↖ Topic Index

Emacs support for Gleam 🚧

Description	Keystroke	Function	Note

Gleam Support
 	
• File associations
 	
Last updated on:
	Gleam is an functional static-type checking language for the **Erlang BEAM**.
	Gleam Emacs support is evolving.
	PEL supports the only supported mode for Gleam: the **tree-sitter**-based **gleam-ts-mode** provided by the **gleam-mode** package.
	📦 Requires the **gleam-mode** file PEL installs it in the utils directory when **pel-use-gleam** user-option is set to **t**.
	• PEL associates the files with the .gleam file extension with **gleam-ts-mode**..
	⚠️ PEL support for Gleam requires **Emacs >= 30.1** because tree-sitter is required by **gleam-mode**, and PEL only support tree-sitter for Emacs >= 30.1:
	  • See 𝛴 **Tree Sitter** and 🚦 **Tree-sitter**.
	• PEL activates 𝛴 **Speedbar** support for the Gleam files when **pel-use-speedbar** user-option is on (set to **t**).
	• imenu support provided by **gleam-ts-mode** is available.
	• The Gleam community decided that indentation in gleam files should always use 2 spaces.
	  • Therefore PEL does not offer control for this; it delegates the logic to the **gleam-ts-mode** which imposes a fixed indentation offset of 2 spaces. However it is still possible to change the value of tab-width (which has no impact on indentation) and whether hard tabs are used.
	
	2025-10-20		

Open this PDF file. See also: 𝛴 **Help/Info**	`<f11> SPC M-G <f1>` `<f12> <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the 𝔊𝔏 - **Gleam** 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 Gleam support	`<f11> SPC M-G <f2>` `<f12> <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Gleam support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Show PEL setup for Gleam	`<f11> SPC M-G ?` `<f12> ?`	(**pel-gleam-setup-info &optional APPEND**)	Display Gleam setup information inside a *pel-gleam-info* buffer with buttons providing quick access to the customization buffer of each variable shown. The information shown includes the value and interpretation of: • gleam-ts-format-on-save • gleam-ts-indent-offset • tab-width To append information in the buffer instead of clearing the previous content type any prefix argument (such as `C-u`) before the command keystroke.
Set visual rendering of hard tabs for the current buffer	`<f11> <tab> w`	(**pel-set-tab-width** N)	Change the tab width of the current buffer, only affecting the display rendering of hard tabs inserted in the buffer text. Prompts for a new value in the [2, 8] range. • This modifies a buffer local value of the the **tab-width** user-option. • The change is temporary and affects the current buffer only. • To change the tab width used for all Gleam source code files, change the '**pel-gleam-tab-width**' user-option variable instead. See 𝛴 **Indentation** for more information.
Toggle running gleam format on buffer save	`<f11> SPC M-G M-s` `<f12> M-s`	(**pel-gleam-toggle-format-on-buffer-save** &optional GLOBALLY)	Toggle automatic run of gleam format when saving Gleam buffer to file. • By default change behaviour for local buffer only. • When GLOBALLY argument is non-nil, change it for all Gleam buffers for the current Emacs editing session (the change does not persist across Emacs sessions). • To modify the global state permanently modify the customized value of the **gleam-ts-format-on-save** user option.
Comments	See also: 𝛴 **Comments**		
Insert, realign, comment/uncomment region **With PEL**: Comment the current line with `M-0 M-;`	`M-;`	(**comment-dwim** ARG)	Insert or realign comment on current line (or region if a region is active). If line/region is already commented, uncomment it. • On a single line, the comment is placed *after* the code. • `C-u M-;` executes comment-kill
		(**pel-comment-dwim** ARG)	Same as **comment-dwim** but comments the current line with a numeric ARG or 0.

Emacs & Gleam— References

Document	Notes		
The Gleam programming language	• Gleam @ Wikipedia • Gleam home • Gleam @ Github	Github repos: • gleam • stdlib • otp • http	• awesome-gleam • gleam cookbook
Learning Gleam	• The language Tour		
Gleam References	• Gleam stdlib @ hexdoc		
Installing Gleam	• Install Gleam	Since Gleam is a BEAM language, you need to install Erlang first, then rebar3 and then Gleam.	
• **Install Gleam from source**	If you have Erlang, rebar3 and Rust already installed you can also build Gleam from source, using the following commands: cd gleam-repos *# Use the directory name that will hold all gleam related repos* git clone https://github.com/gleam-lang/gleam cd gleam git co v1.12.0 *# check out the branch you want to build- at first use the last released* make install # type: make to list possible other actions. gleam --version # gleam is installed in the ~/.cargo/bin directory		
gleam implementation @ Github			
Interview with Gleam creator			
Emacs support	• **gleam-mode** . Now with **tree-sitter** support. The original **gleam-mode** was replaced with **gleam-ts-mode**. • **tree-sitter-gleam** implements the syntax parsing. • gleam's **grammar.js**, the file that controls tree-sitter grammar for gleam.		