























































Emacs support for Perl

Description	Keystroke	Function	Note
Perl Editing	<p>Emacs provides two major modes for Perl: perl-mode (Emacs default, simpler mode) and cperl-mode.</p> <p> The HaraldJoerg/cperl-mode external package has support for new Perl language features.</p> <p> PEL activates Perl support with the pel-use-perl user-options. When turned on:</p> <ul style="list-style-type: none"> • PEL provides supports all of them: pel-perl-mode selects the mode, using  HaraldJoerg/cperl-mode by default: it best supports Perl and perltidy. PEL installs the upstream copy of perl-mode from the HaraldJoerg/cperl-mode repo (which is what Emacs 30 uses) and the master copy of perl-tidy-ediff. • After using HaraldJoerg/cperl-mode if you want to revert to Emacs' own cperl-mode, remove the cperl-mode.el" and perl-tidy-ediff.el" files from ~/.emacs.d/utills • PEL activates minor modes for the perl major mode as specified by the pel-perl-activates-minor-modes user-option. <p> Speedbar support: If pel-use-speedbar is on, speedbar support for Perl is activated.</p> <p> Perl  Indentation control: for both perl-mode and cperl-mode is controlled by the following user-options:</p> <ul style="list-style-type: none"> • pel-general-perl-indent-level: it defaults to 4. It is applied to perl-indent-level and cperl-indent-level as well as tab-width. <p>The use of hard tabs is controlled by:</p> <ul style="list-style-type: none"> • pel-general-perl-use-tabs: it defaults to nil, forcing use of spaces. It is applied to indent-tabs-mode. 		
PEL Perl support improvements	<ul style="list-style-type: none"> • By default cperl-mode shows trailing whitespaces as underscores. Set pel-cperl-show-trailing-whitespace-normally user-option to t to use the trailing-whitespace face instead; by default this is a red background whitespace. • PEL improves iedit support for cperl-mode when pel-use-iedit is on. See  Search/Replace. • pel-open-at-point can find Perl files in Perl directories. It supports the Perl package :: and ' syntax. • Integrated buffer-only perltidy commands from HaraldJoerg/cperl-mode perl-tidy-ediff with improved command interface (2 commands do everything instead of 3). • pel-perl-critic executes Perl Critic over Perl code and provides navigation between compilation-mode error message and code. <p> perl-live-coding external package supported  when pel-use-pel-live-coding user-option is set.</p>		
Last updated on:	2025-03-17	See: Perl 5 Syntax	
Open this PDF file. See also:  Help/Info	<div><f11> SPC P <f1></div> <div><f12> <f1></div>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the  - Perl local PDF. With prefix argument (like C-u or M--) opens the remote GitHub hosted raw PDF instead. If the pel-flip-help-pdf-arg user-option is set it's the other way around.
Open Perl 5 PDF	<div><f11> p per15</div>	(pel-help-pdf-select &optional OPEN-GITHUB-PAGE-P)	Open the Perl 5 local PDF. With prefix argument (like C-u or M--) opens the remote GitHub hosted raw PDF instead. If the pel-flip-help-pdf-arg user-option is set it's the other way around.
 Customize PEL Perl support	<div><f11> SPC P <f2></div> <div><f12> <f2></div>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL Perl support. <ul style="list-style-type: none"> • If OTHER-WINDOW is non-nil (use C-u), display in another window.
 Customize Emacs Perl support	<div><f11> SPC P <f3></div> <div><f12> <f3></div>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs Perl support: perl, cperl, electricity, perl-repl, perl-live. <ul style="list-style-type: none"> • If OTHER-WINDOW is non-nil (use C-u), display in another window.
Show perl-mode status	<div><f12> <f4> ?</div>	(pel-perl-show-status &optional APPEND)	Show current buffer 'cperl-mode' status in specialized *pel-perl-info* buffer. <ul style="list-style-type: none"> • Clear previous buffer content (and move point to top of buffer) unless optional APPEND argument is non-nil,in which case it appends to the previous report and move point to the end of buffer..
Select perl-mode for extension-less file  The <f12> key is available only until a PEL controlled major mode is activated. Then it becomes a buffer prefix key.	<div><f12></div>	(pel-as &optional FORCE)	Inside a fundamental-mode buffer, interactively select major mode for the buffer. Re-do it with arg.
PCRE support	See PCRE support in  Search/Replace for commands to activate Perl Compatible regular expression search and operations.		
Perl Doc	<p>With Perl info and man files installed, Emacs  Help/Info man support provides access to the Perl documentation available on your system.</p> <p> Inside Emacs you can use man instead of perldoc: for example, man perlintro provides the same information as given by the perldoc perlintro command.</p> <div> <div> <ul style="list-style-type: none"> • Overview • Tutorials • Reference Manual </div> <div> <ul style="list-style-type: none"> • Operators • Functions • Variables </div> <div> <ul style="list-style-type: none"> • Modules • Utilities </div> <div> <ul style="list-style-type: none"> • History • Internals and C Language Interface • Language-specific </div> <div> <ul style="list-style-type: none"> • Miscellaneous • Platform-specific </div> </div>		
Show perldoc info switches <ul style="list-style-type: none"> • -f : info on a specific built-in function: -f split • -l (ell): print file location, as in: -l Pod::Simple • -m :to see the raw source, as in: -m Pod::Simple • -q switch to get FAQ info on a topic: as in -q random 	<div> <ul style="list-style-type: none"> • C-c C-h F • C-c C-h f • C-c C-h p • <f12> h </div> <div> C-c C-h F </div> <div> C-c C-h f </div> <div> C-c C-h p </div>	<div> (cperl-perldoc WORD) </div> <div> (cperl-info-on-command COMMAND) </div> <div> (cperl-info-on-current-command) </div> <div> (cperl-perldoc WORD &optional SECTION) </div>	<div> Prompt and show documentation about Perl item: run perldoc on WORD. <ul style="list-style-type: none"> • You can specify perldoc options if needed (see in title column). • Specify any module, as in Pod::Simple, or documents, as in: perldoc, perltoc, perlsyn, perlfunc </div> <div> Show documentation for Perl command COMMAND in other window. Obsolete since Emacs 30.1 <ul style="list-style-type: none"> • If perl-info buffer shown in some frame, uses this frame. Perl info pages are no longer distributed. • Customized by setting variables 'cperl-shrink-wrap-info-frame', 'cperl-max-help-size'. </div> <div> Show documentation for Perl command at point in other window. Obsolete since Emacs 30.1 </div> <div> Prompt for for (default to word at point). Show information about the selected word with 'perldoc'. Obsolete since Emacs 30.1 </div>
perldoc at point	<div> <ul style="list-style-type: none"> • C-c C-h P • <f12> H </div>	(cperl-perldoc-at-point)	Run a 'perldoc' on the word around point to show information about that Perl word.
Help on symbol at point	<div>C-c C-h v</div>	(cperl-get-help)	Get one-line docs on the symbol at the point. <ul style="list-style-type: none"> • The data for these docs is a little bit obsolete and may be in fact longer than a line.
Open file at point (for Perl)  File mngt  Tramp aware 	<p>The following command opens the file identified by the text taken at point (the cursor location). It supports extracting Perl package file names from the Perl package name syntax which uses either :: or ' as path delimiter.</p> <ul style="list-style-type: none"> • It searches the Perl package directories identified by Perl's @INC array and the directories identified by pel-perl-extra-project-root-directories user-option. <p> Note that when using the Ido completion mode, it is possible to instruct Ido to use a file name at point as the basis for the file name to open. This Ido behaviour is controlled by the ido-use-filename-at-point user-option. With PEL you can control it globally or locally with <f11> f M-.</p>		
Show searched directory trees	<div><f12> <f4> .</div>	(pel-perl-show-source-directories)	Display the list of source directories searched by pel-open-at-point. The list is controlled by Perl's @INC array and the extra list provided by pel-perl-source-directories user-option.
Open file or web-page whose name is at point ★★ Command is also specialized for: <ul style="list-style-type: none"> •  reStructuredText •  - C, Perl - C++ •  - Emacs Lisp •  - Erlang •  - UNIX Shell Generic Delimiting characters	<div> <ul style="list-style-type: none"> • M-<f6> • <f11> f . • 6v </div>	(pel-open-at-point &optional N)	<div> Open the file, library or the URL, named at point, with potential line & column #s. <ul style="list-style-type: none">  Supports glob characters, partial directory path. When multiple files are found it prompts using the method selected by pel-prompt-read-method user-option. <p> The 6v key-chord is available if pel-use-key-chord is non-nil. See  Key-Chords.</p> </div>
Select target window ▾	<p>This command works generically (for a normal file or directory name) and is specialized for Perl buffers: it finds Perl files in directories trees identified by Perl's @INC array. The command extracts the file or directory name, and possibly line and column numbers, from text at point and tries to open the file or directory.</p> <ul style="list-style-type: none"> • The generic mode extraction works by identifying the beginning & end of the file/directory/library/URL name string by delimiter characters, one of: tab, newline and and: " ; ` ' <pre>()[]{}<> '""'[] [] <> <> [] [] «»<><> () .</pre> <p> When finding several file names, the command lists them and prompts using the method selected by pel-prompt-read-method user-option.</p> <ul style="list-style-type: none"> • The default is a very primitive function implemented by PEL. You can select a more powerful ivy prompting instead. <ul style="list-style-type: none"> • With ivy selected, PEL will automatically set  pel-use-ivy to  and Ivy mode will be installed automatically when you restart Emacs. • Note that the command shows all files found by the specified search method, it does not only use the first one found. <ul style="list-style-type: none"> •  Use this to detect potential duplication in package and module files. <p>The command opens the file in the window selected by the following logic controlled by presence or absence of typed numerical prefix arguments:</p> <ul style="list-style-type: none"> • Select target window: <ul style="list-style-type: none"> • Without argument: <ul style="list-style-type: none"> • If file or directory is already opened in a window, move point to that window and to the line column coordinates if specified following the file name at point. • If no window holds that file, select the target window according to the number of editable windows in frame: if 1, split that window and use the new window, if 2: use the other window, if 3 or more, use the current window. • With prefix numeric argument N: <ul style="list-style-type: none"> • N < 0 : create a new window and use that. • (abs N) > 20: then open the directory instead of the file. Interpret the window position from the N value adjusted: N-20 (or N+20 if N is negative) • N = 0: use the '<i>other</i>' (the next) window. • N = 1, 3, 7or above (excluding 8, 9 and 10): select the target window based on the number of editable windows in frame: <ul style="list-style-type: none"> • if 1 window: split that window and use the new window, • if 2 windows: use the other window, • if 3 or more windows: use the current window. • N is: 8: up, 2: down, 4:left, 5:current, 6:right. • N is 9: force opening the file in the OS associated application (with N=29 or N=-29, open the file's directory with the OS associated application (eg. macOS Finder, Windows Explorer). If this is a URL, open it in the OS default web browser. • Selecting Minibuffer, inexistent or dedicated window is not allowed. 		
See function docstring for more info.			

Description	Keystroke		Function	Note
Comments	Perl comments start with #. See ℹ Comments for the generic commands that manage and control comments			
Toggle display of comments in buffer or active region	<f11> ; ;	(hide/show-comments-toggle &optional START END)	Toggle hiding/showing of comments in the active region or whole buffer. <ul style="list-style-type: none">If the region is active then toggle in the region. Otherwise, in the whole buffer.	
	 This requires the hide-comnt.el package (see ℹ Comments).  PEL activates it when the pel-use-hide-comnt user option is t .			
Generic code skeletons <ul style="list-style-type: none">tempo skeletons See also: <ul style="list-style-type: none">ℹ Inserting TextT Templates	Several mechanisms have been developed to allow easy insertion of predefined text in Emacs.  PEL does not yet define skeletons for Perl. You can use the generic one. <ul style="list-style-type: none">Emacs provides the built-in skeleton mechanism and the tempo skeletons.<ul style="list-style-type: none">PEL supports both. They are used a little bit differently. PEL provides generic tempo skeletons you can use for Perl until PEL adds Perl-specific skeletons.<ul style="list-style-type: none">PEL provides key bindings to the tempo skeletons: the generic code templates, accessible via the <f6> prefix key, and the language-specific code templates, accessible via the <f12> key prefix.			
ℹ Customize PEL Text Insertions control for Perl code skeletons .	<f6> <f2>	(pel-customize-pel &optional OTHER-WINDOW)	Open the customization groups that control the format of the various skeletons including the generic skeleton used by the <f6> h key and the <f12><f12> h key (see below). <ul style="list-style-type: none">If OTHER-WINDOW is non-nil (use C-u), display in other window.	
	<f12> <f12> <f2>	(pel-customize-generic-skels &optional OTHER-WINDOW)		
Insert generic file module header block — Language agnostic	<f6> h	(pel-generic-file-header)	Insert a file header block at the top of the file. Works only for buffer visiting a file.  The command key binding <f6> h is available only 1 second after Emacs has started.  As mentioned above PEL does not yet define Perl-specific skeletons, this uses the generic one.	
	<f12> <f12> h			
After inserting the template, navigate though areas that must be filled with: <ul style="list-style-type: none">tempo-forward-mark: C-c .tempo-backward-mark: C-c ,	 Specify the format of the header via the user-options in the pel-pkg-generic-code-style customization group accessible via <f6> <f2> <ul style="list-style-type: none">Inside a Perl buffer, <f12> <f2> provides access to the following customization groups:  After inserting a template, use tempo-forward-mark and tempo-backward-mark to move to the beginning of each section that must be filled.			
Toggle pel-tempo-mode	<f6> SPC	(pel-tempo-mode &optional ARG)	Toggle PEL tempo mode on/off.	
	<f12> <f12> SPC			
PEL tempo mode activates C-c . and C-c , as well as to C-c C- . and C-c C- , key bindings to navigate across tempo mark hot-spots. When pel-tempo-mode is active the pel-tempo-mode lighter (⚡) is shown on the status bar. The second set of keys are only available in graphics mode.  The pel-generic-file-header command inserts the text using a tempo skeleton: the PEL tempo mode is automatically activated by typing <f6> h .				
Expand any tag in template	<f6> <f12>	(tempo-complete-tag &optional SILENT)	Look for a tag and expand it. All the tags in the tag lists in 'tempo-local-tags' (this includes 'tempo-tags') are searched for a match for the text before the point. The way the string to match for is determined can be altered with the variable 'tempo-match-finder'. If 'tempo-match-finder' returns nil, then the results are the same as no match at all. <ul style="list-style-type: none">If a single match is found, the corresponding template is expanded in place of the matching string.If a partial completion or no match at all is found, and SILENT is non-nil, the function will give a signal.If a partial completion is found and 'tempo-show-completion-buffer' is non-nil, a buffer containing possible completions is displayed.	
Note: PEL default skeleton does not use tags.	<f12> <f12> <f12>			
Navigation Commands	Some navigation commands have special behaviour for Perl; they are shown below. They are available in perl-mode and cperl-mode . <ul style="list-style-type: none">There's also several generic navigation commands that work in Perl buffer too. See ℹ Navigation for those.			
Move to beginning of function	<ul style="list-style-type: none">C-M-a<f12> <up>	(perl-beginning-of-function &optional ARG)	Move backward to next beginning-of-function, or as far as possible. With argument, repeat that many times; negative args move forward.	
To end of function	<ul style="list-style-type: none">C-M-e<f12> <down>	(perl-end-of-function &optional ARG)	Move forward to next end-of-function. <ul style="list-style-type: none">The end of a function is found by moving forward from the beginning of one.With argument, repeat that many times; negative args move backward.	
Move to next interpolated	C-c C-v	(cperl-next-interpolated-REx &optional SKIP BEG LIMIT)	Move point to next REx which has interpolated parts. <ul style="list-style-type: none">SKIP is a list of possible types to skip, BEG and LIMIT are the starting point and the limit of search (default to point and end of buffer).SKIP may be a number, then it behaves as list of numbers up to SKIP; this semantic may be used as a numeric argument.Types are 0 for / \$rex /o (interpolated once), 1 for /\$rex/ (if \$rex is a result of qr//, this is not a performance hit), t for the rest	
	C-c C-x	(cperl-next-interpolated-REx-0)	Move point to next REx which has interpolated parts without //o.	
	C-c C-y	(cperl-next-interpolated-REx-1)	Move point to next REx which has interpolated parts without //o. <ul style="list-style-type: none">Skips REXes consisting of one interpolated variable.Note that skipped REXen are not performance hits.	
Marking Commands	The following marking commands are specialized for Perl. See ℹ Marking for the generic ones that can also be used in Perl buffers.			
Mark function	C-M-h	(perl-mark-function)	Put mark at end of Perl function, point at beginning	
Indentation Control	The following indentation control commands are specialized for Perl. See ℹ Indentation for the generic ones that can also be used in Perl. <ul style="list-style-type: none">The indentation behaviour of the <tab> key is controlled by the perl-mode and cperl-mode customization user-options. Use <f12> <f3> to open customization.			
Show currently used indentation style	<f12> <f4> s	(pel-perl-show-style)	Show the name of the currently used indentation style.	
Select new/restore old indentation style	<f12> <f4> <TAB>	(pel-perl-set-style)	Set Perl indentation style to named style. Prompt for indentation style name and apply it.  This change does not persist. Manually access the customized buffer and save it.	
Indent <ul style="list-style-type: none">perl-mode ➡	<tab>	(perl-indent-command &optional ARG)	Indent Perl code in the active region or current line. In Transient Mark mode, when the region is active, reindent the region. <ul style="list-style-type: none">Otherwise with a prefix argument, reindent the current line unconditionally.Otherwise if 'perl-tab-always-indent' is nil and point is not in the indentation area at the beginning of the line, insert a tab.Otherwise, indent the current line. If point was within the indentation area, it is moved to the end of the indentation area. If the line was already indented properly and point was not within the indentation area, and if 'perl-tab-to-comment' is non-nil (the default), then do the first possible action from the following list:<ol style="list-style-type: none">delete an empty commentmove forward to start of comment, indenting if necessarymove forward to end of linecreate an empty commentmove backward to start of comment, indenting if necessary.	
		(cperl-indent-command &optional WHOLE-EXP)	Indent current line as Perl code, or in some cases insert a tab character. <ul style="list-style-type: none">If 'cperl-tab-always-indent' is non-nil (the default), always indent current line. Otherwise, indent the current line only if point is at the left margin or in the line's indentation; otherwise insert a tab.A numeric argument, regardless of its value, means indent rigidly all the lines of the expression starting after point so that this line becomes properly indented. The relative indentation among the lines of the expression are preserved.	
Indent continued expression <ul style="list-style-type: none">cperl-mode ➡	C-M-q	(perl-indent-exp)	Indent each line of the Perl grouping following point.	
		(cperl-indent-exp)	Simple variant of indentation of continued-sexp. <ul style="list-style-type: none">Won't indent comment if it starts at 'comment-indent' or looks like continuation of the comment on the previous line.If 'cperl-indent-region-fix-constructs', will improve spacing on conditional/loop constructs.	
Indent region	C-M-\	(cperl-indent-region START END)	Indents all the lines whose first character is between START and END inclusive. <ul style="list-style-type: none">Won't indent comment if it starts at 'comment-indent' or looks like continuation of the comment on the previous line.If 'cperl-indent-region-fix-constructs', will improve spacing on conditional/loop constructs.	
Newline and indent <ul style="list-style-type: none">cperl-mode ➡	C-j	(newline-and-indent)	Insert a newline at point, then indent the newly created line. Use it to split a line. <ul style="list-style-type: none">Indentation is done using the value of 'indent-line-function' which is set to cperl-indent-line.	
Insert Perl new line	C-c C-j	(cperl-linefeed)	Go to end of line, open a new line and indent appropriately. If in POD, insert appropriate lines.	
	It's a convenience replacement for typing return . It puts point in the next line with proper indentation, or if you type it inside the inline block of control construct, like <pre>foreach (@lines) {print; print}</pre> and you are on a boundary of a statement inside braces, it will transform the construct into a multiline and will place you into an appropriately indented blank line. <ul style="list-style-type: none">Use C-j for usual 'newline-and-indent' behavior. See 'cperl-electric-linefeed' documentation.			

Description	Keystroke	Function	Note
Insert matching parens <ul style="list-style-type: none">toggle with C-c C-e	<ul style="list-style-type: none">(<[{	(cperl-electric-paren ARG)	Insert an opening parenthesis or a matching pair of parentheses. <ul style="list-style-type: none">Controlled by 'cperl-electric-parens' and 'cperl-hairy' user-options.
	<ul style="list-style-type: none">)]	(cperl-electric-rparen ARG)	Insert a matching pair of parentheses if marking is active. <ul style="list-style-type: none">If not, or if we are not at the end of marking range, would self-insert.Controlled by 'cperl-electric-parens' and 'cperl-hairy' user-options.
Insert : and indent	:	(cperl-electric-terminator ARG)	Insert character and correct line's indentation.
Insert ; and indent	;	(cperl-electric-semi ARG)	Insert character and correct line's indentation.
Insert { and indent	{	(cperl-electric-lbrace ARG &optional END)	Insert character, correct line's indentation, correct quoting by space.
Insert } and indent	}	(cperl-electric-brace ARG &optional ONLY-BEFORE)	Insert character and correct line's indentation. <ul style="list-style-type: none">If ONLY-BEFORE and 'cperl-auto-newline', will insert newline before the place (even in empty line), but not after. If after ")" and the inserted char is "{", insert extra newline before only if 'cperl-extra-newline-before-brace'.
Deleted and possibly untabify	DEL	(cperl-electric-backspace ARG)	Backspace, or remove whitespace around the point inserted by an electric key. <ul style="list-style-type: none">Will untabify if 'cperl-electric-backspace-untabify' is non-nil.
cperl-mode	PEL supports 2 cperl-mode implementations: Emacs' cperl-mode and  HaraldJoerg/cperl-mode (more complete) activated by the pel-use-perl user-option. <ul style="list-style-type: none">The following cperl commands toggle some of its electric behaviour.		
toggle Perl auto-help	C-c C-h a	(cperl-toggle-autohelp)	Toggle the state of Auto-Help on Perl constructs (put in the message area). <ul style="list-style-type: none">Delay of auto-help controlled by 'cperl-lazy-help-time':  By default it is nil, which (like null) prevents activation of the auto-help. To activate it, set cperl-lazy-help-time to an integer value.
Toggle auto-newline	C-c C-a	(cperl-toggle-auto-newline)	Toggle the state of 'cperl-auto-newline'.
Toggle electric mode	C-c C-e	(cperl-toggle-electric)	Toggle the state of parentheses doubling in CPerl mode. When typing an opening parens character the closing one is automatically entered.
Toggle auto-fill mode	C-c C-f	(auto-fill-mode &optional ARG)	Toggle automatic line breaking (Auto Fill mode). <ul style="list-style-type: none">With a prefix argument, enable Auto Fill mode if the prefix argument is positive, disable it otherwise.When Auto Fill mode is enabled, inserting a space at a column beyond 'current-fill-column' automatically breaks the line at a previous space.
	<ul style="list-style-type: none"><f11> t f a<f11> RET		
Toggle keyword expansion	C-c C-k	(cperl-toggle-abbrev)	Toggle the state of automatic keyword expansion in CPerl mode.
Toggle space fix	C-c C-w	(cperl-toggle-construct-fix)	Toggle whether 'indent-region'/'indent-sexp' fix whitespace too.
here-doc support	The following cperl-mode commands operate on Perl here documents .		
Narrows to here-doc	C-c C-n	(cperl-narrow-to-here-doc &optional POS)	Narrows editing region to the HERE-DOC at POS. POS defaults to the point.
Spell-check here-docs	C-c C-d	(cperl-here-doc-spell)	Spell-check HERE-documents in the Perl buffer. <ul style="list-style-type: none">If a region is highlighted, restricts to the region.
Spell-check POD documentation	C-c C-p	(cperl-pod-spell &optional DO-HERES)	Spell-check POD documentation. <ul style="list-style-type: none">If invoked with prefix argument, will do HERE-DOCs instead.If a region is highlighted, restricts to the region.
Verify & refactor Perl code	The following commands provide Perl verification and refactoring facilities.		
<ul style="list-style-type: none">perltidy  currently requires a lperltidy to be present on local host event when processing aremote file.	The following 2 commands run perltidy between areas of the Perl code in the current buffer and the tidied version of that code. All work is done inside Emacs buffers, no file is affected. After generating the tidied code the functions open an ediff session to compare your code and the tidied code, allowing you to decide what to use. Requires  HaraldJoerg/cperl-mode activated by the pel-use-perl user-option.  The perl-tidy-ediff commands execute the perltidy identified by perl-tidy-command user-option with options identified by the perl-tidy-ediff-args user-option. <ul style="list-style-type: none">See Perl::Tidy @metacpan for the perltidy list of options, also Online PerlTidy , which provides help on the various options categorized by feature set.		
Run perltidy on current buffer or region. Ediff changes.	<f12> T	(pel-perl-tidy-ediff)	Run perltidy on current buffer, than start an ediff session, comparing the original source with perltidy output. Error messages are saved in the "perl-tidy-errors" buffer. <ul style="list-style-type: none">If an area is marked, run perltidy on the marked area only.
Run perltidy on current subroutine. Ediff changes. See ℳ Narrowing	<f12> t	(perl-tidy-ediff-sub)	Run the perltidy program on the subroutine that stats before point and start en ediff session to compare the original code with the tidied version. <ul style="list-style-type: none">Error messages are saved in the "perl-tidy-errors" buffer. The buffer is automatically narrowed to the current function.When quitting the ediff session it remains narrowed. Use C-x n w to widen the buffer back.
<ul style="list-style-type: none">perlcritic	Perform static code analysis with perlcritic (which must be installed separately).		
Check code with perlcritic  ℳ Tramp -aware : run remote host's perlcritic on remote file.	<f12> c	(pel-perl-critic &optional VERBOSE)	Validate the Perl file visited in current buffer with perlcritic . Report error if it's not installed. <ul style="list-style-type: none">With optional VERBOSE prefix argument, print extra information:<ul style="list-style-type: none">Full name of the Policy module that created the violationFull diagnostic discussion of each Perl Best Practice (PBP) violation. Show errors in compilation-mode buffer in a format that allows navigation.
<ul style="list-style-type: none">Other refactoring			
Find/fix missing whitespace code	C-c C-b	(cperl-find-bad-style)	Find places in the buffer where insertion of a whitespace may help. <ul style="list-style-type: none">Prompts user for insertion of spaces. Currently it is tuned to C and Perl syntax.
Refactor: if (A) {B}' → 'B if A;'	C-c C-t	(cperl-invert-if-unless)	Change 'if (A) {B}' into 'B if A;' etc (or visa versa) if possible. <ul style="list-style-type: none">If the cursor is not on the leading keyword of the BLOCK flavor of construct, will assume it is the STATEMENT flavor, so will try to find the appropriate statement modifier.
Lining up Perl Code : ℳ Align	It's often best to line up Perl code vertically , arranging elements as tables, to help reader understand the code intent visually. <ul style="list-style-type: none">This technique is also promoted in Damian Conway's Perl Best Practice book, Vertical Alignment section of chapter 2. See pel-align-words-vertically in ℳ Align.		
Lineup	<ul style="list-style-type: none">C-M- <f12> 	(cperl-lineup BEG END &optional STEP MINSHIFT)	Lineup construction in a region. <ul style="list-style-type: none">Beginning of region should be at the start of a construction.All first occurrences of this construction in the lines that are partially contained in the region are lined up at the same column.MINSHIFT is the minimal amount of space to insert before the construction.STEP is the tabwidth to position constructions.If STEP is nil, 'cperl-lineup-step' will be used (or 'cperl-indent-level', if 'cperl-lineup-step' is nil).Will not move the position at the start to the left.
Testing Perl code	The following command provides a way to test Perl code locally. See Perl 5 Syntax for web-sites that provide a Perl interpreter.		
Start Perl REPL See: ℳ start Shells/REPLs	<f11> z r P <f12> z	(perl-repl)	Run a Perl REPL in a "Perl-REPL" buffer. <ul style="list-style-type: none"> Requires the perl-repl external package  activated by perl-use-perl-repl user-option.The perl-repl-file-path user option specifies the name of the Perl REPL program, which may optionally specify the explicit file path.PEL provides the perl-repl shell script which uses the Perl command line.
Perl Live Coding ★★	 perl-live-coding external package supported  when pel-use-pel-live-coding user-option is set. The "perl-live" buffer provides a much more powerful Perl interpreter.		
Run perl code in *perl-live*	<f12> 1	(pel-perl-live-run)	Start or open an existing perl-live session buffer. Invokes perl-live-run and open the "perl-live" buffer automatically, a comint mode buffer.  The <f12> 1 key binding is always bound in a perl-mode buffer. This is not the case for the following key bindings that become accessible once perl-live-run was first executed (via this command or via M-x perl-live-run)
The *perl-live* buffer operates in comint-mode which provides a set of commands to navigate across lines and history, as well as other. Type <f1> m inside the "perl-live" buffer to see these commands.	C-c C-1	(perl-live-run)	
Stop	C-c C-p	(perl-live-stop)	Stop perl-live session. The "perl-live" buffer gets disconnected from the Perl process. <ul style="list-style-type: none">It can be restarted (without closing the buffer) with a new execution of perl-live-run.
Evaluate Perl code line/region	C-c C-c	(perl-live-eval-region-or-line)	Evaluate a line or region of Perl code. <ul style="list-style-type: none">When evaluating a single line, point moves to the next line.
Evaluate Perl code in block	C-M-x	(perl-live-eval-sexp)	Evaluate Perl code between braces

Emacs & Perl — References	
Document	Notes
Also see: Perl 5 Syntax	
Perl @ Wikipedia	
perl.org	
Learn Perl @ perl.org • Perl Style Guide	<ul style="list-style-type: none"> • Perl Tutorial - a gentle introduction to Perl with several examples over a browsable set of pages. • Perl Intro - a quick introduction to Perl • Online Perl books <ul style="list-style-type: none"> • Beginning Perl
Perl Reference Manuals	<ul style="list-style-type: none"> • Perl Keywords. • Perl Operators. Also see the Perl ABC Operator page: organizes the information in sections (but has some markup typos). • Perl Functions
Is Perl still relevant? Most probably.	<ul style="list-style-type: none"> • What makes Perl relevant in 2022? @ Stackoverflow blog. • Perl is dying quick. Could be extinct by 2023. The HFT Guy. 2019. (which includes several invalid points). <ul style="list-style-type: none"> • My point is that Perl was popular, there's a lot of Perl code still being used and it's worth knowing and being able to write and edit Perl code (which was certainly not the first programming language as state by the article). But anyway, the post represents a point of view (and has many people commenting on it). • Perl is making a comeback: 5 reasons why it's worth learning. Posted January 6, 2023 by Lucas Rees. • Why Perl Didn't Win on outspeaking.com. updated December 21, 2020
Perl and Secure Coding Practice and Tools	<ul style="list-style-type: none"> • Security Issues in Perl Scripts - by Jordan Dimov - Discussion of misused and overlooked features of Perl from the security point of view. • SEI CERT Perl Coding Standard @ Carnegie Mellon University <ul style="list-style-type: none"> • The CERT Perl Secure Coding Standard, by David Svoboda . June 25, 2012 • perlsec describes Perl security. Tools: <ul style="list-style-type: none"> • The perlcritic script uses Perl::Critic to scan Perl code and provides some listing advices. <ul style="list-style-type: none"> • On some Linux distros you can install it with: <code>sudo dnf install perl-Perl-Critic</code> • The perltidy application reformats Perl code to the promoted format. • The zarn static analyzer, hosted on Github, is another static analyzer for Perl. (Quite immature as of Oct. 2024).
Perl File name extensions	<ul style="list-style-type: none"> • .pl Perl non executable libraries, also used for Perl scripts (but a file with no extension and a shebang line is also fine, and preferable, allowing the invocation of the script without having to type the '.pl'). • .pm Perl modules. • .plx Used by Active State implementation. Identifies executable Perl scripts. Also used elsewhere to distinguish from Prolog (which also uses the .pl file extension). • .pls • .xs • .t • .pod Plain Old Documentation files, a lightweight markup language used to document Perl code. <ul style="list-style-type: none"> • See also perlpod. • .ph • .cgi • .psgi
Perl programs	<ul style="list-style-type: none"> • Perl command line options
<ul style="list-style-type: none"> • perl 	Perl language interpreter
<ul style="list-style-type: none"> • perlbug / perlthanks 	Describes how to submit bug report on Perl
<ul style="list-style-type: none"> • perldoc 	Print Perl Documentation, looking it up in the .pod format embedded in perl installation. Support following options: <ul style="list-style-type: none"> • -f : built-in functions • -q : FAQ keyword search • -v : variable • -a perl API
<ul style="list-style-type: none"> • perlivp 	<ul style="list-style-type: none"> • Perl Installation Verification Procedure : checks Perl installation. • Part of perl-level package.
<ul style="list-style-type: none"> • perltidy 	<ul style="list-style-type: none"> • PerlTidy @ Wikipedia , PerlTidy Home Pages: @SourceForge @GitHub. Perl::Tidy GitHub repo, Perl::Tidy @meta::cpan • Online PerlTidy , which provides help on the various options categorized by feature set.
perlsec - Perl security	
Perl Community	Perl has a long history and is quite vast. Here's a collection of links to various web sites that can provide information about it. It is far from complete and collected without much background on what happened in lots of cases and no opinion yet taken on most of this. <ul style="list-style-type: none"> • strictures vs Schmorp common::sense, Marc A.Lehman common::sense package.