Customizing Emacs with PEL

Operation	Keystroke	Function	Note

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.
	This table shows how to quickly gain access to the customized data using commands that open buffers that show the customized data inside buffers that operate in the Customize mode with special key bindings to speed up operation in that mode.
	• The first section shows navigation commands available inside a buffer that shows customized data (also called user options).
	• The later sections show commands that you can use to open buffers in Customization Mode to manage user options of interest.
	
	PEL - Configuration through Customization
	• 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.		

Open this PDF file. See also: ∑ Help/Info	**\<f11\> \<f2\> \<f1\>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the local copy of the ∑ **Customize** PDF file unless a command prefix (like **C-u**) was used. In that case it opens the Github-hosted file instead.

Customization Data	By default Emacs stores the customization data inside the Emacs init.el file, along with your other configuration, as Lisp code inside a **custom-set-variable** form.
	• When using PEL, and perhaps even if you're not, it's best to have Emacs store this data inside a *separate file* that you can put under VCS control independently from your init.el file. PEL promotes storing it inside the file **~/.emacs.d/emacs-customization.el** .
	• Store the following Emacs Lisp code snippet inside your init.el file to do so:
	
	     (setq custom-file "~/.emacs.d/emacs-customization.el")
	     (load custom-file)
	
	• When using PEL, that code must be located before the call to pel-init.
	• The use of a file to store Emacs customization data separate from the Emacs init.el file provides another degree of freedom and flexibility. It becomes possible to easily distribute a configuration to several computers (or even users) without sharing more private information that can be kept inside the init.el file.		

Customize Mode	This section describes commands available in buffer operating in Customize-mode showing the various user options you got access to using the commands described in the sections below.		
Move to Avy/Ace target See also:∑ Navigation	**o**	(**ace-link-custom**)	1. Highlight each target with an Avy/Ace single or double letter target. 2. Type the letter(s) to move to that position. • This is a **very efficient and quick navigation** mechanism. • 📦Requires **ace-link external package** 🔗 PEL downloads, installs and activates it when the **pel-use-ace-link** user option is set to **t**.
Apply customization changes	**C-c C-c**	(**Custom-set** &rest IGNORE)	Set the current value of all edited settings in the buffer.
Apply and Save customization changes	**C-x C-s**	(**Custom-save** &rest IGNORE)	Set all edited settings, then save all settings that have been set. • If a setting was edited and set before, this saves it. If a setting was merely edited before, this sets it then saves it.
Quit Customization and close buffer	**q**	(**Custom-buffer-done** &rest IGNORE)	Exit current Custom buffer according to 'custom-buffer-done-kill'.
Browse customize data tree	The following commands create a tree browser for the customize hierarchy inside a *Customize Browser* buffer. Each node can we expanded down to a single options and any can be collapsed.		
Browse complete customize data tree from root: Emacs	**\<f11\> \<f2\> M-b**	(**customize-browse** &optional GROUP)	Open the customize tree bowser for the entire Emacs customization data already loaded. ⚠ Emacs is only able to show information it knows about. Customization data defined in files not loaded will not be accessible.
Browse PEL customize data tree	**\<f11\> \<f2\> P M-b**	(**pel-customize-browse**)	Open the customize tree bowser for the entire PEL customization data (which is under Emacs/Convenience.
Emacs Easy Customization	With the following command you can gain access to the Customize-mode to customize anything of interest. 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\> \<f2\> 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. • ⚠ **Until a package is loaded its customization group is unknown to Emacs** and no buffer will be opened for it. To see the customization group, first load the package via one of its command that is auto-loaded or load it explicitly. • ☝ Note however that the PEL commands that open customization groups attempt to identify the library where the customization group is defined and will prompt you to load the related library to enable access to the customization group. The groups accessible via the PEL commands are limited to what PEL supports.		
Customize Emacs	**\<f11\> \<f2\> c**	(**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. ⚠ Emacs is only able to show information it knows about. Customization data defined in files not loaded will not be accessible.
Customize a specific group	**\<f11\> \<f2\> 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. • This command provides completion and you can use it to detect groups. ⚠ Emacs is only able to show the name names of groups that are defined in files that have already been loaded. You won't be able to open a group that is not already loaded.
Customize a user option	**\<f11\> \<f2\> o**	(**customize-option** SYMBOL)	Customize SYMBOL, which must be a user option. • As with groups, Emacs provides completion for user options, allowing you to detect user options. ⚠ Emacs is only able to show the name names of user options that are defined in files that have already been loaded. You won't be able to open a group that is not already loaded.

Operation	Keystroke	Function	Note

Input Completion Mode Selection
 	
See also:
∑ **Completion/Input**
	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> <f2> M-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> ? M-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> s <f3>` 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.		

Show which search tool is currently used	`<f1> ? s`	(**pel-show-active-search-tool**)	Display the currently used search tool.

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> s <f2>` 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.
			   👆 Being able to search using either Emacs default ISearch (see below) and Swiper helps as they are both very useful in different scenarios.

Customize PEL support	The following commands opens the Emacs customization group related to a PEL topic. Most of these commands do not prompt; they open the customization buffer at the requested group. 👆 If you prefix the following commands with `C-u` PEL will also open the customization groups related to the specific feature. ⚠️👆 To activate any PEL customization change in the current session, execute `M-x pel-init` after you saving and applying the customized variable. For motion variables that control mode hooks (eg. the flyspell automatic activation for specific major modes), you also need to restart Emacs.		
All PEL	`<f11> <f2> P !`	(**pel-cfg** &optional OTHER-WINDOW)	Customize PEL support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
	The following key bindings almost all use the same PEL command: (**pel-customize-pel** &optional OTHER-WINDOW). The command detects the key sequence that invoked it to select the customization group to open. If there are more than one it prompts for the one to open. If a group is not loaded, PEL prompts for loading it. • All of these commands open the buffer inside another window if a prefix argument (like `C-u`) is typed first.		
∑ **Align**	`<f11> t a <f2>`	Customize PEL support for text alignment.	
∑ **Auto-Completion**	`<f11> , <f2>`	Customize PEL auto-completion support: auto-complete, company and hippie-expand.	
∑ **Bookmarks**	`<f11> ' <f2>`	Customize PEL support for bookmark groups: bookmark, bm.	
∑ **Buffers**	`<f11> b <f2>`	Customize PEL support for buffer management: hexl.	
∑ **Comments**	`<f11> ; <f2>`	Customize Emacs support for comment hide control: hide-cmnt.	
∑ **Cursor**	`<f11> m <f2>`	Customize PEL support for cursor and multiple-cursors.	
∑ **Diff & Merge**	`<f11> d <f2>`	Customize PEL support for diff: ztree.	
∑M **Dired**	`<f11> f <f2> 2`	Customize PEL support for dired, directory editor.	
∑ **Drawing**	`<f11> D <f2>`	Customize PEL drawing mode support.	
∑ **File-mngt**	`<f11> f <f2> 1`	Customize PEL support for file management.	
∑ **File-mngt - dir. tree browser**	`<f11> B <f2>`	Customize PEL support for directory tree browsers: treemacs, ztree.	
∑ **File-mngt - NeoTree**	`<f11> B N <f2>`	Customize PEL support for NeoTree directory browser	
∑ **Frames**	`<f11> F <f2>`	Customize PEL frame management support.	
∑ **Grep**	`<f11> g <f2>`	Customize PEL grep support. Groups: grep, ag, rg, ripgrep, wgrep.	
∑ **Highlight**	`<f11> b h <f2>`	Customize PEL support for buffer highlight management: parens, printer, rainbow-delimiters.	
∑ **Inserting Text**	`<f11> i <f2>`	Customize PEL text insertion support: lice, smart-dash, tempo, time-stamp, yasnippet	
∑ **Keyboard Macros**	• `<f11> k <f2>` • `<f11> k e <f2>` • `<f11> k l <f2>`	Customize the PEL keyboard macro external package support: centimacro, emacros, elmacro.	
Input Completion: ∑ **Completion/Input**	`<f11> <f2> P M-c 1`	(**pel-cfg-pkg-completion** &optional OTHER-WINDOW)	Customize PEL Input Completion support. • If OTHER-WINDOW is non-nil (use `C-u`), display in other window.
∑ **Key-Chords**	`<f11> <f2> P M-K`	(**pel-cfg-pkg-key-chord** &optional OTHER-WINDOW)	Customize PEL Key Chord support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.

Operation	Keystroke	Function	Note
∑ **Navigation**	`<f11> <f2> P n`	**(pel-cfg-pkg-navigation** &optional OTHER-WINDOW)	Customize PEL and Emacs navigation tools support. Provides access to the following customization groups: 1. **PEL project management** 2. **avy** • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
∑ **Projectile**	• `<f11> <f2> P <f8>`	**(pel-cfg-pkg-project-mng** &optional OTHER-WINDOW)	Open the projectile customization group where you can modify projectiles configuration. • The key sequence `<f11> <f2> P <f8>` is always available, the others are only available when the projectile mode is activated.
	• `<f11> <f8> <f2>` • `<f8> <f2>`	**(pel-customize-pel** &optional OTHER-WINDOW)	📦 Available when the **projectile** external package is 🔲 activated by PEL with the **pel-use-projectile** user option is non-nil.
∑ **Scrolling**	`<f11> | <f2>`	Customize PEL Scrolling support.	
∑ **Search/Replace**	`<f11> s <f2>`	Customize PEL basic search support.	
Regular Expression ∑ **Search/Replace**	`<f11> s x <f2>`	Customize PEL regular expression tool support.	
∑ **Sessions**	`<f11> S <f2>`	Customize PEL Session support.	
∑ **Shells**	`<f11> x <f2>`	Customize PEL Shell support.	
∑ **Speedbar**	`<f11> M-s <f2>`	Customize PEL Speedbar support.	
∑ **Spell Checking**	`<f11> $ <f2>`	Customize PEL support for: spell checking. Identify which major modes will automatically activate either flyspell-mode or flyspell-prog-mode.	
∑ **Tags - cross reference**	`<f11> X <f2>`	Customize PEL cross-reference support: ctags/etags/gtags	
∑ **Text Modes**	`<f11> t m <f2>`	Customize PEL text management support.	
∑ **Undo/Redo/Repeat/Arg**	`<f11> u <f2>`	Customize PEL undo support.	
∑ **VCS**	`<f11> v <f2>`	Customize PEL Version Control System support.	
∑ **Windows**	`<f11> w <f2>`	Customize PEL Window support.	
Yasnippet - ∑ **Inserting Text**	`<f11> y <f2>`	Customize PEL Yasnippet text insertion support.	
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> SPC` key prefixes are available globally (for all buffers). • The `<f12> <f2>` 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 audio narration	• `<f11> SPC a <f2>` • `<f12> <f2>`	Customize PEL Applescript support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- C**	• `<f11> SPC c <f2>` • `<f12> <f2>`	Customize PEL C support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- C++**	• `<f11> SPC C <f2>` • `<f12> <f2>`	Customize PEL C++ support: cpp. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- D**	• `<f11> SPC D <f2>` • `<f12> <f2>`	Customize PEL D support: d-mode. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ𝑀**- Lispy**	• `<f11> <f2> SPC M-L`	**(pel-cfg-pkg-lisp** &optional OTHER-WINDOW)	Customize support for Lisp programming languages - A group that also contains the groups for Emacs Lisp and Common Lisp: lispy. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
⅀ℬℓ **- Emacs Lisp**	• `<f11> SPC l <f2>` • `<f12> <f2>`	Customize PEL Elisp support: elint, eldoc, lispy. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- Common Lisp**	• `<f11> SPC L <f2>` • `<f12> <f2>`	Customize PEL Lisp support: lisp, lispy. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- Elixir**	• `<f11> SPC x <f2>` • `<f12> <f2>`	Customize PEL Elixir support: alchemist, alchemist-iex. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- Erlang**	• `<f11> SPC e <f2>` • `<f12> <f2>`	Customize PEL Erlang support: erlang, erldoc, edts, auto-highlight-symbol. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- Forth**	• `<f11> SPC f <f2>` • `<f12> <f2>`	Customize PEL Forth support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- Julia**	• `<f11> SPC j <f2>` • `<f12> <f2>`	Customize PEL Julia support: julia, julia-mode, julia-snail. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- Python**	• `<f11> SPC p <f2>` • `<f12> <f2>`	Customize PEL Python support: python, python-flymake. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
ℬℓ **- REXX**	• `<f11> SPC R <f2>` • `<f12> <f2>`	Customize PEL REXX support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
Configure PEL Markup support	The following commands opens the Emacs customization group related to configure PEL support for the specific markup language. • The `<f11> SPC` key prefixes are available globally (for all buffers). • The `<f12> <f2>` 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.		
𝑀 **reStructuredText**	• `<f11> SPC r <f2>` • `<f12> <f2>`	Customize PEL reStructuredText support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
𝑀 **Graphviz Dot**	• `<f11> SPC g <f2>` • `<f12> <f2>`	Customize PEL Graphviz-Dot support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
𝑀 **PlantUML**	• `<f11> D u <f2>` • `<f11> SPC u <f2>` • `<f12> <f2>`	Customize PEL PlantUML support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.	
Customize Specific Emacs Groups.	PEL provides several key bindings to open customization groups of Emacs built-in or external package. • PEL will prompt you to load their specific file if they are not loaded. • Most of the key bindings are mapped into the PEL key prefixes as the `<f3>` key member. For example to open auto-completion related groups you can use the `<f11> , <f3>` key sequence. These are not listed here. • PEL does not provide key prefixes for all Emacs concepts. It provides, however some key bindings to access the customization buffer for some of those. They are listed just below, here:		

Operation	Keystroke	Function	Note
Permanently change the cursor's color See also: ∑ Cursor	`<f11> <f2> E C-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. ⚠ Only available in graphics mode.
locate	`<f11> <f2> E l`	(pel-cfge-locate &optional OTHER-WINDOW)	Customize locate. With **C-u**, display in another window.
man	`<f11> <f2> E m`	(pel-cfge-man &optional OTHER-WINDOW)	Customize man. With **C-u**, display in another window.
browse-url	`<f11> <f2> E u`	(pel-cfge-browse-url &optional OTHER-WINDOW)	Customize browse-url. With **C-u**, display in another window.
webjump	`<f11> <f2> E j`	(pel-cfge-webjump &optional OTHER-WINDOW)	Customize webjump. With **C-u**, display in another window.
woman	`<f11> <f2> E w`	(pel-cfge-woman &optional OTHER-WINDOW)	Customize woman. With **C-u**, display in another window.
Customize Emacs Libraries	The following key bindings almost all use the same PEL command: (**pel-customize-library** &optional OTHER-WINDOW). The command detects the key sequence that invoked it to select the customization group to open. If there are more than one it prompts for the one to open. If a group is not loaded, PEL prompts for loading it. If the related package is not installed PEL print a warning message. • For external packages you can use the same key sequence except for the last key: replace `<f3>` by `<f2>` : that sequence will open the PEL configuration buffer for the same topic. From that you will find the PEL option variable to activate the external package. • All of these commands open the buffer inside another window if a prefix argument (like **C-u**) is typed first.		
∑ Align	`<f11> t a <f3>`	Customize Emacs text alignment support: open the align group.	
∑ Auto-Completion	`<f11> , <f3>`	Customize Emacs auto-completion support: auto-complete, company and hippie-expand.	
∑ Bookmarks	`<f11> ' <f3>`	Customize Emacs bookmark group which includes: bookmark and bm.	
∑ Buffers	`<f11> b <f3>`	Customize Emacs support for buffer management: Buffer-menu, buffer, minibuffer, hexl, nhexl.	
∑ Comments	`<f11> ; <f3> 1`	Customize Emacs support for comments: comment, hideshow.	
Customization Control	`<f11> <f2> <f3>`	Customize Emacs customization control.	
∑ Hide/Show	`<f11> ; <f3> 2`	Customize Emacs support for comments: comment, hideshow.	
Input Completion: ∑ Completion/Input	`<f11> <f2> P M-c`	(pel-cfg-pkg-completion &optional OTHER-WINDOW)	Customize Emacs Input Completion support: helm, ido, ivy, counsel.. • If OTHER-WINDOW is non-nil (use **C-u**), display in other window.
∑ Cursor	`<f11> m <f3>`	Customize Emacs support for cursor and multiple-cursors.	
∑ Diff & Merge - ediff	`<f11> d e <f3>`	Customize Emacs ediff.	
∑M Dired	`<f11> f <f3> 2`	Customize Emacs support for dired, directory editor. Other choices are available for neotree and ztree.	
∑ Enriched Text	`<f11> t e <f3>`	Customize Emacs Enriched Text support.	
∑ File-mngt	`<f11> f <f3> 1`	Customize Emacs support for file management.	
∑ File-mngt - auto-revert	`<f11> f r <f3>`	Customize Emacs support for file automatic revert management.	
∑ File-mngt - ffap	`<f11> f a <f3>`	Customize Emacs support for management of ffap (find file at point).	
∑ File-mngt - dir. tree browser	`<f11> B <f3>`	Customize directory tree browsers: treemacs, ztree	
∑ File-mngt - NeoTree	`<f11> B N <f3>`	Customize NeoTree directory browser	
∑ Filling/Justification	• `<f11> t f <f3>` • `<f11> t j <f3>`	Customize Emacs fill and justification control.	
∑ Frames	`<f11> F <f3>`	Customize Emacs frame management support.	
∑ Grep	`<f11> g <f3>`	Customize Emacs grep support. Groups: grep, ag, rg, ripgrep, wgrep.	
∑ Highlight	`<f11> b h <f3>`	Customize Emacs support for buffer highlight management: auto-highlight, edit, rainbow-delimited, line, fill-column-indicator (for Emacs version earlier than 27.1)	
∑ Indentation	`<f11> <tab> <f3>`	Customize Emacs indentation. Opens the indent customization group.	
∑ Inserting Text	`<f11> i <f3>`	Customize Emacs text insertion support: lice, smart-dash, tempo, time-stamp, yasnippet	
∑ Keyboard Macros	`<f11> k <f3>`	Customize the Emacs keyboard macro external package support: kmacro, centimacro.	
∑ Keyboard Macros	`<f11> k e <f3>`	Customize the Emacs keyboard macro external package support: emacros.	
∑ Keyboard Macros	`<f11> k l <f3>`	Customize the Emacs keyboard macro external package support: elmacro.	
Line Mngt: ∑ Display - Lines	`<f11> l <f3>`	Customize Emacs support for visual-line.	
∑ Menus	`<f11> <f10> <f3>`	Customize Emacs menu mechanisms.	
∑ Navigation	`<f11> <f2> P n 2`	(pel-cfg-pkg-navigation &optional OTHER-WINDOW)	Customize Emacs navigation tools support: avy. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
∑ Projectile	• `<f11> <f8> <f3>` • `<f8> <f3>`	(pel-customize-projectile)	Open the projectile customization group where you can modify projectiles configuration. • Key sequence `<f11> <f8> <f3>` is available if **pel-use-projectile** is **t**. • Key sequence `<f8> <f2>` is available when the projectile mode is on. 📦 Available when the **projectile** external package is 🔲 activated by PEL with the **pel-use-projectile** user option is non-nil.
Regular Expression ∑ Search/Replace	`<f11> s x <f3>`	Customize Emacs regular expression support: rxt, re-builder, visual-regex.	
∑ Scrolling	`<f11> | <f3>`	Customize Emacs Scrolling support groups: follow, smooth-scrolling.	
∑ Search/Replace	`<f11> s <f3>`	Customize Emacs Search support: search, anzu, swiper, iedit.	
∑ Sessions	`<f11> S <f3>`	Customize Emacs Session support: desktop.	
∑ Shells	`<f11> x <f3>`	Customize Emacs Shells support groups: term, terminals, vterm.	
∑ Speedbar	`<f11> M-s <f3>`	Customize Emacs Speedbar support.	
∑ Spell Checking	`<f11> $ <f3>`	Customize Emacs spelling support. Opens the following customization groups: ispell, flyspell.	
∑ Tags - cross reference	`<f11> X <f3>`	Customize Emacs cross-reference support: ctags/etags/gtags	
Text ∑ Whitespace	`<f11> t w <f3>`	Customize Emacs handling of whitespaces.	
∑ VCS	`<f11> v <f3>`	Customize Emacs Version Control System support: vc, vc-hg, vc-git, magit, monky.	

Operation	Keystroke	Function	Note
∑ **Undo/Redo/Repeat/Arg**	`<f11> u <f3>`	Customize Emacs undo support: undo, undo-tree.	
∑ **Windows**	`<f11> w <f3>`	Customize Emacs Window support groups: windows, ace-window, ace-window-display, winner, windmove.	
Yasnippet ∑ **Inserting Text**	`<f11> y <f3>`	Customize Yasnippet groups: yasnippet, yasnippet-snippets, yas-minor	
Configure Emacs Programming Language support	The following commands opens the Emacs configuration group to configure Emacs support for the specified programming language. • The `<f11> SPC` key prefixes are available globally (for all buffers). • The `<f12> <f3>` key is only available when point is in a buffer for one of the languages supported by PEL and open the Emacs customization group for the programming language for the current buffer.		
AppleScript & text audio narration	• `<f11> SPC a <f3>` • `<f12> <f3>`	Customize Emacs Applescript support. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℬℓ **- C**	• `<f11> SPC c <f3>` • `<f12> <f3>`	Customize Emacs C support. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℬℓ **- C++**	• `<f11> SPC C <f3>` • `<f12> <f3>`	Customize Emacs C++ support: cpp. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℬℓ **- D**	• `<f11> SPC D <f3>` • `<f12> <f3>`	Customize Emacs D support: d-mode. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℒℬℓ **- Emacs Lisp**	• `<f11> SPC l <f3>` • `<f12> <f3>`	Customize Emacs Elisp support: elint, eldoc, lispy. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℬℓ **- Common Lisp**	• `<f11> SPC L <f3>` • `<f12> <f3>`	Customize Emacs Lisp support: lisp, lispy. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℬℓ **- Elixir**	• `<f11> SPC x <f3>` • `<f12> <f3>`	Customize Emacs Elixir support: alchemist, alchemist-iex. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℬℓ **- Erlang**	• `<f11> SPC e <f3>` • `<f12> <f3>`	Customize Emacs Erlang support: erlang, erldoc, edts, auto-highlight-symbol. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℬℓ **- Julia**	• `<f11> SPC j <f3>` • `<f12> <f3>`	Customize Emacs Julia support: julia, julia-mode, julia-snail. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℬℓ **- Python**	• `<f11> SPC p <f3>` • `<f12> <f3>`	Customize Emacs Python support: python, python-flymake. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
Configure PEL Markup support	The following commands opens the Emacs customization group related to configure Emacs support for the specific markup language. • The `<f11> SPC` key prefixes are available globally (for all buffers). • The `<f12> <f3>` key is only available when point is in a buffer for one of the languages supported by PEL and open the Emacs customization group for the markup language for the current buffer.		
ℳ **reStructuredText**	• `<f11> SPC r <f3>` • `<f12> <f3>`	Customize Emacs reStructuredText support. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℳ **Graphviz Dot**	• `<f11> SPC g <f3>` • `<f12> <f3>`	Customize Emacs Graphviz-Dot support. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	
ℳ **PlantUML**	• `<f11> D u <f3>` • `<f11> SPC u <f3>` • `<f12> <f3>`	Customize Emacs PlantUML support. • If OTHER-WINDOW is non-nil (use **C–u**), display in another window.	