↖ Topic Index

🚧 Emacs support for the Lua Programming Language 🚧

Description	Keystroke	Function	Note
Lua Editing Last updated on:	Emacs has built-in support for Lua. The Lua-mode is one of the cc-modes. • Since Lua syntax is very close to C syntax, Emacs implements **Lua-mode** as a descendent of cc-mode. PEL provides extra support, described in this table, when the **pel-use-Lua** user-option is set to t. • Most cc-mode available capabilities are available to **Lua-mode**. PEL integrates a lot of those capabilities, but PEL support for Lua is in its early stages and all available key bindings are not yet identified in this table as they should be. 🚧 2025-03-19		
Open this PDF file. See also: **Σ Help/Info**	`<f11> SPC u <f1>` `<f12> <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **𝔓𝔇𝔉 - Lua** local PDF. If the prefix argument (like **C-u** or **M--**) is used, then it 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.
Σ Customize PEL Lua support	`<f11> SPC u <f2>` `<f12> <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Lua support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Σ Customize Emacs Lua support	`<f11> SPC u <f3>` `<f12> <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs Lua support (which is currently placed in C group): C • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Select Lua-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.	`<f12>`	(**pel-as** &optional FORCE)	Inside a fundamental-mode buffer, interactively select major mode for the buffer. Re-do it with arg. This command is mostly used to set the major mode of a buffer in fundamental-mode', when the `<f12>` key binding is available for it. After being used once in a buffer the major mode is selected and the PEL key binding will not be available when PEL supports the major mode. For Lua file, select **Lua.** It will inserts a perl shebang line specified by **pel-Lua-shebang-line** user option. PEL defines the (as &optional FORCE) alias unless **pel-has-alias-as** user-option is set to nil. You can use `M-x as` to invoke it.
Comments			
Toggle display of comments in buffer or active region See also: **Σ Comments**	`<f11> ; ;`	(**hide/show-comments-toggle** &optional START END)	Toggle hiding/showing of comments in the active region or whole buffer. • If the region is active then toggle in the region. Otherwise, in the whole buffer. 📦 This requires the **hide-comnt.el** package (see **Σ Comments**). PEL activates it when the **pel-use-hide-comnt** user option is **t**.

Generic code skeletons
• **tempo skeletons**
See also:
• **Σ Inserting Text**
• **T Templates**
	Several mechanisms have been developed to allow easy insertion of predefined text in Emacs. ⚠ PEL does not yet define skeletons for Lua. You can use the generic one.
	• Emacs provides the built-in skeleton mechanism and the **tempo skeletons**.
	  • PEL supports both. They are used a little bit differently. PEL provides **generic** tempo skeletons you can use for Lua until PEL adds Lua-specific skeletons.
	  • 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.		

Σ Customize PEL Text Insertions control for Lua code skeletons.	`<f6> <f2>` `<f12> <f12> <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW) (**pel-customize-generic-skels** &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). • If OTHER-WINDOW is non-nil (use **C-u**), display in other window.
Insert generic file module header block — Language agnostic After inserting the template, navigate though areas that must be filled with: • forward: **C-c .** • backward: **C-c ,**	`<f6> h` `<f12> <f12> 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 Lua-specific skeletons, this uses the generic one. ☝ Specify the format of the header via the user-options in the **pel-pkg-generic-code-style** customization group accessible via `<f6> <f2>` • Inside a **Lua** 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` `<f12> <f12> SPC`	(**pel-tempo-mode** &optional ARG)	Toggle PEL tempo mode on/off. 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 Note: PEL default skeleton does not use tags.	`<f6> <f12>` `<f12> <f12> <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. • 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.

Emacs & Pike — References

Document	Notes
The Lua Programming Language	• Lua @ Wikipedia • Lua Home