





# Syntax Checking Tools

Description	Keystroke	Function	Note
<div><b>Syntax Checking</b><ul style="list-style-type: none"><li>◦ <a href="#">Help &amp; Customization</a></li><li>• <a href="#">Using Flycheck</a><ul style="list-style-type: none"><li>• <a href="#">Info/manual</a></li><li>• <a href="#">Setup</a></li><li>• <a href="#">Syntax check buffer/file</a></li><li>• <a href="#">Show/Navigate errors</a></li></ul></li><li>◦ <a href="#">Using Flymake</a></li></ul></div> <div>Last updated on: 2025-05-08</div>	<div>Emacs syntax checking can be performed by the built-in <b>flymake</b> package or the newer (and more powerful) <b>flycheck</b> external package.</div> <div> PEL provides key bindings to activate <b>flycheck</b> globally accessible key sequence <b>&lt;f11&gt;!!</b> , but none for <b>flymake</b>, as <b>flycheck</b> is currently preferred.</div> <div><ul style="list-style-type: none"><li>• However, for some major modes like Erlang, Go and Unix shell, PEL provides key bindings to activate the syntax checking mode selected for the major mode, and either <b>flycheck</b> or <b>flymake</b> may be selected by the key sequence as specified by the appropriate user-option controlling syntax checking for the major mode.</li></ul></div> <div>More info in the pages of the following major modes:<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></div>	<b>&lt;f11&gt; ! &lt;f1&gt;</b>	<b>(pel-help-pdf</b> &optional OPEN-WEB-PAGE)	Open the <a href="#">🔧 SyntaxCheck</a> local 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 PEL syntax checking control</a>	<b>&lt;f11&gt; ! &lt;f2&gt;</b>	<b>(pel-customize-pel</b> &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 Emacs syntax checking control</a>	<b>&lt;f11&gt; ! &lt;f3&gt;</b>	<b>(pel-customize-library</b> &optional OTHER-WINDOW)	Customize Emacs spelling support. Opens the following customization groups: flymake, fly check.
<a href="#">Flycheck</a>	<div>Flycheck is a minor mode for on-the-fly syntax checking.</div> <div> 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.</div> <div> 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.</div> <div>👉 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>.</div>		
<b>Toggle flycheck mode for current buffer</b>	<b>&lt;f11&gt; ! !</b>	<b>(flycheck-mode</b> &optional ARG)	Toggle flycheck minor-mode for the current buffer or flymake syntax checking mode when it is selected by the user-option appropriate for the major mode. Several major modes binds the same key sequence to another command that is specific to the major mode but either activate flycheck-mode or flymake-mode depending on the customization of the major mode. Refer to the documentation of the major mode for more information.
<b>Toggle flycheck mode for all buffers</b>	<b>&lt;f11&gt; ! M-!</b>	<b>(global-flycheck-mode</b> &optional ARG)	Toggle Flycheck mode in all buffers. <ul style="list-style-type: none"><li>• Flycheck mode is enabled in all buffers where ‘flycheck-mode-on-safe’ would do it.</li></ul>
<b>• Info about Flycheck</b>	The following key bindings are available when flycheck-mode is active.		
<b>Open Flycheck manual</b>	<b>C-c ! i</b>	<b>(flycheck-manual)</b>	Open the Flycheck manual.
<b>Display Flycheck version</b>	<b>C-c ! v</b>	<b>(flycheck-version</b> &optional SHOW-VERSION)	Get the Flycheck version as string. <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.		
<b>Display documentation about syntax checker</b>	<b>C-c ! ?</b>	<b>(flycheck-describe-checker</b> CHECKER)	Display the documentation of CHECKER. <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>
<b>Select Flycheck Checker for current buffer</b>	<b>C-c ! s</b>	<b>(flycheck-select-checker</b> CHECKER)	Select <a href="#">CHECKER</a> for the current buffer. <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>
<div><b>Verify Flycheck setup</b> 👉 Identifies available checkers for the major-mode.</div>	<b>C-c ! v</b>	<b>(flycheck-verify-setup)</b>	Check whether Flycheck can be used in this buffer. <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.	
<b>Disable Flycheck checker</b>	<b>C-c ! x</b>	<b>(flycheck-disable-checker</b> CHECKER &optional ENABLE)	Interactively disable CHECKER for the current buffer. <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.		
<b>Syntax Check current buffer</b>	<b>C-c ! c</b>	<b>(flycheck-buffer)</b>	Start checking syntax in the current buffer. <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>
<b>Check syntax of current file</b>	<b>C-c ! C-c</b>	<b>(flycheck-compile</b> CHECKER)	Run CHECKER via ‘compile’. <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.		
<b>Show error list for current buffer</b>	<ul style="list-style-type: none"><li>• <b>C-c ! l</b></li><li>• <b>&lt;f11&gt; ! l</b></li></ul>	<b>(flycheck-list-errors)</b>	Show the error list for the current buffer.
<b>Display all errors at point</b>	<b>C-c ! h</b>	<b>(flycheck-display-error-at-point)</b>	Display all the error messages at point.
<b>Explain error at point</b>	<b>C-c ! e</b>	<b>(flycheck-explain-error-at-point)</b>	Display an explanation for the first explainable error at point. <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>
<b>Copy errors</b>	<b>C-c ! C-w</b>	<b>(flycheck-copy-errors-as-kill</b> POS &optional FORMATTER)	Copy each error at POS into kill ring, using FORMATTER. <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>
<b>Clear all errors</b>	<b>C-c ! C</b>	<b>(flycheck-clear</b> &optional SHALL-INTERRUPT)	Clear all errors in the current buffer. <ul style="list-style-type: none"><li>• With prefix arg or SHALL-INTERRUPT non-nil, also interrupt the current syntax check.</li></ul>
<b>Move point to next error</b>	<b>C-c ! n</b>	<b>(flycheck-next-error</b> &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>Move point to prior error</b>	<b>C-c ! p</b>	<b>(flycheck-previous-error</b> &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	
<a href="#">Using Flymake</a>	<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> : <b>t</b> 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> : <b>t</b> 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	M-x flymake-mode	(flymake-mode &optional ARG)	Toggle Flymake mode on or off. <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>	
	<f11> ! !		Several major modes binds this key sequence to another command that is specific to the major mode but either activate flycheck-mode or flymake-mode depending on the customization of the major mode. Refer to the documentation of the major mode for more information. <ul style="list-style-type: none"> <li>• If it is not bound, invoke the command manually as above.</li> </ul>	
Go to next flymake diagnostic	M-n	(flymake-goto-next-error &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>	
Go to previous flymake diagnostic	M-p	(flymake-goto-prev-error &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>	

## Syntax Checking Tools— References

Topic & link	Description
<a href="#">Flymake</a>	
<a href="#">GNU Flymake Manual</a>	Flymake is part of Emacs. It has its own manual.
<a href="#">Flycheck</a>	
<a href="#">Spotlight: Flycheck, a Flymake replacement</a>	Flycheck description by Mickey Petersen
<a href="#">Flycheck home page</a>	
<a href="#">Flycheck supported languages</a>	List of programming and markup languages supported by Flycheck.
<a href="#">Modern Emacs setup for Erlang (with autocompletion and lint)</a>	LambdaCat December 2015 blog, which describes how to use Flycheck for Erlang.