








## Syntax Checking Tools

Description	Keystroke	Function	Note	
<b>Syntax Checking</b> <ul style="list-style-type: none"> <li>Help &amp; Customization</li> <li>PEL common syntax check commands for flymake and flycheck</li> <li>Using Flycheck <ul style="list-style-type: none"> <li>Info/manual</li> <li>Setup</li> <li>Syntax check buffer/file</li> <li>Show/Navigate errors</li> </ul> </li> <li>Using Flymake</li> </ul>			<p>Emacs syntax checking can be performed by the built-in <a href="#">flymake</a> package or the <a href="#">flycheck</a> external package.</p> <ul style="list-style-type: none"> <li>Historically <b>flymake</b> appeared first and <b>flycheck</b> used to be more versatile and powerful. However the latest version Emacs improved <b>flymake</b>. It is also used by the eglot Emacs built-in Language Server Protocol client. Note that eaglet only supports flymake but lsp-mode supports flymake and flycheck.</li> <li>👉 <b>Emacs 30.1</b> fixed a shell injection vulnerability in man.el (CVE-2025-1244) that affects <b>flymake</b>, preventing it to execute code deemed unsafe by default. <ul style="list-style-type: none"> <li>Starting with Emacs 30.1, <b>flymake</b> will not execute elisp code unless the <b>trusted-content</b> customizable user-option identifies it as trustable.</li> </ul> </li> <li>There is a large set of external package extensions for both <b>flymake</b> and <b>flycheck</b>. Language specific packages are described in the related pages. The packages listed below provide general feature sets. PEL installs and activates packages when the corresponding <b>pel-use-</b> user-option is turned on.</li> </ul> <p>PEL supports both <b>flymake</b> and <b>flycheck</b>, and implements dispatching commands that use the engine currently used in the buffer.</p> <ul style="list-style-type: none"> <li>PEL uses the &lt;f11&gt; ! key prefix for syntax checking command bindings. The same key bindings are used for PEL dispatching commands common to both engines.</li> </ul> <p>Aside from specific checker, described in the PDF page for their corresponding major mode, PEL can install and activate the following packages when the corresponding user-option is turned on:</p> <div> <div>📦 <a href="#">flymake-collection</a></div> <div>🔧 <a href="#">pel-use-flymake-collection</a></div> <div>A collection of checkers for <a href="#">flymake</a> , replacing a lot of flymake specific extensions.</div> </div> <div> <div>📦 <a href="#">flycheck</a></div> <div>🔧 <a href="#">pel-use-flycheck</a></div> <div>Syntax checking for Emacs, an alternative to the built-in flymake.</div> </div> <div> <div>📦 <a href="#">flycheck-eglot</a></div> <div>🔧 <a href="#">pel-use-flycheck-eglot</a></div> <div>Flycheck support for eglot, which by default, only supports flymake.</div> </div> <div> <div>📦 <a href="#">flycheck-inline</a></div> <div>🔧 <a href="#">pel-use-flycheck-inline</a></div> <div>Display flycheck detected diagnostics at the end of the affected line.</div> </div> <div> <div>📦 <a href="#">flycheck-projectile</a></div> <div>🔧 <a href="#">pel-use-flycheck-projectile</a></div> <div>Project-wide flycheck diagnostics.</div> </div> <p>PEL provide the following 3 user-options you can use to identify whether you want syntax checking activated automatically for specific major modes, for specific files or directories and which syntax checking engine is used, <a href="#">flymake</a> or <a href="#">flycheck</a>, if any.</p> <div> <div>👤👤 <a href="#">pel-fly-engine-for-mode</a></div> <div>An list that maps the major mode name to the syntax checking engine to use for the major mode: <a href="#">flymake</a> and <a href="#">flycheck</a>. By default, it associates emacs-lisp to <a href="#">flymake</a>. <ul style="list-style-type: none"> <li>If a major mode is associated with nil, then automatic activation of syntax checking for that mode is disabled, regardless of the values of <a href="#">pel-modes-activating-syntax-check</a> and <a href="#">pel-auto-activatefly-engines-in-files</a>.</li> </ul> </div> <div> <div>👤👤 <a href="#">pel-modes-activating-syntax-check</a></div> <div>A list of major modes that automatically activate the selected syntax checker.</div> </div> <div> <div>👤👤 <a href="#">pel-files-activating-syntax-check</a></div> <div>A list of file or directories where the syntax checking engine should be activated automatically when the file opens in the Emacs buffer. There are none by default.</div> </div> </div> <p>👉 Syntax checking can also be activated by <a href="#">eglot</a> or <a href="#">lsp-mode</a>.</p> <p>More info in the pages of the following major modes:</p> <div> <div> <ul style="list-style-type: none"> <li>🔧 <a href="#">Pel - Emacs Lisp</a></li> <li>🔧 <a href="#">Pel - Erlang</a></li> <li>🔧 <a href="#">Pel - Go</a></li> <li>🔧 <a href="#">Pel - UNIX Shell</a>.</li> </ul> </div> <div>Other modes support it but they are not yet well documented. 🙏 This includes: <ul style="list-style-type: none"> <li>🔧 <a href="#">Pel - Odin</a></li> <li>🔧 <a href="#">Pel - Rust</a></li> </ul> </div> </div>	
<b>Open this PDF file.</b> See also: <a href="#">🔧 Help/Info</a>	<f11> ! <f1>	( <a href="#">pel-help-pdf</a> &optional OPEN-WEB-PAGE)	Open the <a href="#">🔧 SyntaxChecklocal</a> PDF. If the prefix argument (like <b>C-u</b> or <b>M--</b> ) is used, then it opens the remote GitHub hosted raw PDF instead. If the <b>pel-flip-help-pdf-arg</b> user-option is set it's the other way around.	
<a href="#">🔧 Customize</a> PEL syntax checking control	<f11> ! <f2>	( <a href="#">pel-customize-pel</a> &optional OTHER-WINDOW)	Customize PEL support for: syntax checking. <ul style="list-style-type: none"> <li>If OTHER-WINDOW is non-nil (use <b>C-u</b> ), display in other window.</li> </ul>	
<a href="#">🔧 Customize</a> Emacs syntax checking control	<f11> ! <f3>	( <a href="#">pel-customize-library</a> &optional OTHER-WINDOW)	Customize Emacs spelling support. Opens the following customization groups: flymake, flymake-collection, flycheck, flycheck-eglot, flycheck-inline, flycheck-projectile.	
<b>PEL Common Syntax Checking</b>			PEL provides the following dispatching commands that use the <a href="#">flymake</a> and <a href="#">flycheck</a> supporting command depending which one is being use. If you have a checker for both engine in the current major mode you can also use the command to switch engine.	
<b>Show syntax check setup information</b>	<f11> ! ?	( <a href="#">pel-fly-show-setup-info</a> &optional APPEND)	Print syntax check tools setup information in specialized buffer: *pel-fly-info*. <ul style="list-style-type: none"> <li>Lists all relevant user-options and their values. Each one is a button leading to more information and to its customization buffer.</li> <li>If APPEND is non-nil, append information to the buffer, otherwise clear it and display information from the top.</li> </ul>	
<b>Toggle between flymake and flycheck</b>	<f11> ! E	( <a href="#">pel-fly-toggle-engine</a> )	Toggle between using Flymake or Flycheck when one is used.	
<b>Toggle activation of the syntax checker engine in the current buffer</b>	<f11> ! !	( <a href="#">pel-fly-toggle-syntax-check</a> &optional GLOBALLY)	Toggle the current syntax check engine ( <a href="#">flymake</a> or <a href="#">flycheck</a> ) on/off. <ul style="list-style-type: none"> <li>The engine is first selected by the value of <a href="#">pel-fly-engine-for-mode</a> user-option for the current major mode and whether <a href="#">flycheck</a> is available (🔧 available when <a href="#">pel-use-flycheck</a> is turned on).</li> <li>It changes the buffer local state of the syntax check.</li> <li>You can also toggle the global state of flycheck with the optional GLOBALLY parameter. That parameter is ignored for flymake.</li> </ul>	
<b>Toggle inline display of diagnostics</b>	<f11> ! I	( <a href="#">pel-fly-toggle-diag-at-eol</a> &optional ONLY-ERROR)	Toggle diagnostics display at end of line. <ul style="list-style-type: none"> <li>If optional ONLY-ERROR is specified affect only the display of errors when activating otherwise activate display of all diagnostics at the end of line.</li> </ul> <p>This works fine with <a href="#">flymake</a>. For <a href="#">flycheck</a> it 📦 requires <a href="#">flycheck-inline</a> 🔧 activated by <a href="#">pel-use-flycheck-inline</a>.</p> <p>⚠ When using <a href="#">flycheck-inline</a> do NOT disable it while point is over an error message. It won't be removed.</p>	
<b>List all diagnostics inside a specialized buffer</b>	<f11> ! L	( <a href="#">pel-fly-list-diagnostics</a> &optional ALL-FILES)	List all syntax diagnostics in a specialized buffer. <ul style="list-style-type: none"> <li>If ALL-FILES optional prefix used, list diagnostics of all project files.</li> </ul> <p>Show a list of diagnostics of current buffer inside a "Flymake diagnostic.*" or "Flycheck diagnostic" buffer.</p> <ul style="list-style-type: none"> <li>The following keys are available in those buffers:</li> </ul> <div> <div> <div>&lt;</div> <div><b>p</b></div> <div><b>S-TAB</b></div> <div>{</div> </div> <div> <div>beginning of buffer</div> <div>previous line</div> <div>previous line in description column</div> <div>Narrow the width of current column</div> </div> <div> <div>&gt;</div> <div><b>n</b></div> <div><b>TAB</b></div> <div>}</div> </div> <div> <div>end of buffer</div> <div>next line</div> <div>next line in description column</div> <div>Increase thee width of current column</div> </div> <div> <div><b>S</b></div> <div><b>SPC</b></div> <div><b>RET</b></div> <div>buffer</div> <div><b>g</b></div> <div><b>q</b></div> </div> <div> <div>Sort on current column</div> <div>Show diagnostic in affected buffer</div> <div>Move to diagnostic in affected</div> <div>Refresh buffer</div> <div>Quit</div> </div> </div>	
<b>Go to next flymake/ flycheck diagnostic</b>	<a href="#">M-n</a>	( <a href="#">flymake-goto-next-error</a> &optional N FILTER INTERACTIVE)	Move point to the next Flymake diagnostic. <ul style="list-style-type: none"> <li>With a prefix arg, skip any diagnostics with a severity less than ‘:warning’.</li> <li>Display the error message in the echo line.</li> </ul>	
		( <a href="#">flycheck-next-error</a> &optional N RESET)	Visit the N-th error from the current point. <ul style="list-style-type: none"> <li>N is the number of errors to advance by, where a negative N advances backwards. With non-nil RESET, advance from the beginning of the buffer, otherwise advance from the current position.</li> </ul>	
<b>Go to previous flymake/ flycheck diagnostic</b>	<a href="#">M-p</a>	( <a href="#">flymake-goto-prev-error</a> &optional N FILTER INTERACTIVE)	Move point to the previous Flymake diagnostic. <ul style="list-style-type: none"> <li>With a prefix arg, skip any diagnostics with a severity less than ‘:warning’.</li> <li>Display the error message in the echo line.</li> </ul>	
		( <a href="#">flycheck-previous-error</a> &optional N)	Visit the N-th previous error. <ul style="list-style-type: none"> <li>If given, N specifies the number of errors to move backwards by.</li> <li>If N is negative, move forwards instead.</li> </ul>	

Description	Keystroke	Function	Note	
<b><a href="#">Flycheck</a></b>	<p>Flycheck is a minor mode for on-the-fly syntax checking.</p> <p> The <b>flycheck</b> external package  is activated by PEL when <b>pel-use-flycheck</b> user-option is turned on or another activated PEL user-option requires it.</p> <p> Aside from the following 2 key bindings that PEL provides to toggle the flycheck-mode, flycheck key prefix is <b>C-c !</b> as set by its <b>flycheck-keymap-prefix</b> user-option. You can change it for a different key prefix.</p> <p> Type <b>&lt;f10&gt;</b> to open the <a href="#">menu bar</a> and navigate to Tools/Syntax Checking for the <a href="#">list of flycheck commands from the menu</a>.</p>			
• <b>Info about Flycheck</b>	The following key bindings are available when flycheck-mode is active.			
Open Flycheck manual	C-c ! i	(flycheck-manual)	Open the Flycheck manual.	
Display Flycheck version	C-c ! v	(flycheck-version &optional SHOW-VERSION)	<p>Get the Flycheck version as string.</p> <ul style="list-style-type: none"> <li>If called interactively or if SHOW-VERSION is non-nil, show the version in the echo area and the messages buffer.</li> <li>The returned string includes both, the version from package.el and the library version, if both a present and different.</li> <li>If the version number could not be determined, signal an error, if called interactively, or if SHOW-VERSION is non-nil, otherwise just return nil.</li> </ul>	
• <b>Flycheck setup</b>	The following key bindings are available when flycheck-mode is active.			
Display documentation about syntax checker	C-c ! ?	(flycheck-describe-checker CHECKER)	<p>Display the documentation of CHECKER.</p> <ul style="list-style-type: none"> <li>CHECKER is a checker symbol.</li> <li>Pop up a help buffer with the documentation of CHECKER.</li> </ul>	
Select Flycheck Checker for current buffer	C-c ! s	(flycheck-select-checker CHECKER)	<p>Select <u>CHECKER</u> for the current buffer.</p> <ul style="list-style-type: none"> <li>CHECKER is a syntax checker symbol (see ‘flycheck-checkers’) or nil. In the former case, use CHECKER for the current buffer, otherwise deselect the current syntax checker (if any) and use automatic checker selection via ‘flycheck-checkers’.</li> <li>If called interactively prompt for CHECKER. With prefix arg deselect the current syntax checker and enable automatic selection again.</li> <li>Set ‘flycheck-checker’ to CHECKER and automatically start a new syntax check if the syntax checker changed.</li> <li>CHECKER will be used, even if it is not contained in ‘flycheck-checkers’, or if it is disabled via ‘flycheck-disabled-checkers’.</li> </ul>	
Verify Flycheck setup  Identifies available checkers for the major-mode.	C-c ! v	(flycheck-verify-setup)	<p>Check whether Flycheck can be used in this buffer.</p> <ul style="list-style-type: none"> <li>Display a new buffer listing all syntax checkers that could be applicable in the current buffer. <ul style="list-style-type: none"> <li>For each syntax checkers, possible problems are shown.</li> </ul> </li> </ul>	
		 Use this to identify the various checkers available for the current major mode and find how to install the missing ones.		
Disable Flycheck checker	C-c ! x	(flycheck-disable-checker CHECKER &optional ENABLE)	<p>Interactively disable CHECKER for the current buffer.</p> <ul style="list-style-type: none"> <li>Prompt for a syntax checker to disable, and add the syntax checker to the buffer-local value of ‘flycheck-disabled-checkers’.</li> <li>With non-nil ENABLE or with prefix arg, prompt for a disabled syntax checker and re-enable it by removing it from the buffer-local value of ‘flycheck-disabled-checkers’.</li> </ul>	
• <b>Flycheck buffer/file</b>	The following key bindings are available when flycheck-mode is active.			
Syntax Check current buffer	C-c ! c	(flycheck-buffer)	<p>Start checking syntax in the current buffer.</p> <ul style="list-style-type: none"> <li>Get a syntax checker for the current buffer with ‘flycheck-get-checker-for-buffer’, and start it.</li> </ul>	
Check syntax of current file	C-c ! C-c	(flycheck-compile CHECKER)	<p>Run CHECKER via ‘compile’.</p> <ul style="list-style-type: none"> <li>CHECKER must be a valid syntax checker. Interactively, prompt for a syntax checker to run.</li> <li>Instead of highlighting errors in the buffer, this command pops up a separate buffer with the entire output of the syntax checker tool, just like ‘compile’.</li> </ul>	
• <b>Manage Errors</b>	The following key bindings are available when flycheck-mode is active.			
Show error list for current buffer	<ul style="list-style-type: none"> <li>C-c ! l</li> <li>• &lt;f11&gt; ! l</li> </ul>	(flycheck-list-errors)	Show the error list for the current buffer.	
Display all errors at point	C-c ! h	(flycheck-display-error-at-point)	Display all the error messages at point.	
Explain error at point	C-c ! e	(flycheck-explain-error-at-point)	<p>Display an explanation for the first explainable error at point.</p> <ul style="list-style-type: none"> <li>The first explainable error at point is the first error at point with a non-nil ‘:error-explainer’ function defined in its checker. The ‘:error-explainer’ function is then called with this error to produce the explanation to display.</li> </ul>	
Copy errors	C-c ! C-w	(flycheck-copy-errors-as-kill POS &optional FORMATTER)	<p>Copy each error at POS into kill ring, using FORMATTER.</p> <ul style="list-style-type: none"> <li>FORMATTER is a function to turn an error into a string, defaulting to ‘flycheck-error-message’.</li> <li>Interactively, use ‘flycheck-error-format-message-and-id’ as FORMATTER with universal prefix arg, and ‘flycheck-error-id’ with normal prefix arg, i.e. copy the message and the ID with universal prefix arg, and only the id with normal prefix arg.</li> </ul>	
Clear all errors	C-c ! C	(flycheck-clear &optional SHALL-INTERRUPT)	<p>Clear all errors in the current buffer.</p> <ul style="list-style-type: none"> <li>With prefix arg or SHALL-INTERRUPT non-nil, also interrupt the current syntax check.</li> </ul>	
Move point to next error	<ul style="list-style-type: none"> <li>C-c ! n</li> <li>• M-n</li> </ul>	(flycheck-next-error &optional N RESET)	<p>Visit the N-th error from the current point.</p> <ul style="list-style-type: none"> <li>N is the number of errors to advance by, where a negative N advances backwards. With non-nil RESET, advance from the beginning of the buffer, otherwise advance from the current position.</li> </ul>	
Move point to prior error	<ul style="list-style-type: none"> <li>C-c ! p</li> <li>• M-p</li> </ul>	(flycheck-previous-error &optional N)	<p>Visit the N-th previous error.</p> <ul style="list-style-type: none"> <li>If given, N specifies the number of errors to move backwards by.</li> <li>If N is negative, move forwards instead.</li> </ul>	
<b><a href="#">Using Flymake</a></b>	<p>Flymake performs syntax checking while the user is editing. PEL provides flymake support for some major modes.</p> <p> Flymake has several customizable variables, which some listed here:</p> <p>The following customization variables determine the exact circumstances whereupon Flymake decides to initiate a check of the buffer:</p> <ul style="list-style-type: none"> <li><b>flymake-start-on-flymake-mode</b> : t to start checking when flymake-mode is started. <b>nil</b> to prevent check.</li> <li><b>flymake-no-changes-timeout</b> : time to wait after last change to start checking. Default = 0.5 seconds.</li> <li><b>flymake-start-syntax-check-on-newline</b> : t to check after insertion or removal of newline char from buffer. <b>nil</b> to prevent check.</li> </ul> <p>The following variable control navigation to next or previous error:</p> <ul style="list-style-type: none"> <li><b>flymake-wrap-around</b> : If non-nil, moving to errors wraps around buffer boundaries.</li> <li><b>flymake-diagnostic-types-alist</b> : Alist (KEY . PROPS) of properties of Flymake diagnostic types. See Emacs documentation for more info.</li> </ul>			
Toggle Flymake mode on/off explicitly	M-x flymake-mode	(flymake-mode &optional ARG)	<p>Toggle Flymake mode on or off.</p> <ul style="list-style-type: none"> <li>With a prefix argument ARG, enable Flymake mode if ARG is positive, and disable it otherwise.</li> <li>Flymake is an Emacs minor mode for on-the-fly syntax checking.</li> <li>Flymake collects diagnostic information from multiple sources, called backends, and visually annotates the buffer with the results.</li> </ul>	

## Syntax Checking Tools— References

Topic & link	Description
<b>Flymake</b>	
<b><a href="#">GNU Flymake Manual</a></b>	Flymake is part of Emacs. It has its own manual.
<b>Flycheck</b>	
<b><a href="#">Spotlight: Flycheck, a Flymake replacement</a></b>	Flycheck description by Mickey Petersen

Topic & link	Description
<b><a href="#">Flycheck home page</a></b>	
<b><a href="#">Flycheck supported languages</a></b>	List of programming and markup languages supported by Flycheck.
<b><a href="#">Modern Emacs setup for Erlang (with autocompletion and lint)</a></b>	LambdaCat December 2015 blog, which describes how to use Flycheck for Erlang.