PEL Key Maps

Operation	Keystroke	Key Map	Note
Emacs Key Bindings See also: **∑ ⌨ Modifier Keys**	Emacs has a large set of key bindings. • Some commands are bound to single keys like the **`a`** key which normally inserts the letter ‘a’ in the current buffer. • Some commands are bound to functions keys like **`<f1>`** or use key modifiers like **`C-a`** or **`M-a`**. See **∑ ⌨ Modifier Keys** for more info. • Some commands are bound to longer key sequences lie **`C-x s`**. • The first key, or the first set of keys, can be used as an Emacs key prefix. And then several other keys can follow, all under that prefix. The prefix creates some sort of scope: the key-map under that prefix. • There’s really no limit to the way you can combine keys, the modifier keys, with or without short or longer key prefixes. • On top of that you can have key bindings that are • global, always accessible if the related code was loaded, or • local, only available while a specific major or minor mode is activated inside a specific buffer. • All of this provides great flexibility. But it makes Emacs more difficult to learn: you need to remember all the keys.		
PEL Key maps See also: **⌨Keys - Fn**	Although PEL itself adds a large amount of keys to what’s already in Emacs, it leaves most Emacs key binding intact and mainly uses the function keys organized under a tree of key prefixes, trying to provide easy-to-remember key prefixes. • PEL key bindings are accessible from Emacs running in graphics mode and in terminal mode (you may have to configure your termcap terminal software to support ASNI key sequences for function and cursor keys). • By default, PEL also activates the **which-key external package** which allows you to see all command key bindings for each key prefix in the echo area at the bottom of your Emacs screen. • PEL provides documentation of the Emacs and PEL key bindings, organized in topics inside PEL files such as this one. • All PEL key prefix groups provide a **`<f1>`** key binding to a command that opens a local copy of a PDF file describing the topic. To open this PDF file from Emacs using PEL, just type **`<f11> <f1>`**. The **`<f11>`** key is the most often used PEL global key prefix. Inside its group the **`<f1>`** key opens this file. This page lists PEL’s key maps. • Column 1, the title column, shows the name of the PEL specific PDF page and it’s also a link to the Github hosted pdf page. • Column 2 shows the key sequence for the topic. • Column 3 shows the name of PEL key prefix for the topic. Some topics do not have commands organized under on specific PEL key map, but the commands and keys are described inside topic specific PDF tables. These are listed first set of rows below. ☝ **Firefox** will open the PDF files and will render it inside the browser page instead of downloading it. This is a great way to navigate through the various links if you are online. For other browsers, you may have to install pdf rendering plugins to do the same.		
Topics with no PEL key maps	The following topics do not have a PEL topic-specific key-map. You can use the **`<f11> ? p`** key sequence and enter the topic name to open the file. The command support tab completion. See **∑ Help/Info**		
➢Legend	Describes all conventions and symbols used in the PEL PDF files.		
Ⅿ AsciiDoc	AsciiDoc support		
∑ Autosave/Backup	Emacs commands for autosave and backup control		
∑ Case Conversions	Commands for case conversion of text.		
∑ Closing/Suspending	Commands to close or suspend Emacs.		
∑ Completion/Input	Commands to complete user input at prompts.		
∑Ⅿ CUA	CUA mode commands.		
∑ Enriched Text	Commands that support the enriched text concept.		
∑ ERT	Emacs Lisp unit testing commands.		
∑ Faces/Fonts	Commands that control Emacs faces and fonts.		
∑ Key-Chords	Commands to enable/disable key chords (typing 2 normal keys together to invoke a command).		
⌨Keys - Fn	Table that shows the way PEL uses function keys.		
Ⅿ Outline/Org-Mode	Org-mode commands.		
∑ ⌨ Modifier Keys	Describes Emacs modifier keys and ways of describing keys in Emacs.		
∑ Mouse	Mouse commands. Available both in graphics and terminal modes.		
∑ Narrowing	Narrowing commands. A way to narrow your view to only a portion of the current buffer, protecting the rest of the buffer from any modification.		
∑ Navigation	The navigation commands available in Emacs with the additions provided by PEL and other packages.		
∑⌨ Numkeypad	Describes the way the numerical keypad is handled in Emacs.		
∑ Packages	Commands to download and manipulate external packages.		
∑ Rectangles	Commands to manipulate rectangle areas of text inside a buffer.		
∑ Semantic	🚧 Planned topic		
∑ SyntaxCheck	🚧 Planned topic		
Global Key Maps	The key maps are listed in order of the key they use. The keys were selected mnemonic naming as much as possible. For that reason some key maps are accessible via several key prefix sequences.		
Top level prefix	**`<f11>`**	**pel:**	Key prefix
∑ Indentation	**`<f11> TAB`**	**pel:indent**	
∑ Spell Checking	**`<f11> $`**	**pel:spell**	
∑ Bookmarks	**`<f11> ‘`**	**pel:bookMark**	
∑ Auto-Completion	**`<f11> ,`**	**pel:auto-completion**	
∑ Cut & Paste - Kill	**`<f11> -`**	**pel:kill**	Kill (cut) operations
∑ Marking	**`<f11> .`**	**pel:mark**	
• **∑ Comments** • **∑ Hide/Show**	**`<f11> ;`**	**pel:comment**	
∑ Cut & Paste - Copy	**`<f11> =`**	**pel:copy**	Copy operations
∑ Help/Info	**`<f11> ?`**	**pel:help**	
	`<f11> ? a`	**pel:apropos**	
	`<f11> ? d`	**pel:describe**	
	`<f11> ? e`	**pel:emacs**	
	`<f11> ? i`	**pel:info**	

Operation	Keystroke	Key Map	Note
	`<f11> ? k`	pel:keys	
∑ File-mngt	`<f11> B`	pel:browse	Directory tree browsing (for now: it will evolve) 🚧
∑ File-mngt - NeoTree	`<f11> B N`	pel:neotree	NeoTree directory tree browser
∑ Cut & Paste - OS Clipboard	`<f11> C`	pel:clipboard	
∑ Drawing	`<f11> D`	pel:draw	
Ϻ PlantUML	`<f11> D u`	pel:plantuml	
∑ Frames	`<f11> F`	pel:frame	
∑ Sessions	`<f11> S`	pel:session	
∑ Tags - Cross References	`<f11> X`	pel:xref	
∑ Inserting Text - underlining	`<f11> _`	pel:underline	Underline text with specified character.
∑ Abbreviations	`<f11> a`	pel:abbrev	
∑ Buffers	`<f11> b`	pel:buffer	
∑ Buffers	`<f11> b I`	pel:indirect-buffer	
∑ Highlight	`<f11> b h`	pel:highlight	
∑ Counting	`<f11> c`	pel:count	Counting text elements in current buffer
∑ Diff & Merge	`<f11> d`	pel:diff	
∑ Diff & Merge	`<f11> d e`	pel:ediff	
• ∑ File-mngt • ∑Ϻ Dired • ∑ Web	`<f11> f`	pel:file	File & directory management
• ∑ File-mngt • ∑Ϻ Dired	`<f11> f a`	pel:ffap	
∑ File-mngt	`<f11> f r`	pel:file-revert	
∑ File/Directory Variables	`<f11> f v`	pel:filevar	
∑ Grep	`<f11> g`	pel:grep	
∑ Grep - with ag	`<f11> g a`	pel:ag	Grep operations with **ag**, the silver searcher (a fast grep alternative)
∑ Grep - with ag	`<f11> g a p`	pel:ag-project	ag commands to search in project-related files
∑ Grep - with ag	`<f11> g a d`	pel:ag-dired	ag commands to teach for file names and spend the list in dired buffer
∑ Grep - with ag	`<f11> g a k`	pel:ag-kill	ag command to kill buffer and process
∑ Inserting Text	`<f11> i`	pel:insert	
∑ Keyboard Macros	`<f11> k`	pel:kbmacro	Emacs keyboard macros, centimacro, emacros, elmacros.
∑ Keyboard Macros - emacros	`<f11> k e`	pel:emacros	
∑ Keyboard Macros - elmacros	`<f11> k l`	pel:elmacros	
∑ Display - Lines	`<f11> l`	pel:linectrl	
∑ Cursor	`<f11> m`	pel:mcursor	Multiple cursor editing.
∑ Sorting	`<f11> o`	pel:order	Ordering/Sorting.
∑ Registers	`<f11> r`	pel:register	
∑ Search/Replace	`<f11> s`	pel:search-replace	
	`<f11> s m`	pel:search-mode	
	`<f11> s w`	pel:search-word	
	`<f11> s x`	pel:regexp	
∑ Text Modes	`<f11> t`	pel:text	
∑ Align	`<f11> t a`	pel:align	
∑ Filling/Justification	`<f11> t f`	pel:fill	Text fill
	`<f11> t j`	pel:justification	Text justification
∑ Text Modes	`<f11> t m`	pel:text-modes	
∑ Transpose	`<f11> t t`	pel:text-transpose	
∑ Whitespace	`<f11> t w`	pel:text-whitespace	
∑ Undo/Redo/Repeat/Arg	`<f11> u`	pel:undo	
∑ VCS-Mercurial	`<f11> v`	pel:vcs	PEL also supports Git, a page dedicated for Git is not yet written
∑ Windows	`<f11> w`	pel:window	
∑ Windows	`<f11> w d`	pel:window-dedicated	
∑ Windows	`<f11> w s`	pel:window-size	
∑ Shells	`<f11> x`	pel:eXecute	
∑ Inserting Text	`<f11> y`	pel:yasnippet	Yasnippet text template insertion/expansion.
∑ Scrolling	`<f11> |`	pel:scroll	
∑ Customize	`<f11> <f2>`	pel:cfg	
	`<f11> <f2> SPC`	pel:cfg-pel-lang	
	`<f11> <f2> E`	pel:cfg-emacs	
	`<f11> <f2> P`	pel:cfg-pel	
∑ Projectile	`<f11> <f8>`	pel:projectile	
∑ Menus	`<f11> <f10>`	pel:menu	
∑ Speedbar	`<f11> M-s`	pel:speedbar	

Operation	**Keystroke**	**Key Map**	**Note**
Major mode specific key maps	PEL provides a set of global key-maps that are specific to major modes for markup and programming languages. The key maps have 2 set of bindings. • One set has a key prefix that uses `<f11> SPC` followed by a key identifying the language. • The other set is only available inside buffers that use the specific major mode and they all use the same `<f12>` key prefix, simulating a local mode prefix. • The following list is ordered by programming languages names (sorting all Lisp under L) and then listing the markup languages after.		
𝔓𝔏- AppleScript	• `<f11> SPC a` • `<f12>`	pel:for-applescript	
𝔓𝔏 - C	• `<f11> SPC c` • `<f12>`	pel:for-c	
𝔓𝔏 - C - C pre-processor	• `<f11> SPC c #` • `<f12> #`	pel:for-c-propoc	
𝔓𝔏 - C - C tempo skeleton	• `<f11> SPC c <f12>` • `<f12> <f12>`	pel:c-skel	Prefix for tempo skeletons for the C programming language.
𝔓𝔏 - C++	• `<f11> SPC C` • `<f12>`	pel:for-c++	
𝔓𝔏 - C++ - C pre-processor	• `<f11> SPC C #` • `<f12> #`	pel:for-c++-preproc	
𝔓𝔏 - D	• `<f11> SPC D` • `<f12>`	pel:for-d	
𝔓𝔏 - Elixir	• `<f11> SPC x` • `<f12>`	pel:for-elixir	
𝔓𝔏 - Erlang	• `<f11> SPC e` • `<f12>`	pel:for-erlang	
𝔓𝔏 - Erlang	• `<f11> SPC e a` • `<f12> a`	pel:erlang-analysis	🚧 Planned
𝔓𝔏 - Erlang - clause	• `<f11> SPC e c` • `<f12> c`	pel:erlang-clause	
𝔓𝔏 - Erlang - debug	• `<f11> SPC e d` • `<f12> d`	pel:erlang-debug	
𝔓𝔏 - Erlang - functions	• `<f11> SPC e f` • `<f12> f`	pel:erlang-function	
𝔓𝔏 - Erlang - tempo skeletons	• `<f11> SPC e <f12>` • `<f12> <f12>`	pel:erlang-skel	Prefix for tempo skeletons for the Erlang programming language.
𝔓𝔏 - Forth	• `<f11> SPC f` • `<f12>`	pel:for-forth	
𝔓𝔏 - Javascript	• `<f11> SPC i` • `<f12>`	pel:for-javascript	Experimental support for Javascript
𝔓𝔏 - Julia	• `<f11> SPC j` • `<f12>`	pel:for-julia	
𝔓𝔏 - Common Lisp	• `<f11> SPC L` • `<f12>`	pel:for-lisp	
𝔏𝔓𝔏 - Emacs Lisp	• `<f11> SPC l` • `<f12>`	pel:for-elisp	
𝔏𝔓𝔏 - Emacs Lisp - help	• `<f11> SPC l ?` • `<f12> ?`	pel:elisp-help	
𝔏𝔓𝔏 - Emacs Lisp - analyze	• `<f11> SPC l a` • `<f12> a`	pel:elisp-analyze	
𝔏𝔓𝔏 - Emacs Lisp - compile	• `<f11> SPC l c` • `<f12> c`	pel:elisp-compile	
𝔏𝔓𝔏 - Emacs Lisp - debug	• `<f11> SPC l d` • `<f12> d`	pel:elisp-debug	
𝔏𝔓𝔏 - Emacs Lisp - eval	• `<f11> SPC l e` • `<f12> e`	pel:elisp-eval	
𝔏𝔓𝔏 - Emacs Lisp - function	• `<f11> SPC l f` • `<f12> f`	pel:elisp-function	
𝔏𝔓𝔏 - Emacs Lisp - library	• `<f11> SPC l l` • `<f12> l`	pel:elisp-lib	
𝔏𝔓𝔏 - Emacs Lisp - tempo skeletons	• `<f11> SPC l <f12>` • `<f12> <f12>`	pel:elisp-skel	
𝔓𝔏 - Python	• `<f11> SPC p` • `<f12>`	pel:for-python	
𝔓𝔏 - REXX	• `<f11> SPC R` • `<f12>`	pel:for-rexx	
𝔓𝔏 - V	• `<f11> SPC v` • `<f12>`	pel:for-v	Experimental support for the emerging **V programming language**
𝔐 Graphviz Dot	• `<f11> SPC g` • `<f12>`	pel:for-graphviz-dot	
𝔐 PlantUML	• `<f11> SPC u` • `<f12>`	pel:for-plantuml	
𝔐 reStructuredText	• `<f11> SPC r` • `<f12>`	pel:for-reST	
𝔐 reStructuredText - adorn style	• `<f11> SPC r A` • `<f12> A`	pel:for-rst-adorn	
𝔐 reStructuredText - tempo skeletons	• `<f11> SPC r <f12>` • `<f12> <f12>`	pel:for-rst-skel	🚧 Planned
Other Function Keys	PEL also uses the function keys for other purpose. See the ⌨**Keys - Fn** table: it describes PEL's use of the functions keys with and without key modifiers.		

Operation	**Keystroke**	**Key Map**	**Note**
Move point to next visible bookmark	`<f2>`	(**bm-next**)	Not a prefix, a command: Move point to next visible bookmark. Activated only when **pel-use-bm** is set to t. See **∑ Bookmarks**.
Repeat last operation	`<f5>`	(**repeat** REPEAT-ARG)	Not a prefix, a command: Repeat most recently executed command. See **∑ Undo/Redo/Repeat/Arg**
Text Insertion	`<f6>`	**pel:f6**	
PEL Hydras	`<f7>`	**PEL Hydras**	The head of all PEL Hydras. Activated on first use. The PEL Hydras are described in: • **𝔓𝔩- AppleScript** • **∑ Hide/Show** • **∑ Windows**
∑ Projectile	`<f8>`	**projectile-command-map**	Activated by `<f11> <f8> <f8>` when **pel-use-projectile** is set to activate projectile.