perl-support.vim

HOT KEYS

Key mappings for Vim and gVim. Plugin: http://vim.sourceforge.net Fritz Mehner (mehner@fh-swf.de)

(i) insert mode, (n) normal mode, (v) visual mode

| Load / Unload Perl Support | | | | |
|----------------------------|-----------------------------------|--------------|--|--|
| \lps | load menues | (n) | | |
| \ups | unload menues | (n) | | |
| | | H elp | | |
| \hp | help (plugin) | (n,i) | | |
| | Cc | omments | | |
| \cl | end-of-line comment | (n, v, i) | | |
| \cj | adjust end-of-line comments | (n, v, i) | | |
| \cs | set end-of-line comment col. | (n) | | |
| \cfr | frame comment | (n, i) | | |
| \cfu | function description | (n, i) | | |
| \cm | method description | (n, i) | | |
| \chpl | file header (.pl) | (n) | | |
| \chpm | file header (.pm) | (n) | | |
| \cht | file header (.t) | (n) | | |
| \chpo | file header (.pod) | (n) | | |
| \ckb | keyword comm. BUG | (n, i) | | |
| \ckt | keyword comm. TODO | (n, i) | | |
| \ckr | keyword comm. TRICKY | (n, i) | | |
| \ckw | keyword comm. WARNING | (n, i) | | |
| \cko | keyword comm. WORKAROUND | (n, i) | | |
| \ckn | keyword comm. new keyword | (n, i) | | |
| \cc | $code \leftrightarrow comment$ | (n, v) | | |
| \cb | $code\ block \rightarrow comment$ | (n, v) | | |
| \cn | uncomment code block | (n) | | |
| \cd | date | (n, i) | | |
| \ct | date & time | (n, i) | | |
| \cv | vim modeline | (n, i) | | |

| \sd do { } while (n, v, i) \ \sf for { } (n, v, i) \ \sf for { } (n, v, i) \ \si for { } (n, v, i) \ \si if { } (n, v, i) \ \se else { } (n, v, i) \ \se else { } (n, v, i) \ \se unless { } (n, v, i) \ \su unless { } (n, v, i) \ \s | | | $oldsymbol{S} tatements$ |
|---|-------|----------------------|--------------------------|
| \sfe foreach { } (n, v, i) \ \si if { } (n, v, i) \ \sie if { } else { } (n, v, i) \ \se elsif { } (n, v, i) \ \su unless { } (n, v, i) \ \su unless { } else { } (n, v, i) \ \su until { } (n, v, i) \ \sw while { } (n, v, i) \ \st until { } (n, v, i) \ \sw while { } (n, v, i) \ \sw my \$; (n, i) \ \sw my \$; (n, i) \ \sw my \$ = ; (n, i) \ \w my \$ = (-,,); (| \sd | do { } while | (n, v, i) |
| \sfe foreach { } (n, v, i) \ \si if { } (n, v, i) \ \sie if { } else { } (n, v, i) \ \se elsif { } (n, v, i) \ \su unless { } (n, v, i) \ \su unless { } else { } (n, v, i) \ \su until { } (n, v, i) \ \sw while { } (n, v, i) \ \st until { } (n, v, i) \ \sw while { } (n, v, i) \ \sw my \$; (n, i) \ \sw my \$; (n, i) \ \sw my \$ = ; (n, i) \ \w my \$ = (-,,); (| \sf | for { } | (n, v, i) |
| \sie if { } else { } (n, v, i) \se else { } (n, v, i) \sei elsif { } (n, v, i) \su unless { } else { } (n, v, i) \su unless { } else { } (n, v, i) \st until { } (n, v, i) \sw while { } (n, i, i) \sw wy { } (n, i) \sw my { } (n, i) \sw my { } (n, i) \w my { } (n, i) | \sfe | | |
| \sie | \si | if { } | (n, v, i) |
| \sei elsif { } (n, v, i) \su unless { } (n, v, i) \sue unless { } else { } (n, v, i) \st until { } (n, v, i) \sw while { } (n, v, i) \s{ \sb { } (n, v, i) \st until { } (n, v, i) \sw while { } (n, i, i) \sw while { } (n, i, i) \sw wy \$ (n, i) \sw wy \$ (n, i) \w while { } (n, v, i) \w while { | \sie | if { } else { } | (n, v, i) |
| \su | ∖se | else { } | (n, v, i) |
| \sue \ unless { } else { } \ (n, v, i) \ \st \ until { } \ (n, v, i) \ (n, i) \ (| ∖sei | elsif { } | (n, v, i) |
| \st | \su | | |
| \sw while { } (n, v, i) \\s{ \sb { }} (n, v, i) \\ \s{ \sb { }} (n, v, i) \\ \s my \$; (n, i) \\ \\$ my \$; (n, i) \\ \\$ my \$ = ; (n, i) \\ \\$ my \$ = ; (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ my \$ = (n, i); (n, i) \\ \\$ in my \$ = (n, i); (n, i) \\ \\$ in my \$ = (n, i); (n, i) \\ \\$ in my \$ = (n, i); (n, i) \\ \\$ in my \$ = (n, i); (n, i) \\ \\$ in my \$ = (n, i); (n, i) \\ \\$ in my \$ = (n, i); (n, i) \\ \\$ in my \$ = (n, i); (n, i) \\ \\$ in my \$ = (n, i); (n, i) \\ \\$ in open input file (n, i) \\ \\$ in open output file (n, i) \\ \\$ in open open output file (n, i) \\ \\$ in open open output file (n, i) \\ \\$ in open open output file (n, i) \\ \\$ in open open open open (n, i) \\ \\$ in open open open open (n, i) \\ \\$ in open open open open (n, i) \\ \\$ in | \sue | | |
| \sb \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ | | until { } | (n, v, i) |
| Idioms | \sw | while { } | (n, v, i) |
| \\$ my \$; (n, i) \\$ my \$ = ; (n, i) \\$\$ my (\$, \$); (n, i) \\$ my @; (n, i) \\$ my @; (n, i) \\$ my @ = (,,); (n, i) \\$ my %; (n, i) \\$ my % = (=>,=>,); (n, i) \\$ my % = (=>,=>,); (n, i) \\$ ir my \$rgx_ = q//; (n, i) \\$ is \$ =~ m//xm (n, i) \\$ is \$ =~ m//xm (n, i) \\$ is \$ =~ tr//xm (n, i) \\$ is u subroutine (n, v, i) \\$ if u (n, v, i) \\$ ip print "\n"; (n, i) \\$ ii open input file (n, v, i) \\$ io open output file (n, v, i) \\$ io open output file (n, v, i) \\$ io open pipe (n, v, i) \\$ in read code snippet (n, v, i) \\$ nw write code snippet (n) \\$ nw edit code snippet (n) \\$ ntl edit local templates (n) \\$ ntg edit global templates (n) | \sb | { } | (n, v, i) |
| \\$= my \$ = ; (n, i) \\$\$ my (\$, \$); (n, i) \\@ my \@; (n, i) \\@ my \@; (n, i) \\\@ my \\@; (n, i) \\\ my \\\@; (n, i) \\\\ my \\\\\ my \\\\ = my \\\ = (=>,=>,); (n, i) \\\ in my \\\ rgx_= q//; (n, i) \\\ in \\ = m//xm (n, i) \\\\ in \\ = m//xm (n, i) \\\\\ in \\ = m//xm (n, i) \\\\\\\\ in \\ = m//xm (n, i) \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ | | | I dioms |
| \\$\$ my (\$, \$); | \\$ | my \$; | (n, i) |
| \(\(\) \(\ | \\$= | my \$ = ; | (n, i) |
| \@= my @ = (,,); (n, i) \% my %; (n, i) \%= my % = (=>,=>,); (n, i) \ir my \$rgx_ = q//; (n, i) \im \$ =~ m//xm (n, i) \is \$ =~ s//xm (n, i) \it \$ =~ tr//xm (n, i) \it \$ subroutine (n, v, i) \if (n, v | \\$\$ | my (\$, \$); | (n, i) |
| \% my %; (n, i) \%= my % = (=>,=>,); (n, i) \ir my \$rgx_ = q//; (n, i) \im \$ =~ m//xm (n, i) \is \$ =~ s//xm (n, i) \it \$ =~ tr//xm (n, i) \it \$ s=~ tr//xm (n, i) \it \$ subroutine (n, v, i) \if print "\n"; (n, i) \ii open input file (n, v, i) \io open output file (n, v, i) \io open output file (n, v, i) \io open output file (n, v, i) \in open input file (n, v, i) \in open open output file (n, v, i) \in open open open open open (n, v, i) \in open open open (n, v, i) \in open open open open open (n, v, i) \in open open open (n, v, i) \in open open op | \@ | | (n, i) |
| \%= my % = (=>,=>,); (n, i) \ir my \(\frac{\text{srgx}}{\text{srgx}} = \text{q}//; \) \im \(\frac{\text{srgx}}{\text{srgx}} = \text{q}//; \) \is \(\frac{\text{srgx}}{\text{srgx}} = \text{q}//; \) \it \(\frac{\text{srgx}}{\text{srgx}} = \text{q}//x \) \it \(\frac{\text{srgx}}{\text{srgx}} = \text{q}/x \) \it \(\frac{\text{srgx}}{\text{srgx}} = \text{q}/x \) \ | \@= | my @ = (,,); | (n, i) |
| \ir my \$rgx_ = q//; (n, i) \im \$ =~ m//xm (n, i) \is \$ =~ s///xm (n, i) \it \$ =~ tr///xm (n, i) \it \$ =~ tr///xm (n, i) \is subroutine (n, v, i) \if (n, v, i) \ip print "\n"; (n, i) \ii open input file (n, v, i) \io open output file (n, v, i) \io open pipe (n, v, i) \in print read code snippet (n, v, i) \in read code snippet (n) \nw write code snippet (n, v) \ne edit code snippet (n) \nt edit local templates (n) \ntg edit global templates (n) | | | · · / |
| \im \text{s} = \pi m / \text{xm} (n, i) \text{is} \text{s} = \pi s / / \text{xm} (n, i) \text{it} \text{subroutine} (n, v, i) \text{ifu} (n, v, i) \text{ifu} \text{open input file} (n, v, i) \text{ii} \text{open open output file} (n, v, i) \text{ipi} \text{open pipe} (n, v, i) \text{snippet} \text{nr} \text{read code snippet} (n, v) \text{nw} \text{write code snippet} (n, v) \text{ne} \text{edit code snippet} (n) \text{ntl} \text{edit global templates} (n) \text{ntg} \text{edit global templates} (n) \text{ntg} \text{edit global templates} (n) \text{ntg} \text{edit global templates} (n) \text{ntg} \text{edit global templates} (n) \text{ntg} \text{edit global templates} (n) \text{edit global templates} (n) | | | (n, i) |
| \is \$ =~ s///xm | | | (n, i) |
| \it \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ | | $=\sim m//xm$ | (n, i) |
| \isu subroutine (n, v, i) \ifu (n, v, i) \ifu (n, v, i) \ip print "\n"; (n, i) \ii open input file (n, v, i) \io open output file (n, v, i) \ip open pipe (n, v, i) \text{Snippet} \nr read code snippet (n, v, i) \nw write code snippet (n, v, i) \ne edit code snippet (n, v, i) | | | (n, i) |
| \ifu | | \$ =~ tr///xm | (n, i) |
| \ip print "\n"; (n ,i) \ii open input file (n, v, i) \io open output file (n, v, i) \ip open pipe (n, v, i) \ip Snippet \nr read code snippet (n) \nw write code snippet (n, v) \ne edit code snippet (n) \nt edit local templates (n) \ntg edit global templates (n) | | subroutine | |
| $\begin{tabular}{lllllllllllllllllllllllllllllllllll$ | | | |
| $\begin{tabular}{ c c c c c } \hline \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $ | | | |
| $\begin{tabular}{c c c c c c c c c c c c c c c c c c c $ | | | |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ | | | |
| $\begin{array}{c ccccccccccccccccccccccccccccccccccc$ | \ipi | open pipe | |
| $\begin{array}{cccc} \mbox{\sc nw} & \mbox{\sc write code snippet} & (n, v) \\ \mbox{\sc ne} & \mbox{\sc edit code snippet} & (n) \\ \mbox{\sc ne} & \mbox{\sc code snippet} & (n) \\ \mbox{\sc ne} & \mbox{\sc code snippet} & (n) \\ \mbox{\sc ne} & \mbox{\sc code snippet} & (n) \\ \mbox{\sc ne} & \mbox{\sc code snippet} & (n) \\ $ | | | S n $ippet$ |
| \ne edit code snippet (n) \nt1 edit local templates (n) \ntg edit global templates (n) | \nr | | (n) |
| \ntl edit local templates (n) \ntg edit global templates (n) | ∖nw | | (n, v) |
| \ntg edit global templates (n) | | | (n) |
| | | | (n) |
| \ntr reread the templates (n) | | | (n) |
| | \ntr | reread the templates | (n) |

| | D 1 E | |
|------------------|--------------------------------|-----------------|
| | Regular Exp. | |
| \xr | pick up Regex | (n, v) |
| ∖xs | pick up string | (n, v) |
| \xf | pick up flag(s) | (n, v) |
| \xm | match | (n) |
| \xmm | match multiple (Regex/target) |) (n) |
| \xe | explain Regex | (n, v) |
| | POSIX Character | Classes |
| \pa | [:alnum:] | (n, i) |
| \ph | [:alpha:] | (n, i) |
| \pi | [:ascii:] | (n, i) |
| \pb | [:blank:] | (n, i) |
| \pc | [:cntrl:] | (n, i) |
| \pd | [:digit:] | (n, i) |
| \pg | [:graph:] | (n, i) |
| \pl | [:lower:] | (n, i) |
| \pp | [:print:] | (n, i) |
| \pn | [:punct:] | (n, i) |
| \ps | [:space:] | (n, i) |
| \pu | [:upper:] | (n, i) |
| \pw | [:word:] | (n, i) |
| \px | [:xdigit:] | (n, i) |
| | | \mathbf{R} un |
| \rr | update file, run script | (n) |
| \rs | update file, check syntax | (n) |
| \ra | set command line arguments | (n) |
| \rw | set Perl cmd. line switches | (n) |
| \rd | start debugger | (n) |
| \re | make script executable | (n) |
| \rp \h | read perldoc for word under cu | \ / |
| \ri | show installed Perl modules | (n) |
| \rg | generate Perl module list | (n) |
| \ry | run perltidy | (n, v) |
| \rps | run Devel::SmallProf | (n) |
| \rpf | run Devel::FastProf | (n) |
| \rpn | run Devel::NYTProf | (n) |
| \rc | run perlcritic | (n) |
| \rt | save buffer with timestamp | (n) |
| \rh | hardcopy buffer | (n, v) |
| \rk | settings and hotkeys | (n) |
| \rx | | GUI only) |
| \ro | change output destination | (n) |
| \ - - | | () |

perlcritic

Ex commands for perlcritic (version 1.01+) Use tab expansion to choose the severity or the verbosity.

:CriticSeverity 1 2 3 4 5 brutal cruel harsh stern gentle

:CriticVerbosity1...11

:CriticOptions option(s), see perlcritic(1)

Profiling

The following ex commands can be used to sort a profiler report in the quickfix window.

Use tab expansion to choose the sort criterion or the file name.

For Devel::SmallProf

:SmallProfSort file-name|line-number|line-count|time|ctime

For Devel::FastProf

:FastProfSort file-name|line-number|time|line-count

For Devel::NYTProf

:NYTProfCSV Read a CSV-file.

:NYTProfHTML Read the HTML-reports with an external viewer (GUI only).

 $: NYTProfSort \qquad \textit{file-name} \, | \, line-number \, | \, time \, | \, calls \, | \, time-call$