↖ Topic Index

Emacs Support for Gerbil Scheme 🚧

Description	Keystroke	Function	Note
Gerbil Scheme Programming Language Support See also: • **𝔓𝔏 - Scheme** • **∑ File/ Directory Variables**	🚧 PEL support for Gerbil Scheme is preliminary. PEL activates Gerbil Scheme support when the **pel-use-scheme** user-option is turned on (**t**). PEL provide extra support for the Scheme programming language and its various implementations by providing access to the following external packages: 📦 The **gerbil-mode** external package. PEL activates it when the **pel-use-gerbil** user-option is turned on (**t**). Used only for **Gerbil Scheme**. • The Gerbil programming language is a specialized Scheme. The Gerbil files use the same extension as Scheme: .ss. • To activate the **gerbil-mode** automatically for Gerbil files, it is customary to use Emacs file variable to identify the mode: the first line of the file should have the following text: `;; -*- Gerbil -*-` • The gerbil-mode package can also be installed by installing Gerbil. • PEL provides Lispy support for Gerbil Scheme when the gerbil-mode is added to the list specified by **pel-modes-activating-lispy** user-option. • See **𝔓𝔏- Lispy**		
Open this PDF file. See also: **∑ Help/ Info**	`<f11> SPC C-s C-e <f1>` `<f12> <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **𝔓𝔏 - Gerbil Scheme** 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 Gerbil Scheme support	`<f11> SPC C-s C-e <f2>` `<f12> <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Gerbil Scheme support. • If OTHER-WINDOW is non-nil (use **`C-u`**), display in another window.
∑ Customize Emacs Gerbil Scheme support	`<f11> SPC C-s C-e <f3>` `<f12> <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs Scheme support: gerbil-mode, scheme, geiser, quack, lispy. • If OTHER-WINDOW is non-nil (use **`C-u`**), display in another window.
	Use the following commands to interact with the **gxi Scheme Gerbil REPL**		
Compile current buffer	`C-c C-f`	(**gerbil-compile-current-buffer**)	Compile the current buffer
Import current buffer	• `C-c C-i` • `C-c <tab>`	(**gerbil-import-current-buffer**)	Import current buffer
Reload current buffer	`C-c C-r`	(**gerbil-reload-current-buffer**)	Reload current buffer
Build	`C-c C-b`	(**gerbil-build**)	Build
Evaluate current definition	`C-c C-e`	(**scheme-send-definition**)	Send the current definition to the inferior Gerbil Scheme process.
Evaluate marked region	`C-c C-c`	(**scheme-send-region** START END)	Send the current region to the inferior Gerbil Scheme process.
Restart inferior scheme process	`C-x 9`	(**restart-scheme**)	Restart the inferior Gerbil Scheme Process
Open the Gerbil REPL	`<f11> z r C-e` `<f12> z`	(**pel-gerbil-repl** &optional N)	Run the Gerbil REPL in window specified by N. • By default use the other window. If a numeric argument is specified, its value correspond to the direction of a numeric keypad: 8 4 6 2 That is: • 8: up • 4: left • 6: right • 2: down • 0 and 5 identify the current window.
Erase the content of REPL	`<f12> C-l`	(**pel-clear-scheme-repl-buffer**)	Erase content of the Gerbil Scheme REPL running under Emacs.