Topic Index

Syntax Checking Tools

Description	Keystroke	Function	Note
Syntax Checking	Emacs syntax checking can be performed by the built-in **flymake** package or the newer **flycheck** external package. • More information for each programming language will be identified in the language specific page.		
Using Flymake	Flymake performs these checks while the user is editing. Flymake has several customizable variables, which some listed here: The following customization variables determine the exact circumstances whereupon Flymake decides to initiate a check of the buffer: • **flymake-start-on-flymake-mode** : **t** to start checking when flymake-mode is started. **nil** to prevent check. • **flymake-no-changes-timeout** : time to wait after last change to start checking. Default = 0.5 seconds. • **flymake-start-syntax-check-on-newline** : **t** to check after insertion or removal of newline char from buffer. **nil** to prevent check. The following variable control navigation to next or previous error: • **flymake-wrap-around** : If non-nil, moving to errors wraps around buffer boundaries. • **flymake-diagnostic-types-alist** : Alist ((KEY . PROPS)*) of properties of Flymake diagnostic types. See Emacs documentation for more info.		
Toggle Flymake mode on/off	`<f12> !`	(**flymake-mode** &optional ARG)	Toggle Flymake mode on or off. • With a prefix argument ARG, enable Flymake mode if ARG is positive, and disable it otherwise. • Flymake is an Emacs minor mode for on-the-fly syntax checking. • Flymake collects diagnostic information from multiple sources, called backends, and visually annotates the buffer with the results.
Go to next flymake diagnostic	`M-n`	(**flymake-goto-next-error** &optional N FILTER INTERACTIVE)	Move point to the next Flymake diagnostic. • With a prefix arg, skip any diagnostics with a severity less than ':warning'. • Display the error message in the echo line.
Go to previous flymake diagnostic	`M-p`	(**flymake-goto-prev-error** &optional N FILTER INTERACTIVE)	Move point to the previous Flymake diagnostic. • With a prefix arg, skip any diagnostics with a severity less than ':warning'. • Display the error message in the echo line.
Flycheck	Flycheck is a minor mode for on-the-fly syntax checking. The **flycheck** external package is activated by PEL when the **pel-use-flycheck** user-option is turned on or another activated PEL user-option requires it. Aside from the following 2 key bindings that PEL provides to toggle the flycheck mode, flycheck key prefix is **`C-c !`** as set by its **flycheck-keymap-prefix** user-option. You can change it for a different key prefix.		
Toggle flycheck mode for current buffer	`<f11> ! !`	(**flycheck-mode** &optional ARG)	Toggle flycheck minor-mode for the current buffer.
Toggle flycheck mode for all buffers	`<f11> ! M-!`	(**global-flycheck-mode** &optional ARG)	Toggle Flycheck mode in all buffers. • Flycheck mode is enabled in all buffers where 'flycheck-mode-on-safe' would do it.
• **Info about Flycheck**			
Open Flycheck manual	`C-c ! i`	(**flycheck-manual**)	Open the Flycheck manual.
Display Flycheck version	`C-c ! V`	(**flycheck-version** &optional SHOW-VERSION)	Get the Flycheck version as string. • If called interactively or if SHOW-VERSION is non-nil, show the version in the echo area and the messages buffer. • The returned string includes both, the version from package.el and the library version, if both a present and different. • 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.
• **Flycheck setup**			
Display documentation about syntax checker	`C-c ! ?`	(**flycheck-describe-checker** CHECKER)	Display the documentation of CHECKER. • CHECKER is a checker symbol. • Pop up a help buffer with the documentation of CHECKER.
Select Flycheck Checker for current buffer	`C-c ! s`	(**flycheck-select-checker** CHECKER)	Select CHECKER for the current buffer. • 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'. • If called interactively prompt for CHECKER. With prefix arg deselect the current syntax checker and enable automatic selection again. • Set 'flycheck-checker' to CHECKER and automatically start a new syntax check if the syntax checker changed. • CHECKER will be used, even if it is not contained in 'flycheck-checkers', or if it is disabled via 'flycheck-disabled-checkers'.
Verify Flycheck setup	`C-c ! v`	(**flycheck-verify-setup**)	Check whether Flycheck can be used in this buffer. • Display a new buffer listing all syntax checkers that could be applicable in the current buffer. For each syntax checkers, possible problems are shown.
Disable Flycheck checker	`C-c ! x`	(**flycheck-disable-checker** CHECKER &optional ENABLE)	Interactively disable CHECKER for the current buffer. • Prompt for a syntax checker to disable, and add the syntax checker to the buffer-local value of 'flycheck-disabled-checkers'. • 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'.
• **Flycheck buffer/file**			
Syntax Check current buffer	`C-c ! c`	(**flycheck-buffer**)	Start checking syntax in the current buffer. • Get a syntax checker for the current buffer with 'flycheck-get-checker-for-buffer', and start it.
Check syntax of current file	`C-c ! C-c`	(**flycheck-compile** CHECKER)	Run CHECKER via 'compile'. • CHECKER must be a valid syntax checker. Interactively, prompt for a syntax checker to run. • 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'.
• **Manage Errors**			
Show error list for current buffer	`C-c ! l`	(**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**)	Display an explanation for the first explainable error at point. • 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.
Copy errors	`C-c ! C-w`	(**flycheck-copy-errors-as-kill** POS &optional FORMATTER)	Copy each error at POS into kill ring, using FORMATTER. • FORMATTER is a function to turn an error into a string, defaulting to 'flycheck-error-message'. • 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.
Clear all errors	`C-c ! C`	(**flycheck-clear** &optional SHALL-INTERRUPT)	Clear all errors in the current buffer. • With prefix arg or SHALL-INTERRUPT non-nil, also interrupt the current syntax check.

Description	Keystroke	Function	Note
Move point to next error	`C-c ! n`	(**flycheck-next-error** &optional N RESET)	Visit the N-th error from the current point. • 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.
Move point to prior error	`C-c ! p`	(**flycheck-previous-error** &optional N)	Visit the N-th previous error. • If given, N specifies the number of errors to move backwards by. • If N is negative, move forwards instead.

Syntax Checking Tools— References

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