

 Personal Open source Business Explore Pricing Blog Support

This repository Search

Sign in

Sign up

1995eaton / chromium-vim

Watch35Star996Fork109

<> Code🕒 Issues180🔗 Pull requests9📁 Projects0📖 Wiki📈 Pulse📊 Graphs

Vim bindings for Google Chrome. <https://chrome.google.com/webstore/de...>

📦 992 commits


🌿 1 branch

📦 13 releases

👤 24 contributors

📄 MIT

Branch: masterNew pull requestFind fileClone or download

	1995eaton Add support for auto-invoked script blocks	Latest commit a686ba6 on Jul 26
background_scripts	fix #415	6 months ago
content_scripts	Add support for auto-invoked script blocks	5 months ago
cvimrc_parser	Add support for auto-invoked script blocks	5 months ago
icons	No commit message	3 years ago
pages	implement #451	6 months ago
scripts	update CSS	7 months ago
.eslintrc	jshint -> eslint + stylistic changes	8 months ago
.gitignore	rework build scripts	7 months ago
CHANGELOG.md	update to 1.2.86	6 months ago
LICENSE.txt	No commit message	3 years ago
Makefile	move pegjs dependency up one directory into package.json (#167)	2 years ago
README.md	implement #451	6 months ago
cmdline_frame.html	rework build scripts	7 months ago
cmdline_frame.js	iframe -- initial commit	2 years ago
cvim_server.py	listen on 127.0.0.1 rather than verify uuid token	2 years ago
manifest.json	update to 1.2.86	6 months ago
package.json	move from marked to marked-it + highlight.js for syntax highlighting	7 months ago

📖 README.md

## What is cVim?

Vim for Google Chrome. I hate using the mouse, especially after learning Vim. With my desktop (Linux), I have a lot of key bindings that make doing things easier: I open Chrome with `Alt+w`, I close a window with `Alt+Shift+d`, I open a terminal with `Alt+t`. This is harder to do with Chrome because it has no section for customizing keyboard shortcuts, and it is still necessary to use the mouse to do things like click links. cVim aims to eliminate this problem as best as the Chrome extensions API will allow it to.

## Where can I get cVim?

- There are two ways:
  - You can install it through the [Chrome web store](#)
  - You can download the `.zip` file [here](#) and enable cVim by going to the `chrome://extensions` URL and checking developer mode, then pointing Chrome to the unzipped folder via the `Load unpacked extensions...` button.

## Why is this different than Vimium, ViChrome, or Vrome?

These extensions do a wonderful job of adding Vim-like keybindings to Google Chrome, but they lack many of the features that Firefox Addon, Pentadactyl, have.

- What features does cVim add to Chrome?
  - Google/IMDB/Wikipedia/Amazon/Duckduckgo/Yahoo/Bing search completion
  - Support for custom search engines
  - History and Bookmark search/completion with bookmark folder support
  - Caret/Visual mode
  - Efficient link hints (with support for custom mappings)
  - Support for custom keyboard mappings
  - Regex page search with highlighting
  - Command bar with tab-completion
  - Smooth scrolling

## cVim Help

### cVimrc

- Boolean cVimrc settings are enabled with the command `'set' + <SETTING_NAME>` and disabled with the command `'set' + no<SETTING_NAME>` (for example, `set regexp` and `set noregexp` )
- Boolean cVimrc settings can be inversed by adding `"!"` to the end
- Other settings are defined with `=` used as a separator and are prefixed by `let` (for example, `let hintcharacters="abc"` )

setting	type	description	default
searchlimit	integer	set the amount of results displayed in the command bar	25
scrollstep	integer	set the amount of pixels scrolled when using the scrollUp and scrollDown commands	70
timeoutlen	integer	The amount of time to wait for a <code>&lt;Leader&gt;</code> mapping in milliseconds	1000
fullpagescrollpercent	integer	set the percent of the page to be scrolled by when using the scrollFullPageUp and scrollFullPageDown commands	0
typelinkhintsdelay	integer	the amount of time (in milliseconds) to wait before taking input after opening a link hint with typelinkhints and numerichints enabled	300
scrollduration	integer	the duration of smooth scrolling	500
vimport	integer	set the port to be used with the <code>editWithVim</code> insert mode command	8001
zoomfactor	integer / double	the step size when zooming the page in/out	0.1
scalehints	boolean	animate link hints as they appear	false
hud	boolean	show the heads-up-display	true
regexp	boolean	use regexp in find mode	true
ignorecase	boolean	ignore search case in find mode	true
linkanimations	boolean	show fade effect when link hints open and close	false

setting	type	description	default
numerichints	boolean	use numbers for link hints instead of a set of characters	false
dimhintcharacters	boolean	dim letter matches in hint characters rather than remove them from the hint	true
defaultnewtabpage	boolean	use the default chrome://newtab page instead of a blank page	false
cncpcompletion	boolean	use <C-n> and <C-p> to cycle through completion results (requires you to set the nextCompletionResult keybinding in the chrome://extensions page (bottom right))	false
smartcase	boolean	case-insensitive find mode searches except when input contains a capital letter	true
incsearch	boolean	begin auto-highlighting find mode matches when input length is greater than two characters	true
typelinkhints	boolean	(numerichints required) type text in the link to narrow down numeric hints	false
autohidecursor	boolean	hide the mouse cursor when scrolling (useful for Linux, which doesn't auto-hide the cursor on keydown)	false
autofocus	boolean	allows websites to automatically focus an input box when they are first loaded	true
insertmappings	boolean	use insert mappings to navigate the cursor in text boxes (see bindings below)	true
smoothscroll	boolean	use smooth scrolling	false
autoupdategist	boolean	if a GitHub Gist is used to sync settings, pull updates every hour (and when Chrome restarts)	false
nativelinkorder	boolean	Open new tabs like Chrome does rather than next to the currently opened tab	false
showtabindices	boolean	Display the tab index in the tab's title	false
sortlinkhints	boolean	Sort link hint lettering by the link's distance from the top-left corner of the page	false
localconfig	boolean	Read the cVimrc config from configpath (when this is set, you cannot save from cVim's options page)	false
completeonopen	boolean	Automatically show a list of command completions when the command bar is opened	false
configpath	string	Read the cVimrc from this local file when configpath is set	""
changelog	boolean	Auto open the changelog when cVim is updated	true
completionengines	array of strings	use only the specified search engines	["google", "duckduckgo", "wikipedia", "amazon"]
blacklists	array of strings	disable cVim on the sites matching one of the patterns	[]
mapleader	string	The default <Leader> key	\

setting	type	description	default
defaultengine	string	set the default search engine	"google"
locale	string	set the locale of the site being completed/searched on (see example configuration below)	""
homedirectory	string	the directory to replace ~ when using the file command	""
qmark <alphanumeric character>	string	add a persistent QuickMark (e.g. let qmark a = ["http://google.com", "http://reddit.com"] )	none
previousmatchpattern	string (regexp)	the pattern looked for when navigating a page's back button	((?!last)(prev(ious)?  newer back « less < <  )+)
nextmatchpattern	string (regexp)	the pattern looked for when navigation a page's next button	((?!first) (next older more > > » forward  )+)
hintcharacters	string (alphanumeric)	set the default characters to be used in link hint mode	"asdfgqwertyxcvb"
barposition	string ["top", "bottom"]	set the default position of the command bar	"top"
vimcommand	string	set the command to be issued with the editWithVim command	"gvim -f"
langmap	string	set a list of characters to be remapped (see vims langmap)	""

## Example configuration

```
" Settings
set nohud
set nosmoothscroll
set noautofocus " The opposite of autofocus; this setting stops
                  " sites from focusing on an input box when they load
set typelinkhints
let searchlimit = 30
let scrollstep = 70
let barposition = "bottom"

let locale = "uk" " Current choices are 'jp' and 'uk'. This allows cVim to use sites like google.co.uk
                  " or google.co.jp to search rather than google.com. Support is currently limited.
                  " Let me know if you need a different locale for one of the completion/search engines
let hintcharacters = "abc123"

let searchengine dogpile = "http://www.dogpile.com/search/web?q=%s " " If you leave out the '%s' at the en
                              " your query will be appended to the
                              " Otherwise, your query will replace

" This will do the same thing as above, except typing ':tabnew withbase' into to command bar
" without any search parameters will open 'http://www.dogpile.com'
let searchengine withbase = ["http://www.dogpile.com ", "http://www.dogpile.com/search/web?q=%s "]

" alias ':g' to ':tabnew google'
command g tabnew google

let completionengines = ["google", "amazon", "imdb", "dogpile"]

let searchalias g = "google" " Create a shortcut for search engines.
                              " For example, typing ':tabnew g example'
                              " would act the same way as ':tabnew google example'

" Open all of these in a tab with `gnb` or open one of these with <N>goa where <N>
let qmark a = ["http://www.reddit.com ", "http://www.google.com ", "http://twitter.com "]

let blacklists = ["https://mail.google.com/* ", "*/mail.google.com/* ", "@https://mail.google.com/mail/* "]
" blacklists prefixed by '@' act as a whitelist
```

```

let mapleader = ","

" Mappings

map <Leader>r reloadTabUncached
map <Leader>x :restore<Space>

" This remaps the default 'j' mapping
map j scrollUp

" You can use <Space>, which is interpreted as a
" literal " " character, to enter buffer completion mode
map gb :buffer<Space>

" This unmaps the default 'k' mapping
unmap k

" This unmaps the default 'h', 'j', 'k', and 'l' mappings
unmap h j k l

" This remaps the default 'f' mapping to the current 'F' mapping
map f F

" Toggle the current HUD display value
map <C-h> :set hud!<CR>

" Switch between alphabetical hint characters and numeric hints
map <C-i> :set numerichints!<CR>

map <C-u> rootFrame
map <M-h> previousTab
map <C-d> scrollPageDown
map <C-e> scrollPageUp
iunmap <C-y>
imap <C-m> deleteWord

" Create a variable that can be used/referenced in the command bar
let @@reddit_prog = 'http://www.reddit.com/r/programming '
let @@top_all = 'top?sort=top&t=all '
let @@top_day = 'top?sort=top&t=day '

" TA binding opens 'http://www.reddit.com/r/programming/top?sort=top&t=all' in a new tab
map TA :to @@reddit_prog/@@top_all<CR>
map TD :to @@reddit_prog/@@top_day<CR>

" Code blocks (see below for more info)
getIP() -> {{
  httpRequest({url: 'http://api.ipify.org/?format=json ', json: true},
    function(res) { Status.setMessage('IP: ' + res.ip); });
}}
" Displays your public IP address in the status bar
map ci :call getIP<CR>

" Script hints
echo(link) -> {{
  alert(link.href);
}}
map <C-f> createScriptHint(echo)

let configpath = '/path/to/your/.cvimrc '
set localconfig " Update settings via a local file (and the `:source` command) rather
                  " than the default options page in chrome
" As long as localconfig is set in the .cvimrc file. cvim will continue to read
" settings from there

```

## Blacklists

- The blacklists setting uses a custom implementation of Chrome's @match pattern guidelines. See [https://developer.chrome.com/extensions/match\\_patterns](https://developer.chrome.com/extensions/match_patterns) for a description of the syntax.

## Site-specific Configuration

- You can enable certain rc settings for sites using the blacklist match pattern as described above

```
" this will enable the config block below on the domain 'reddit.com'
site '*/**/.reddit.com/*' {
    unmap j
    unmap k
    set numerichints
}
```

## Running commands when a page loads

- In a similar fashion to the site-specific configuration described above, cVim can run commands when a page is loaded with the `call` keyword

```
" In this case, when pages with a file ending in '.js' are loaded,
" cVim will pin the tab and then scroll down
site '*/**/*.js' {
    call :pintab
    call scrollDown
}
```

## Mappings

- Normal mappings are defined with the following structure: `map <KEY> <MAPPING_NAME>`
- Insert mappings use the same structure, but use the command "imap" instead of "map"
- Control, meta, and alt can be used also:

```
<C-u> " Ctrl + u
<M-u> " Meta + u
<A-u> " Alt + u
```

- It is also possible to unmap default bindings with `unmap <KEY>` and insert bindings with `iunmap <KEY>`
- To unmap all default keybindings, use `unmapAll` . To unmap all default insert bindings, use `iunmapAll`

## Tabs

- Commands that open links ( `:tabnew` and `:open` ) have three different properties
  - `!` => Open in a new tab
  - `$` => Open in a new window
  - `|` => Open in an incognito window
  - `&` => Open in a new tab (inactive/unfocused)
  - `*` => Pin the tab
  - `?` => Treat the query as a search
  - `=` => Treat the query as a URL
- The use of these properties are best explained with examples:

```
:open! google<CR> " This is the same as :tabnew google<CR>

:open google!<CR> " This is another way of writing the above
                  " (these flags can be added to either
                  " the base command or the end of the final command)

:open& google<CR> " This will open Google in a new inactive tab

:open$ google<CR> " This will open Google in a new window

:open* google<CR> " The will open Google in a new inactive, pinned tab

:tabnew google*<CR> " Once again, this will do the same thing as the above command

:open google*<CR> " Again, same as above

:open google!& " Here, the & flag will cancel out the ! flag,
               " opening Google in a new inactive tab

" More examples
:bookmarks my_bookmark.com& " inactive,new tab
```

```
:bookmarks&* my_bookmark.com " inactive,pinned,new tab
:bookmarks! my_bookmark.com " new tab
:bookmarks$ my_bookmark.com " new window
:bookmarks my_bookmark.com " same tab
```

Code blocks

- Code blocks allow you to interact with cVim's content scripts via the cVimrc.
- Since code blocks use `eval(...)` , you should only use them if you know what you're doing.

```
" To be used by the code block
set hintset_a

" Create a code block named switchHintCharacters
switchHintCharacters -> {{
  // We are now in JavaScript mode

  // Settings are contained in an object named settings
  settings.hintset_a = !settings.hintset_a;
  if (settings.hintset_a) {
    settings.hintcharacters = 'abc'; // equivalent to "let hintcharacters = 'abc'"
  } else {
    settings.hintcharacters = 'xyz';
  }

  // Propagate the current settings to all tabs for the
  // rest of the session
  PORT('syncSettings', { settings: settings });

  // Display cVim's status bar for 2 seconds.
  Status.setMessage('Hint Set: ' + (true ? 'a' : 'b'), 2);
}}

" Run the JavaScript block
map <Tab> :call switchHintCharacters<CR>
```

Completion Engines

- These are a list of completion engines that can be used in the command bar. They can be set by assigning their names to an array with the `completionengines` variable.
  - google, wikipedia, youtube, imdb, amazon, google-maps, wolframalpha, google-image, ebay, webster, wictionary, urbandictionary, duckduckgo, answers, google-trends, google-finance, yahoo, bing, themoviedb
- Example usage:

```
let completionengines = ['google', 'google-image', 'youtube'] " Show only these engines in the command ba
```

Keybindings

Movement		Mapping name
j , s	scroll down	scrollDown
k , w	scroll up	scrollUp
h	scroll left	scrollLeft
l	scroll right	scrollRight
d	scroll half-page down	scrollPageDown
unmapped	scroll full-page down	scrollFullPageDown
u , e	scroll half-page up	scrollPageUp
unmapped	scroll full-page up	scrollFullPageUp
gg	scroll to the top of the page	scrollToTop

Movement		Mapping name
G	scroll to the bottom of the page	scrollToBottom
O	scroll to the left of the page	scrollToLeft
\$	scroll to the right of the page	scrollToRight
#	reset the scroll focus to the main page	resetScrollFocus
gi	go to first input box	goToInput
gI	go to the last focused input box by gi	goToLastInput
zz	center page to current search match (middle)	centerMatchH
zt	center page to current search match (top)	centerMatchT
zb	center page to current search match (bottom)	centerMatchB
<b>Link Hints</b>		
f	open link in current tab	createHint
F	open link in new tab	createTabbedHint
unmapped	open link in new tab (active)	createActiveTabbedHint
w	open link in new window	createHintWindow
A	repeat last hint command	openLastHint
q	trigger a hover event (mouseover + mouseenter)	createHoverHint
Q	trigger a unhover event (mouseout + mouseleave)	createUnhoverHint
mf	open multiple links	createMultiHint
unmapped	edit text with external editor	createEditHint
unmapped	call a code block with the link as the first argument	createScriptHint( <FUNCTION_NAME> )
unmapped	opens images in a new tab	fullImageHint
mr	reverse image search multiple links	multiReverseImage
my	yank multiple links (open the list of links with P)	multiYankUrl
gy	copy URL from link to clipboard	yankUrl
gr	reverse image search (google images)	reverseImage
;	change the link hint focus	
<b>QuickMarks</b>		
M<*>	create quickmark <*>	addQuickMark
go<*>	open quickmark <*> in the current tab	openQuickMark
gn<*>	open quickmark <*> in a new tab	openQuickMarkTabbed
gw<*>	open quickmark <*> in a new window	openQuickMarkWindowed
<b>Miscellaneous</b>		
a	alias to ":tabnew google "	:tabnew google
.	repeat the last command	repeatCommand
:	open command bar	openCommandBar
/	open search bar	openSearchBar
?	open search bar (reverse search)	openSearchBarReverse
unmapped	open link search bar (same as pressing /? )	openLinkSearchBar
I	search through browser history	:history



Movement		Mapping name
<N>g%	scroll <N> percent down the page	percentScroll
<N> unmapped	pass <N> keys through to the current page	passKeys
zr	restart Google Chrome	:chrome://restart<CR>
i	enter insert mode (escape to exit)	insertMode
r	reload the current tab	reloadTab
gR	reload the current tab + local cache	reloadTabUncached
; <*>	create mark <*>	setMark
' '	go to last scroll position	lastScrollPosition
' <*>	go to mark <*>	goToMark
cm	mute/unmute a tab	muteTab
none	reload all tabs	reloadAllTabs
cr	reload all tabs but current	reloadAllButCurrent
zi	zoom page in	zoomPageIn
zo	zoom page out	zoomPageOut
z0	zoom page to original size	zoomOrig
z<Enter>	toggle image zoom (same as clicking the image on image-only pages)	toggleImageZoom
gd	alias to :chrome://downloads<CR>	:chrome://downloads<CR>
ge	alias to :chrome://extensions<CR>	:chrome://extensions<CR>
yy	copy the URL of the current page to the clipboard	yankDocumentUrl
yY	copy the URL of the current frame to the clipboard	yankRootUrl
ya	copy the URLs in the current window	yankWindowUrls
yh	copy the currently matched text from find mode (if any)	yankHighlight
b	search through bookmarks	:bookmarks
p	open the clipboard selection	openPaste
P	open the clipboard selection in a new tab	openPasteTab
gj	hide the download shelf	hideDownloadsShelf
gf	cycle through iframes	nextFrame
gF	go to the root frame	rootFrame
gq	stop the current tab from loading	cancelWebRequest
gQ	stop all tabs from loading	cancelAllWebRequests
gu	go up one path in the URL	goUpUrl
gU	go to to the base URL	goToRootUrl
gs	go to the view-source:// page for the current Url	:viewsource!
<C-b>	create or toggle a bookmark for the current URL	createBookmark
unmapped	close all browser windows	quitChrome
g-	decrement the first number in the URL path (e.g www.example.com/5 => www.example.com/4 )	decrementURLPath
g+	increment the first number in the URL path	incrementURLPath

Movement		Mapping name
<b>Tab Navigation</b>		
gt , K , R	navigate to the next tab	nextTab
gT , J , E	navigate to the previous tab	previousTab
g0 , g\$	go to the first/last tab	firstTab, lastTab
<C-S-h> , gh	open the last URL in the current tab's history in a new tab	openLastLinkInTab
<C-S-l> , gl	open the next URL from the current tab's history in a new tab	openNextLinkInTab
x	close the current tab	closeTab
gxT	close the tab to the left of the current tab	closeTabLeft
gxt	close the tab to the right of the current tab	closeTabRight
gx0	close all tabs to the left of the current tab	closeTabsToLeft
gx\$	close all tabs to the right of the current tab	closeTabsToRight
X	open the last closed tab	lastClosedTab
t	:tabnew	:tabnew
T	:tabnew <CURRENT URL>	:tabnew @%
O	:open <CURRENT URL>	:open @%
<N>%	switch to tab <N>	goToTab
H , S	go back	goBack
L , D	go forward	goForward
B	search for another active tab	:buffer
<	move current tab left	moveTabLeft
>	move current tab right	moveTabRight
]]	click the "next" link on the page (see nextmatchpattern above)	nextMatchPattern
[[	click the "back" link on the page (see previousmatchpattern above)	previousMatchPattern
gp	pin/unpin the current tab	pinTab
<C-6>	toggle the focus between the last used tabs	lastUsedTab
<b>Find Mode</b>		
n	next search result	nextSearchResult
N	previous search result	previousSearchResult
v	enter visual/caret mode (highlight current search/selection)	toggleVisualMode
V	enter visual line mode from caret mode/currently highlighted search	toggleVisualLineMode
unmapped	clear search mode highlighting	clearSearchHighlight
<b>Visual/Caret Mode</b>		
<Esc>	exit visual mode to caret mode/exit caret mode to normal mode	
v	toggle between visual/caret mode	
h , j , k , l	move the caret position/extend the visual selection	

Movement		Mapping name
y	copy the current selection	
n	select the next search result	
N	select the previous search result	
p	open highlighted text in current tab	
P	open highlighted text in new tab	
<b>Text boxes</b>		
<C-i>	move cursor to the beginning of the line	beginningOfLine
<C-e>	move cursor to the end of the line	endOfLine
<C-u>	delete to the beginning of the line	deleteToBeginning
<C-o>	delete to the end of the line	deleteToEnd
<C-y>	delete back one word	deleteWord
<C-p>	delete forward one word	deleteForwardWord
unmapped	delete back one character	deleteChar
unmapped	delete forward one character	deleteForwardChar
<C-h>	move cursor back one word	backwardWord
<C-l>	move cursor forward one word	forwardWord
<C-f>	move cursor forward one letter	forwardChar
<C-b>	move cursor back one letter	backwardChar
<C-j>	move cursor forward one line	forwardLine
<C-k>	move cursor back one line	backwardLine
unmapped	select input text (equivalent to <C-a> )	selectAll
unmapped	edit with Vim in a terminal (need the <a href="#">cvim_server.py</a> script running for this to work)	editWithVim

## Command Mode

Command	Description
:tabnew (autocomplete)	open a new tab with the typed/completed search
:new (autocomplete)	open a new window with the typed/completed search
:open (autocomplete)	open the typed/completed URL/google search
:history (autocomplete)	search through browser history
:bookmarks (autocomplete)	search through bookmarks
:bookmarks /<folder> (autocomplete)	browse bookmarks by folder/open all bookmarks from folder
:set (autocomplete)	temporarily change a cVim setting
:chrome:// (autocomplete)	open a chrome:// URL
:tabhistory (autocomplete)	browse the different history states of the current tab

Command	Description
:command <NAME> <ACTION>	aliases : <NAME> to : <ACTION>
:quit	close the current tab
:qall	close the current window
:restore (autocomplete)	restore a previously closed tab (newer versions of Chrome only)
:tabattach (autocomplete)	move the current tab to another open window
:tabdetach	move the current tab to a new window
:file (autocomplete)	open a local file
:source (autocomplete)	load a cVimrc file into memory (this will overwrite the settings in the options page if the <code>localconfig</code> setting had been set previously)
:duplicate	duplicate the current tab
:settings	open the settings page
:nohlsearch	clear the highlighted text from the last search
:execute	execute a sequence of keys (Useful for mappings. For example, "map j :execute 2j")
:buffer (autocomplete)	change to a different tab
:mksession	create a new session from the current tabs in the active window
:delsession (autocomplete)	delete a saved session
:session (autocomplete)	open the tabs from a saved session in a new window
:script	run JavaScript on the current page
:togglepin	toggle the pin state of the current tab
:pintab	pin the current tab
:unpintab	unpin the current tab

## Tips

- You can use `@%` in "open" commands to specify the current URL. For example, `:open @%` would essentially refresh the current page.
- Prepend a number to the command to repeat that command N times
- Use the up/down arrows in command/find mode to navigate through previously executed commands/searches -- you can also use this to search for previously executed commands starting with a certain combination of letters (for example, entering `ta` in the command bar and pressing the up arrow will search command history for all matches beginning with `ta`)

## Contributing

Nice that you want to spend some time improving this extension. Solving issues is always appreciated. If you're going to add a feature, it would be best to [submit an issue](#). You'll get feedback whether it will likely be merged.

1. Run `npm install` in the repository's root folder
2. Run `make`
3. Navigate to `chrome://extensions`
4. Toggle into Developer Mode
5. Click on "Load Unpacked Extension..."
6. Select the cVim directory.

