Customizing Emacs with PEL

Operation	Keystroke	Function	Notes
PEL: Control Emacs via Easy Customization	PEL is designed to help you get going quickly with Emacs. Instead of having to write Emacs Lisp code, you use Emacs easy-to-use customization system. • PEL provides a growing set of customization groups and user option variables that control several aspects of Emacs: • The "**pel-use-**" activation user options identify what built-in or external Emacs Lisp package to use. PEL has logic to autoload the packages only when you need them. This way your Emacs will start quickly even if you have identified a large number of packages. • Once a package or feature is activated with the "pel-use-" user option, the other options control different behaviour of the activated package. ☝ Once you have modified the configuration, execute `M-x pel-init`. PEL will activate the new configuration.		
Emacs Easy Customization	With the following command you can customize anything. With the first command you open the customization buffer and then you can search or browse what you want to customize. The second command allow you to open the buffer at a specific customization group and the third one at a specific user option. These commands prompt for the information you are looking for. You can always use completion by typing `<tab>` at any point to get a list of available groups or variables. Several of the commands below open the PEL customization group and one or several other groups related to the same topic, when these groups are already loaded. • If you set the OTHER-WINDOW argument, the command open s the buffer in another window and also open any group related that exists. For example if you open the PEL group for grep with `C-u <f11> <f1> g`, this will also open the grep group, the rg and ripgrep groups if they are loaded. Each group will open inside its own bugger and the command will create the necessary windows.		
Customize Emacs	`<f11> <f1> E`	(customize)	Select a customization buffer which you can use to set user options. • User options are structured into "groups". • Initially the top-level group 'Emacs' and its immediate subgroups are shown; the contents of those subgroups are initially hidden.
Customize a specific group	`<f11> <f1> G`	(**customize-group** &optional GROUP OTHER-WINDOW)	Customize GROUP, which must be a customization group. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Customize a user option	`<f11> <f1> O`	(**customize-option** SYMBOL)	Customize SYMBOL, which must be a user option.
Customize Specific Emacs Groups. Customize:	The following key bindings are mapped to PEL commands that open the customization buffer for one of Emacs specific customization group. These are all specializations of calls to the customize-group command. They are defined commands with their own name so they can be displayed by which-key in Emacs mini-buffer. These commands are also referred in the pages that use the related commands.		
webjump	`<f11> <f1> M-g j`	(**pel-cfge-webjump** &optional OTHER-WINDOW)	Customize webjump. With `C-u`, display in another window.
locate	`<f11> <f1> M-g l`	(**pel-cfge-locate** &optional OTHER-WINDOW)	Customize locate. With `C-u`, display in another window.
man	`<f11> <f1> M-g m`	(**pel-cfge-man** &optional OTHER-WINDOW)	Customize man. With `C-u`, display in another window.
browse-url	`<f11> <f1> M-g u`	(**pel-cfge-browse-url** &optional OTHER-WINDOW)	Customize browse-url. With `C-u`, display in another window.
woman	`<f11> <f1> M-g w`	(**pel-cfge-woman** &optional OTHER-WINDOW)	Customize woman. With `C-u`, display in another window.
Customize PEL	The following commands opens the Emacs customization group related to a PEL topic. None of these commands prompt; they open the customization buffer at the requested group. ⚠ ☝ To activate any PEL customization change in the current session, execute `M-x pel-init` after you saving and applying the customized variable.		
All PEL	`<f11> <f1> !`	(**pel-cfg** &optional OTHER-WINDOW)	Customize PEL support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Permanently change the cursor's color (See also:∑ Cursor)	`<f11> <f1> M-c`	(**pel-customize-cursor** &optional OTHER-WINDOW)	Quicks access to the customize buffer to set the cursor default color. • It sets the color permanently if the customization is saved. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window. ⚠ Only available in graphics mode.
Cursor Support (See also:∑ Cursor)	`<f11> <f1> _`	(**pel-cfg-pkg-cursor** &optional OTHER-WINDOW)	Customize PEL cursor support. Multiple-cursors is controlled here. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Completion Support	`<f11> <f1> c`	(**pel-cfg-pkg-completion** &optional OTHER-WINDOW)	Customize PEL Completion support. • If OTHER-WINDOW is non-nil (use `C-u)`, display in other window.
Dired support	`<f11> <f1> d`	(**pel-cfg-pkg-dired** &optional OTHER-WINDOW)	Customize PEL Dired support. • If OTHER-WINDOW is non-nil (use `C-u)`, display in other window and open Dired related groups.
File/Directory Management	`<f11> <f1> f`	(**pel-cfg-pkg-filemng** &optional OTHER-WINDOW)	Customize PEL Filemng support. • If OTHER-WINDOW is non-nil (use `C-u`), display in other window and open file management related groups.
Grep	`<f11> <f1> g`	(**pel-cfg-pkg-grep** &optional OTHER-WINDOW)	Customize PEL Grep support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window and open grep related groups.
User Identification	`<f11> <f1> i`	(**pel-cfg-identification** &optional OTHER-WINDOW)	Customize PEL Identification support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Key Chords (See also:∑ Key-Chords)	`<f11> <f1> K`	(**pel-customize-key-chords** &optional OTHER-WINDOW)	Customize PEL Key Chord support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Regular Expression	`<f11> <f1> r`	(**pel-cfg-pkg-regexp** &optional OTHER-WINDOW)	Customize PEL regular expression tool support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Search	`<f11> <f1> s`	(**pel-cfg-pkg-search** &optional OTHER-WINDOW)	Customize PEL Search support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window and open search related groups.
Session management	`<f11> <f1> S`	(**pel-cfg-pkg-session** &optional OTHER-WINDOW)	Customize PEL Session support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Speedbar Management	`<f11> <f1> M-s`	(**pel-cfg-pkg-speedbar** &optional OTHER-WINDOW)	Customize PEL Speedbar support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window and open speedbar management groups.
Window management	`<f11> <f1> w`	(**pel-cfg-pkg-window** &optional OTHER-WINDOW)	Customize PEL Window support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window and open window management groups.
Configure PEL Programming Language support	The following commands opens the Emacs configuration group to configure PEL support for the specified programming language. • You should be able to control most of the important features of the programming languages through these customizations including the activation of important packages as well as aspects of programming language styles like indentation style and width. • The `<f11> <f1> SPC` key prefix is available globally (for all buffers). • The `<f12> <f1>` key is only available when point is in a buffer for one of the languages supported by PEL and open the PEL customization group for the programming language for the current buffer. ⚠ ☝ To activate any PEL customization change in the current session, execute `M-x pel-init` after you saving and applying the customized variable.		
AppleScript & Text-to-Speech (audio narration)	• `<f11> <f1> SPC a` • `<f12> <f1>`	(**pel-cfg-pkg-applescript** &optional OTHER-WINDOW)	Customize PEL Applescript support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.

Operation	Keystroke	Function	Notes
C	• `<f11> <f1> SPC c` • `<f12> <f1>`	**(pel-cfg-pkg-c** &optional OTHER-WINDOW)	Customize PEL C support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open C programming groups.
C++	• `<f11> <f1> SPC C` • `<f12> <f1>`	**(pel-cfg-pkg-c++** &optional OTHER-WINDOW)	Customize PEL C++ support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open C++ programming groups.
D	• `<f11> <f1> SPC D` • `<f12> <f1>`	**(pel-cfg-pkg-d** &optional OTHER-WINDOW	Customize PEL D support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open D programming groups.
Emacs Lisp	• `<f11> <f1> SPC l` • `<f12> <f1>`	**(pel-cfg-pkg-elisp** &optional OTHER-WINDOW)	Customize PEL Elisp support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open Elisp programming groups.
Elixir	• `<f11> <f1> SPC x` • `<f12> <f1>`	**(pel-cfg-pkg-elixir** &optional OTHER-WINDOW)	Customize PEL Elixir support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open Elixir programming groups.
Erlang	• `<f11> <f1> SPC e` • `<f12> <f1>`	**(pel-cfg-pkg-erlang** &optional OTHER-WINDOW)	Customize PEL Erlang support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open Erlang programming groups.
Julia	• `<f11> <f1> SPC j` • `<f12> <f1>`	**(pel-cfg-pkg-julia** &optional OTHER-WINDOW)	Customize PEL Julia support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open Julia programming groups.
Common Lisp	• `<f11> <f1> SPC L` • `<f12> <f1>`	**(pel-cfg-pkg-lisp** &optional OTHER-WINDOW)	Customize PEL Lisp support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open Lisp programming groups.
Python	• `<f11> <f1> SPC p` • `<f12> <f1>`	**(pel-cfg-pkg-python** &optional OTHER-WINDOW	Customize PEL Python support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open Python programming groups.
Configure PEL Markup support	The following commands opens the Emacs customization group related to configure PEL support for the specific markup language. • The `<f11> <f1> SPC` key prefix is available globally (for all buffers). • The `<f12> <f1>` key is only available when point is in a buffer for one of the languages supported by PEL and open the PEL customization group for the markup language for the current buffer. ⚠️ ☝️To activate any PEL customization change in the current session, execute **M-x pel-init** after you saving and applying the customized variable.		
reStructuredText	• `<f11> <f1> SPC r` • `<f12> <f1>`	**(pel-cfg-pkg-reST** &optional OTHER-WINDOW)	Customize PEL Rest support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open rst group.
Graphviz DOT	• `<f11> <f1> SPC g` • `<f12> <f1>`	**(pel-cfg-pkg-graphviz-dot** &optional OTHER-WINDOW)	Customize PEL Graphviz-Dot support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open graphviz group.
PlantUML	• `<f11> <f1> SPC u` • `<f12> <f1>`	**(pel-cfg-pkg-plantuml** &optional OTHER-WINDOW)	Customize PEL PlantUML support. • If OTHER-WINDOW is non-nil (use **C−u**), display in another window and open plantuml-mode group.
Input Completion Mode Selection (See also: ∑ Input Completion)	PEL supports several input completion modes that kick in with the `M−x`, `C−x b`, `C−x C−f`, `<f1> o` and many other commands. PEL supports the following input completion modes: 1. Emacs' default tab completion 2. 📦 Helm mode completion : 📝 set **pel-use-helm** to **t**. 3. Ido mode completion : 📝 set **pel-use-ido** to **t** 4. 📦 Ivy mode completion : 📝 set **pel-use-ivy** to **t** 5. 📦 Ivy mode completion with Counsel mode : 📝 set **pel-use-counsel** to **t** 6. 📦 Ido/Helm mode where Ido is used for dealing with Files and buffers and Helm is used everywhere else (including all Helm specific commands). • Use `<f11><f1> c`, to customize the PEL completion group user options above. • Set the **pel-initial-completion-mode** user option to select which completion mode is used when Emacs starts. As soon as one of the extra completion mode is activated via the corresponding pel-use- user option, PEL makes the following commands available to change the completion mode and to see which one is currently active.		
Select the completion mode	`<f11> M−c`	**(pel-select-completion-mode)**	Prompt user for completion mode to activate. The available modes depend on what is currently activated by customization. See the list above.
Show what completion mode is currently used.	`<f11> ? c`	**(pel-show-active-completion-mode)**	Display the completion mode currently used.
Search Tools Selection (See also: ∑ Search/Replace)	PEL supports several search tools that impact the way the `C−s` command operates. PEL supports the following search tools: • Emacs' default ISearch • 📦 Anzu, ISearch with match count : 📝 set **pel-use-anzu** to **t**. • 📦 Swiper search with overview match list : 📝 set **pel-use-swiper** to **t** • Use `<f11><f1> s`, to customize the PEL completion group user options above. • Set the **pel-initial-search-tool** user option to select which search tool is used when Emacs starts. As soon as one of the extra search tool is activated via the corresponding pel-use- user option, PEL makes the following commands available to change the currently used search tool and to see which one is currently active.		
Select search tool to use	`<f11> s s`	**(pel-select-search-tool)**	Prompt user for search tool to use with **C−s**. Show new active one. • Emacs normally maps the search-forward command to **C−s**. • PEL provides the ability to activate the following tools that can be activated for searching: • 📦 The Anzu external package 📝 activated by **pel-use-anzu** user option. Anzu provides a match count in the modeline when search command is used. • 📦 The Swiper external package 📝 activated by **pel-use-swiper** user option. Swiper is not using isarch-forward; it shows a list of matching lines in the mini-buffer. • Use the `<f11> <f1> s` command to open the PEL search customize group and set the **pel-initial-search-tool** user option to identify which tool is used when Emacs starts. See the ∑Customize table for more information. ☝️Being able to search using either Emacs default ISearch (see below) and Swiper helps as they are both very useful in different scenarios.
Show which search tool is currently used	`<f1> ? s`	**(pel-show-active-search-tool)**	Display the currently used search tool.