





Emacs support for Gleam

Description	Keystroke	Function	Note
Gleam Support <ul style="list-style-type: none"> File associations See also: 🔗 Speedbar <div> Last updated on: 2025-10-14 </div>	<p>Gleam is an functional static-type checking language for the Erlang BEAM. Gleam Emacs support is evolving. PEL supports the original gleam-mode and the newer tree-sitter-based gleam-ts-mode now provided by the gleam-mode package.</p> <p> Requires the gleam-mode file  PEL installs it in the utils directory when pel-use-gleam user-option is set to t.</p> <ul style="list-style-type: none"> PEL associates the files with the .gleam file extension with gleam-mode and gleam-ts-mode but invokes gleam-ts-mode. <div> ⚠ Because tree-sitter is now required by gleam-mode, and because PEL only support tree-sitter for Emacs >= 30.1: PEL support for Gleam is only available for Emacs >= 30.1 </div> PEL activates 🔗 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 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. 		
Open this PDF file. See also: 🔗 Help/Info	<div><f11> SPC M-G <f1></div> <div><f12> <f1></div>	<div>(pel-help-pdf &optional OPEN-WEB-PAGE)</div>	Open the  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.
🔗 Customize PEL Gleam support	<div><f11> SPC M-G <f2></div> <div><f12> <f2></div>	<div>(pel-customize-pel &optional OTHER-WINDOW)</div>	Customize PEL Gleam support. <ul style="list-style-type: none"> If OTHER-WINDOW is non-nil (use C-u), display in another window.
Show PEL setup for Gleam	<div><f11> SPC M-G ?</div> <div><f12> ?</div>	<div>(pel-gleam-setup-info & optional APPEND)</div>	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: <ul style="list-style-type: none"> 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	<div><f11> M-t</div>	<div>(pel-set-tab-width N)</div>	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. <ul style="list-style-type: none"> 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 🔗 Indentation for more information.
Toggle running gleam format on buffer save	<div><f11> SPC M-G M-s</div> <div><f12> M-s</div>	<div>(pel-gleam-toggle-format-on-buffer-save &optional GLOBALLY)</div>	Toggle automatic run of gleam format when saving Gleam buffer to file. <ul style="list-style-type: none"> 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: 🔗 Comments		
Insert, realign, comment/uncomment region With PEL: Comment the current line with M-0 M-;	<div>M-;</div>	<div>(comment-dwim ARG)</div> <div>(pel-comment-dwim ARG)</div>	Insert or realign comment on current line (or region if a region is active). If line/region is already commented, uncomment it. <ul style="list-style-type: none"> On a single line, the comment is placed <i>after</i> the code. C-u M-; executes comment-kill Same as comment-dwim but comments the current line with a numeric ARG or 0.

Emacs & Gleam— References

Document	Notes	
The Gleam programming language	<ul style="list-style-type: none"> Gleam @ Wikipedia Gleam home Gleam @ Github 	Github repos: <div> <ul style="list-style-type: none"> gleam stdlibb otp http </div> <div> <ul style="list-style-type: none"> awesome-gleam gleam cookbook </div>
Learning Gleam	<ul style="list-style-type: none"> The language Tour 	
Gleam References	<ul style="list-style-type: none"> Gleam stdlib @ hexdoc 	
Installing Gleam	<ul style="list-style-type: none"> Install Gleam 	Since Gleam is a BEAM language , you need to install Erlang first, then rebar3 and then Gleam.
<ul style="list-style-type: none"> Install Gleam from source 	If you have Erlang, rebar3 and Rust already installed you can also build Gleam from source, using the following commands: <pre> cd gleam-repos git clone https://github.com/gleam-lang/gleam cd gleam git co v1.12.0 make install gleam --version </pre> <div> # Use the directory name that will hold all gleam related repos # check out the branch you want to build- at first use the last released # type: make to list possible other actions. # gleam is installed in the ~/.cargo/bin directory </div>	
gleam implementation @ Github		
Interview with Gleam creator		
Emacs support	<ul style="list-style-type: none"> gleam-mode . Now with tree-sitter support. The original gleam-mode was replaced with gleam-ts-mode. tree-sitter-gleam implements the syntax parsing. <ul style="list-style-type: none"> gleam's grammar.js, the file that controls tree-sitter grammar for gleam. 	