.insert_between('"', '"')
inside (graves | back ticks)	user.insert_between("`", "`")
inside (quotes | string)	user.insert_between("'", "'")
inside (squares | brackets | square brackets | list)	user.insert_between("[", "]")
inside percent	user.insert_between("%", "%")
new line	"\n"
percent that	text = edit.selected_text() user.paste("%{text}%")
quote that	text = edit.selected_text() user.paste('"{text}"')
triple quote	""""

DRAFT GLOBAL

Input	Result
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)
draft empty	user.draft_show("")
draft show	
draft show <user.draft_window_position>	
draft show large	
draft show small	

DRAFT WINDOW

Input	Result
(change | chuck | clear) <user.draft_anchor>	user.draft_select("{draft_anchor}", "", 1) key(backspace)
(change | chuck | clear) <user.draft_anchor> (through | past) <user.draft_anchor>	user.draft_select(draft_anchor_1, draft_anchor_2, 1) key(backspace)
(post | cursor after) <user.draft_anchor>	user.draft_position_caret("{draft_anchor}", 1)
(pre | cursor | cursor before) <user.draft_anchor>	user.draft_position_caret("{draft_anchor}")
(take | select) <user.draft_anchor>	user.draft_select("{draft_anchor}")
(take | select) <user.draft_anchor> (through | past) <user.draft_anchor>	user.draft_select("{draft_anchor_1}", "{draft_anchor_2}")
<user.formatters> <user.draft_anchor> (through | past) <user.draft_anchor>	user.draft_select(draft_anchor_1, draft_anchor_2, 1) user.formatters_reformat_selection(user.formatters)
<user.formatters> word <user.draft_anchor>	user.draft_select("{draft_anchor}", "", 1) user.formatters_reformat_selection(user.formatters)
replace <user.draft_anchor> with <user.text>	user.draft_select("{draft_anchor}") result = user.formatted_text(text, "NOOP") insert(result)

DRAFT WINDOW OPEN

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

TALON HELPERS

Input	Result
talon copy active app	result = user.talon_get_active_application_info() clip.set_text(result)
talon copy list {user.talon_lists}	user.talon_copy_list(talon_lists)
talon create app context	user.talon_create_app_context()
talon create linux app context	user.talon_create_app_context("linux")
talon create mac app context	user.talon_create_app_context("mac")
talon create windows app context	user.talon_create_app_context("win")
talon debug action {user.talon_actions}	user.talon_action_find("{user.talon_actions}")
talon debug active app	result = user.talon_get_active_application_info() print("**** Dumping active application **** ") print(result) print("**********")
talon debug all settings	user.talon_debug_all_settings()
talon debug list {user.talon_lists}	user.talon_debug_list(talon_lists)
talon debug modes	user.talon_debug_modes()
talon debug scope {user.talon_scopes}	user.talon_debug_scope(talon_scopes)
talon debug setting {user.talon_settings}	user.talon_debug_setting(talon_settings)
talon debug tags	user.talon_debug_tags()
talon test <phrase>	user.talon_sim_phrase(phrase)
talon test last	phrase = user.history_get(1) user.talon_sim_phrase(phrase)
talon test numb <number_small>	phrase = user.history_get(number_small) user.talon_sim_phrase(phrase)
talon (bug report | report bug)	user.open_url("https://github.com/talonhub/community/issues")
talon check updates	menu.check_for_updates()
talon copy bundle	bundle = app.bundle() clip.set_text(bundle)
talon copy context	user.talon_add_context_clipboard()
talon copy context pie	user.talon_add_context_clipboard_python()
talon copy executable	executable = app.executable() clip.set_text(executable)
talon copy name	name = app.name() clip.set_text(name)
talon copy title	title = win.title() clip.set_text(title)
talon dump context	result = user.talon_get_active_context() print(result)
talon dump version	result = user.talon_version_info() print(result)
talon home	menu.open_talon_home()
talon insert version	result = user.talon_version_info() user.paste(result)
talon open log	menu.open_log()
talon open rebel	menu.open_repl()

TEXT NAVIGATION

Input	Result
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)
navigate [{user.arrow_key}] [{user.navigation_action}] [{user.navigation_target_name}] [{user.before_or_after}] [<user.ordinals>] <user.navigation_target>	
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)
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)

BROWSER

Input	Result
(refresh | reload) it	browser.reload()
(refresh | reload) it hard	browser.reload_hard()
[go] forward	browser.go_forward()
address bar | go address | go url	browser.focus_address()
address copy | url copy | copy address | copy url	browser.focus_address() sleep(50ms) edit.copy()
bookmark bar [show]	browser.bookmarks_bar()
bookmark it	browser.bookmark()
bookmark show	browser.bookmarks()
bookmark tabs	browser.bookmark_tabs()
cache show	browser.show_clear_cache()
dev tools [show]	browser.toggle_dev_tools()
downloads show	browser.show_downloads()
extensions show	browser.show_extensions()
go (back | backward)	browser.go_back()
go home	browser.go_home()
go page | page focus	browser.focus_page()
go private	browser.open_private_window()
go to {user.website}	browser.go(website)
history show	browser.show_history()
show cache	browser.show_clear_cache()
show downloads	browser.show_downloads()
show extensions	browser.show_extensions()
show history	browser.show_history()

CHAPTERS

Input	Result
chapter last	user.chapter_previous()
chapter next	user.chapter_next()
go chapter <number>	user.chapter_jump(number)
go chapter final	user.chapter_final()

DEBUGGER

Input	Result
(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()
(stack | back) trace	user.debugger_backtrace()
break here	user.debugger_break_here()
break now	user.debugger_break_now()
clear (break | break point)	user.debugger_clear_breakpoint()
clear (break | break point) <number_small>	user.debugger_clear_breakpoint_id(number_small)
clear all (breaks | break points)	user.debugger_clear_all_breakpoints()
clear line	user.debugger_clear_line()
continue	user.debugger_continue()
debug detach	user.debugger_detach()
debug exit	user.debugger_exit()
debug restart	user.debugger_restart()
debug start	user.debugger_start()
debug stop	user.debugger_stop()
disable (break | break point)	user.debugger_disable_breakpoint()
disable (break | break point) <number_small>	user.debugger_disable_breakpoint_id(number_small)
disable all (breaks | break points)	user.debugger_disable_all_breakpoints()
disassemble	user.debugger_disassemble()
disassemble clipboard	user.debugger_disassemble_clipboard()
disassemble here	user.debugger_disassemble_here()
dump pointers	user.debugger_dump_pointers()
dump string	user.debugger_dump_ascii_string()
dump unicode [string]	user.debugger_dump_unicode_string()
enable (break | break point)	user.debugger_enable_breakpoint()
enable (break | break point) <number_small>	user.debugger_enable_breakpoint_id(number_small)
enable all (breaks | break points)	user.debugger_enable_all_breakpoints()
get register	user.debugger_get_register()
inspect type	user.debugger_inspect_type()
jump to address	user.debugger_goto_address()
jump to clipboard	user.debugger_goto_clipboard()
jump to highlighted	user.debugger_goto_highlighted()
list modules	user.debugger_list_modules()
set register	user.debugger_set_register()
show registers	user.debugger_show_registers()
step into	user.debugger_step_into()
step line	user.debugger_step_line()
step out	user.debugger_step_out()
step over	user.debugger_step_over()
step over line	user.debugger_step_over_line()

EMOJI

Input	Result
emoji {user.emoji}	user.paste(emoji)
emoticon {user.emoticon}	"{emoticon}"
kaomoji {user.kaomoji}	user.paste(kaomoji)

FILE MANAGER

Input	Result
(select | cell) file {user.file_manager_files}	user.file_manager_select_file(file_manager_files)
(select | cell) folder {user.file_manager_directories}	user.file_manager_select_directory(file_manager_directories)
file {user.file_manager_files}	user.file_manager_select_file(file_manager_files)
file numb <number_small>	file = user.file_manager_get_file_by_index(number_small - 1) user.file_manager_select_file(file)
folder numb <number_small>	directory = user.file_manager_get_directory_by_index(number_small - 1) user.file_manager_select_directory(directory)
follow {user.file_manager_directories}	user.file_manager_open_directory(file_manager_directories)
follow numb <number_small>	directory = user.file_manager_get_directory_by_index(number_small - 1) user.file_manager_open_directory(directory)
open <number_small>	file = user.file_manager_get_file_by_index(number_small - 1) user.file_manager_open_file(file)
(go parent | daddy)	user.file_manager_open_parent()
file last	user.file_manager_previous_file_page()
file next	user.file_manager_next_file_page()
folder last	user.file_manager_previous_folder_page()
folder new <user.text>	user.file_manager_new_folder(text)
folder next	user.file_manager_next_folder_page()
go <user.system_path>	user.file_manager_open_directory(system_path)
go back	user.file_manager_go_back()
go forward	user.file_manager_go_forward()
manager close	user.file_manager_hide_pickers()
manager refresh	user.file_manager_update_lists()
manager show	user.file_manager_toggle_pickers()
properties show	user.file_manager_show_properties()
terminal here	user.file_manager_terminal_here()
title force	user.file_manager_refresh_title()

FIND AND REPLACE

Input	Result
clear last <user.text> [over]	user.select_previous_occurrence(text) sleep(100ms) edit.delete()
clear last clip	user.select_previous_occurrence(clip.text()) edit.delete()
clear next <user.text> [over]	user.select_next_occurrence(text) sleep(100ms) edit.delete()
clear next clip	user.select_next_occurrence(clip.text()) sleep(100ms) edit.delete()
comment last <user.text> [over]	user.select_previous_occurrence(text) sleep(100ms) code.toggle_comment()
comment last clip	user.select_previous_occurrence(clip.text()) 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.text()) sleep(100ms) code.toggle_comment()
go last <user.text> [over]	user.select_previous_occurrence(text) sleep(100ms) edit.right()
go last clip	user.select_previous_occurrence(clip.text()) sleep(100ms) edit.right()
go next <user.text> [over]	user.select_next_occurrence(text) edit.right()
go next clip	user.select_next_occurrence(clip.text()) edit.right()
hunt all	user.find_everywhere("")
hunt all (pace | paste)	user.find_everywhere("") sleep(25ms) edit.paste()
hunt all <user.text>	user.find_everywhere(text)
hunt case	user.find_toggle_match_by_case()
hunt expression	user.find_toggle_match_by_regex()
hunt next	user.find_next()
hunt previous	user.find_previous()
hunt this	user.find("")
hunt this (pace | paste)	user.find("") sleep(25ms) edit.paste()
hunt this <user.text>	user.find(text)
hunt word	user.find_toggle_match_by_word()
paste last <user.text> [over]	user.select_previous_occurrence(text) sleep(100ms) edit.right() edit.paste()
paste next <user.text> [over]	user.select_next_occurrence(text) sleep(100ms) edit.right() edit.paste()
replace <user.text> all	user.replace_everywhere(text)
replace all	user.replace_everywhere("")
replace confirm all	user.replace_confirm_all()
replace confirm that	user.replace_confirm()
replace last <user.text> [over]	user.select_previous_occurrence(text) sleep(100ms) edit.paste()

replace next <user.text> [over]	user.select_next_occurrence(text) sleep(100ms) edit.paste()
replace this [<user.text>]	user.replace(text or "")
select last <user.text> [over]	user.select_previous_occurrence(text)
select last clip	user.select_previous_occurrence(clip.text())
select next <user.text> [over]	user.select_next_occurrence(text)
select next clip	user.select_next_occurrence(clip.text())

LINE COMMANDS

Input	Result
(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)
bend	edit.line_start()
clear [line] <number>	user.select_range(number, number) edit.delete()
clear <number> until <number>	user.select_range(number_1, number_2) edit.delete()
clone [line] <number>	user.line_clone(number)
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()
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()
drag [line] down	edit.line_swap_down()
drag [line] up	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()
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()
go <number>	edit.jump_line(number)
go <number> end	edit.jump_line(number) edit.line_end()
go camel left	user.camel_left()
go camel right	user.camel_right()
lend	edit.line_end()
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()
retab that	edit.indent_less()
select camel left	user.extend_camel_left()
select camel right	user.extend_camel_right()
tab [line] <number>	edit.jump_line(number) edit.indent_more()
tab <number> until <number>	user.select_range(number_1, number_2) edit.indent_more()
tab that	edit.indent_more()

MESSAGING

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

MULTIPLE CURSORS

Input	Result
cursor all	user.multi_cursor_select_all_occurrences()
cursor down	user.multi_cursor_add_below()
cursor less	user.multi_cursor_select_fewer_occurrences()
cursor lines	user.multi_cursor_add_to_line_ends()
cursor more	user.multi_cursor_select_more_occurrences()
cursor multiple	user.multi_cursor_enable()
cursor skip	user.multi_cursor_skip_occurrence()
cursor stop	user.multi_cursor_disable()
cursor up	user.multi_cursor_add_above()

PAGES

Input	Result
go page <number>	user.page_jump(number)
go page final	user.page_final()
page last	user.page_previous()
page next	user.page_next()

SPLITS

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

TERMINAL

Input	Result
clear screen	user.terminal_clear_screen()
copy paste	edit.copy() sleep(50ms) edit.paste()
go <user.system_path>	insert('cd "{system_path}"\n')
katie (up | back)	user.terminal_change_directory("..")
katie [dir] [<user.text>]	user.terminal_change_directory(text or "")
katie root	user.terminal_change_directory_root()
kill all	user.terminal_kill_all()
lisa	user.terminal_list_directories()
lisa all	user.terminal_list_all_directories()
path <user.system_path>	insert('"{system_path}"')
rerun [<user.text>]	user.terminal_rerun_search(text or "")
rerun search	user.terminal_rerun_search("")
run last	user.terminal_run_last()

UNIX UTILITIES

Input	Result
core {user.unix_utility}	"{unix_utility} "

GAZE OCR

Input	Result
{user.ocr_actions} [{user.ocr_modifiers}] (seen | scene) <user.prose_range>	user.perform_ocr_action(ocr_actions, ocr_modifiers or "", prose_range)
(eye | i) (hover | [cursor] move)	user.move_cursor_to_gaze_point()
(eye | i) [left] (touch | click)	user.move_cursor_to_gaze_point() mouse_click(0)
(eye | i) [left] double (touch | click)	user.move_cursor_to_gaze_point() mouse_click(0) mouse_click(0)
(eye | i) <user.modifiers> (touch | click)	user.move_cursor_to_gaze_point() key("{modifiers}:down") mouse_click(0) key("{modifiers}:up")
(eye | i) middle (touch | click)	user.move_cursor_to_gaze_point() mouse_click(2)
(eye | i) right (touch | click)	user.move_cursor_to_gaze_point() mouse_click(1)
(eye | i) scroll down	user.move_cursor_to_gaze_point(0, -40) user.mouse_scroll_down()
(eye | i) scroll down half	user.move_cursor_to_gaze_point(0, -40) user.mouse_scroll_down(0.5)
(eye | i) scroll left	user.move_cursor_to_gaze_point(40, 0) user.mouse_scroll_left()
(eye | i) scroll left half	user.move_cursor_to_gaze_point(40, 0) user.mouse_scroll_left(0.5)
(eye | i) scroll right	user.move_cursor_to_gaze_point(-40, 0) user.mouse_scroll_right()
(eye | i) scroll right half	user.move_cursor_to_gaze_point(-40, 0) user.mouse_scroll_right(0.5)
(eye | i) scroll up	user.move_cursor_to_gaze_point(0, 40) user.mouse_scroll_up()
(eye | i) scroll up half	user.move_cursor_to_gaze_point(0, 40) user.mouse_scroll_up(0.5)
(go after | post (seen | scene)) <user.timestamped_prose>	user.move_text_cursor_to_word(timestamped_prose, "after")
(go before | pre (seen | scene)) <user.timestamped_prose>	user.move_text_cursor_to_word(timestamped_prose, "before")
(hover (seen | scene) | cursor move) <user.timestamped_prose>	user.move_cursor_to_word(timestamped_prose)
(revise from <user.timestamped_prose_only> | revise with <user.timestamped_prose_only> cursor)	user.revise_text_starting_with(timestamped_prose_only)
[go] after <user.timestamped_prose> say <user.prose>	user.insert_adjacent_to_text(timestamped_prose, "after", prose)
[go] before <user.timestamped_prose> say <user.prose>	user.insert_adjacent_to_text(timestamped_prose, "before", prose)
[left] (touch | click) <user.timestamped_prose>	user.click_text(timestamped_prose)
[left] double (touch | click) <user.timestamped_prose>	user.double_click_text(timestamped_prose)
<user.modifiers> (touch | click) <user.timestamped_prose>	user.modifier_click_text(modifiers, timestamped_prose)
append with <user.timestamped_prose_only>	user.append_text(timestamped_prose_only)
insert with <user.timestamped_prose_only>	user.insert_text_difference(timestamped_prose_only)
middle (touch | click) <user.timestamped_prose>	user.middle_click_text(timestamped_prose)
ocr show [text]	user.show_ocr_overlay("text")
ocr show [text] near <user.timestamped_prose>	user.show_ocr_overlay("text", timestamped_prose)
ocr show boxes	user.show_ocr_overlay("boxes")
ocr tracker off	user.disconnect_ocr_eye_tracker()
ocr tracker on	user.connect_ocr_eye_tracker()
phones [word] (seen | scene) <user.timestamped_prose>	user.change_text_homophone(timestamped_prose)
prepend with <user.timestamped_prose_only>	user.prepend_text(timestamped_prose_only)
replace [{user.ocr_modifiers}] [seen | scene] <user.prose_range> with <user.prose>	user.replace_text(ocr_modifiers or "", prose_range, prose)
revise through <user.timestamped_prose_only>	user.revise_text_ending_with(timestamped_prose_only)
revise with <user.timestamped_prose_only>	user.revise_text(timestamped_prose_only)
right (touch | click) <user.timestamped_prose>	user.right_click_text(timestamped_prose)
screen [left] (touch | click) <user.onscreen_text>	user.click_text(onscreen_text)

select <user.prose_range>	user.perform_ocr_action("select", "", prose_range)

GAZE OCR DISAMBIGUATION

Input	Result
choose <number_small>	user.choose_gaze_ocr_option(number_small)
choose to	user.choose_gaze_ocr_option(2)
numbers hide	user.hide_gaze_ocr_options()

HUD CONTENT FOCUS

Input	Result
focus indicator hide	user.hud_deactivate_focus_indicator()
focus indicator show	user.hud_activate_focus_indicator()

HUD CONTENT TOOLKIT

Input	Result
toolkit debugging	user.hud_toolkit_debug_options()
toolkit documentation	user.hud_show_documentation()
toolkit lists	user.hud_toolkit_lists()
toolkit microphones	user.show_microphone_options()
toolkit options	user.hud_toolkit_options()
toolkit scope	user.hud_toolkit_scope()
toolkit speech	user.hud_toolkit_speech()
toolkit walkthroughs	user.hud_show_walkthroughs()

HUD CONTENT WALKTHROUGH

Input	Result
head up theme dark	sleep(0.5) user.switch_hud_theme('dark')
head up theme light	sleep(1.5) user.switch_hud_theme('light')
music and video playlist	user.open_url("https://www.youtube.com/watch?v=lyVICt4vdB0&list=PLEelxuGt2Io5jGNnA44S9lRhclhz7po1U&index=1")
(continue|skip step)	user.hud_skip_walkthrough_step()
(previous [step]|last step)	user.hud_previous_walkthrough_step()
skip everything	user.hud_skip_walkthrough_all()

HUD CHOICE COMMANDS

Input	Result
{user.talon_hud_widget_enabled_voice_commands}	user.hud_activate_enabled_voice_command(talon_hud_widget_enabled_voice_commands)
[option] {user.talon_hud_choices}	user.hud_activate_choice(talon_hud_choices)
option {user.talon_hud_numerical_choices}+ {user.talon_hud_choices}	user.hud_activate_choices(talon_hud_numerical_choices_list) user.hud_activate_choice(talon_hud_choices)
option {user.talon_hud_numerical_choices}+	user.hud_activate_choices(talon_hud_numerical_choices_list)

HUD COMMANDS

Input	Result
{user.talon_hud_widget_names} (back|previous)	user.hud_decrease_widget_page(talon_hud_widget_names)
{user.talon_hud_widget_names} (hide|close)	user.hud_disable_id(talon_hud_widget_names)
{user.talon_hud_widget_names} (show|open)	user.hud_enable_id(talon_hud_widget_names)
{user.talon_hud_widget_names} maximize	user.hud_set_widget_preference(talon_hud_widget_names, "minimized", 0)
{user.talon_hud_widget_names} minimize	user.hud_set_widget_preference(talon_hud_widget_names, "minimized", 1)
{user.talon_hud_widget_names} next	user.hud_increase_widget_page(talon_hud_widget_names)
{user.talon_hud_widget_names} options	user.hud_widget_options(talon_hud_widget_names)
(head up focus | focus head up)	user.hud_focus()
head up (drop|stop|confirm)	user.hud_set_setup_mode("*", "")
head up (hide|close) {user.talon_hud_widget_names}	user.hud_disable_id(talon_hud_widget_names)
head up (hide|close)	user.hud_disable()
head up (show|open) {user.talon_hud_widget_names}	user.hud_enable_id(talon_hud_widget_names)
head up (show|open)	user.hud_enable()
head up [enable] auto focus	user.hud_set_auto_focus(1)
head up align {user.talon_hud_widget_names} bottom	user.hud_set_widget_preference(talon_hud_widget_names, "expand_direction", "up")
head up align {user.talon_hud_widget_names} center	user.hud_set_widget_preference(talon_hud_widget_names, "alignment", "center")
head up align {user.talon_hud_widget_names} left	user.hud_set_widget_preference(talon_hud_widget_names, "alignment", "left")
head up align {user.talon_hud_widget_names} right	user.hud_set_widget_preference(talon_hud_widget_names, "alignment", "right")
head up align {user.talon_hud_widget_names} top	user.hud_set_widget_preference(talon_hud_widget_names, "expand_direction", "down")
head up basic {user.talon_hud_widget_names}	user.hud_set_widget_preference(talon_hud_widget_names, "show_animations", 0)
head up blur	user.hud_blur()
head up cancel	user.hud_set_setup_mode("*", "cancel")
head up development start	user.hud_watch_directories() user.hud_watch_walkthrough_files() user.hud_watch_documentation_files()
head up development stop	user.hud_unwatch_directories() user.hud_unwatch_walkthrough_files() user.hud_unwatch_documentation_files()
head up disable auto focus	user.hud_set_auto_focus(0)
head up drag {user.talon_hud_widget_names}+	user.hud_set_setup_mode_multi(talon_hud_widget_names_list, "position")
head up expand {user.talon_hud_widget_names}	user.hud_set_setup_mode(talon_hud_widget_names, "limit")
head up fancy {user.talon_hud_widget_names}	user.hud_set_widget_preference(talon_hud_widget_names, "show_animations", 1)
head up hide {user.talon_hud_widget_names} on sleep	user.hud_set_widget_preference(talon_hud_widget_names, "sleep_enabled", 0)
head up resize {user.talon_hud_widget_names}	user.hud_set_setup_mode(talon_hud_widget_names, "dimension")
head up show {user.talon_hud_widget_names} on sleep	user.hud_set_widget_preference(talon_hud_widget_names, "sleep_enabled", 1)
head up text scale {user.talon_hud_widget_names}	user.hud_set_setup_mode(talon_hud_widget_names, "font_size")
head up theme {user.talon_hud_themes}	user.hud_switch_theme(talon_hud_themes)

HUD WIDGET QUICK CHOICES

Input	Result
{user.talon_hud_quick_choices}	user.hud_activate_choice(talon_hud_quick_choices)

End of Talon Cheat Sheet