↖ Topic Index

Getting Help / Apropos / Descriptions / Info Manuals / Queries

Description	Keystroke	Function	Note

Getting Help
○ PDF & customize
• Key bindings
• Packages, functions…
○ Apropos help
• Key sequence help
• short doc
○ info
○ Helpful, Log
• programming, extra topics
○ About Emacs
○ man, woman
• Emacs bugs report
○ PEL setup/used packages
○ PEL PDF Help
 	
⚠ **With Emacs under SSH** ☛
	Emacs is a *heavily* documented. All of this documentation is accessible from within Emacs: the manuals, the info page, the **docstrings** of functions and variables, the customization system. You can search for manual, topic, command, function, variable, object names, values inside variables.
	PEL also provides a **large set of topic-specific PDF files** such as this one (identified as 𝕀 **Help/Info**). See the ≻**Index** it has links to all PEL PDFs.
	• These PDFs are heavily hyper-linked to each other, to the Emacs manual and to external package home and description sites.
	• Use the *context sensitive* **pel-help-pdf** command to open the PDF of interest from within Emacs. That command can be invoked by:
	  • several global key sequences; each one identifies a specific PDF to open. These key sequences all start with **`<f11>`** and end with **`<f1>`**.
	  • with the **`<f12> <f1>`** local key sequence that open the PDF related to the buffer's major mode.
	  • For some of these key sequences, the command also supports one or several *secondary topics* ; these are mostly related to PDF describing the languages, but also some topics specific to complex minor modes. For example, in a make file using the GNU make syntax, the secondary topic is a description of the GNU make syntax. Inside an emacs-lisp buffer, the secondary topics are lispy and Emacs Lisp syntax.
	  • To select the secondary topic PDF, use a positive key command prefix with an absolute value greater than 1; such as **`C-u`** or **`M-2`** .
	By default the **pel-help-pdf** command opens a local PDF file with the local PDF reader. To open the GitHub hosted PDF web page instead use a **negative** prefix key. To open the main topic, use the **`M--`** prefix or the **`M--1`** prefix to the command. To open the secondary topic use **`M--2`**.
	The default behaviour can be modified by the following user-options:
	• **pel-flip-help-pdf-arg :** If set to **t**, the command opens the GitHub file with no (or positive) prefix and opens the local PDF file with negative prefix.
	• **pel-open-pdf-method :** Selects how to open the local PDF files: with PDF reader (default) or with the web browser identified by **pel-browser-used**.
	• **pel-browser-used :** Selects how the browsing mechanism: the default is to use the method identified by the **browse-url-browser-function** user-option. The alternative is to force using **Firefox** or Chrome, two browsers that can render the PEL PDF files. That greatly helps browsing the PDFs.
	• **browse-url-browser-function :** selects the default browsing behaviour, applied to all Emacs commands that browse files.
	• **pel-emacs-source-directory** : identifies the root directory of Emacs source code repo. Set it to permanently remember Emacs C source code.
	When running Emacs under a SSH session PEL prevents opening these PDF help files unless you set **pel-help-under-ssh** user-option to t.		

Last updated on: 2025-07-14	• **help on any symbol**: **`<f1> o`** • **info topic**: **`<f11> ? i a`**		• **Text in any elisp doctring**: **`C-u <f1> d`** • **Value in any symbol**: **`<f11> ? a u`**
Open this PDF file.	• **`<f11> ? <f1>`** • **`<f11> ? k <f1>`**	(**pel-help-pdf** &optional N)	Open the 𝕀 **Help/Info** local PDF. See argument description above.
𝕀 **Customize PEL Help Support**	• **`<f11> ? <f2>`** • **`<f11> ? k <f2>`**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL help support and syntax tools groups: pel-pkg-for-help, pel-pkg-syntax • If OTHER-WINDOW is non-nil (use **`C-u )`**, display in other window.
𝕀 **Customize Emacs Help Support**	• **`<f11> ? <f3>`** • **`<f11> ? k <f3>`**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs grep support. Groups: apropos, command-log, debbugs, help, helpful, hydra, keycast, info, interaction-log, man, minibuffer, which-func, which-key.
Emacs Reference Cards	Emacs has a set of short **PDF reference cards**, and next command can open it. 🎛 Access customization group with **`<f11> ? <f2>`** • 🛂 If PEL code cannot locate the directory you can identify it in the **pel-emacs-refcard-dirpath** user option.		
Open local copy of Emacs PDF reference card	**`<f11> ? e r`**	(**pel-open-emacs-refcard**)	Prompt for an Emacs REFCARD and open it. Supports tab completion. • Attempts to find the directory where the Emacs PDF reference card files are stored. Otherwise uses the directory identified by the **pel-emacs-refcard-dirpath** user option.
Emacs Help System	As described above, Emacs provides help for almost everything. The list of commands to access this information is shown in the following rows.		
• **Key bindings**	Get help/info on global or buffer local key bindings using the following commands. ☝ The **help-window-select** user-option controls if new window is selected.		
List all keys that belong to a prefix key	• *<prefix>* **`C-h`** • *<prefix>* **`<f1>`**		Type **C-h** (or **`<f1>`**) after the prefix keystroke to list all key bindings that belong to that prefix. For example to list all **C-x r** keys, type **C-x r C-h**
Print name of function invoked by key	• **`C-h c`** *<keys>* • **`<f1> c`** *<keys>*	(**describe-key-briefly** &optional KEY INSERT UNTRANSLATED)	Print the name of the function KEY invokes. KEY is a string.
Help on key binding	• **`C-h k`** *<keys>* • **`<f1> k`** *<keys>*	(**describe-key** &optional KEY UNTRANSLATED UP-EVENT)	Display documentation of the function invoked by KEY in the current context. • KEY can be any kind of a key sequence; it can include keyboard events, mouse events, and/or menu events.
Open Info manual describing the command for the specific key	• **`C-h K`** *<keys>* • **`<f1> K`** *<keys>*	(**Info-goto-emacs-key-command-node** KEY)	Open the info node in the Emacs manual which describes the command bound to KEY. • Interactively, if the binding is 'execute-extended-command', a command is read. • The command is found by looking up in Emacs manual's indices or in another manual found via COMMAND's '**info-file**' property or the variable '**Info-file-list-for-emacs**'
Show all key commands for this buffer	• **`C-h b`** • **`<f1> b`**	(**describe-bindings** &optional PREFIX BUFFER)	Display a buffer showing a list of all defined keys, their definitions, in order of precedence. With 🛂 **pel-use-helm-descbinds** you can either bind these keys to **helm-descbinds** to use **helm-descbinds-mode** (bound to **`<f11> ? k B`** to do it.
🗡 **Toggle helm-descbinds mode**	**`<f11> ? k B`**	(**helm-descbinds-mode** &optional ARG)	Toggle **helm-descbings-mode** on/off. When active, the **`C-h b`** and **`<f1> b`** keys invoke **helm-descbinds** by using **helm** with its powerful search and filtering capabilities. Requires 📦 **helm-descbinds** package. 🛂 Set **pel-use-helm-descbinds** user-option to t to install & activate it, via **`<f11> ? k <f2>`** .
Describe active major/ minor(s) modes and the key bindings	• **`C-h m`** • **`<f1> m`** • **`<f11> ? k m`**	(**describe-mode** &optional BUFFER)	Lists the active major mode, all active minor modes and the bound keystrokes. ☝ Use the outline minor mode to collapse all headings and list all active minor modes quickly. • With PEL, use **`<f2> q`** to collapse all headings, **`<f2> a`** to expand all. See 𝕀 **Outline** for info.
Describe bindings for a command	• **`C-h w`** • **`<f1> w`**	(**where-is** DEFINITION &optional INSERT)	Print message listing key sequences that invoke the command DEFINITION. Prompt for command name, supports completion. With prefix key, insert the message in the buffer.
• **Packages, functions symbols, variables describe/help**	The following commands display a description of the item the command requests. The information is displayed in a read-only *Help* buffer. • To search for a function that does something special, one method is to try **`C-h f`** first, then **`C-h d`**. • Example: looking for bolp: **`C-h a beginning of line <RET>`**. If that doesn't bring the info, next try **`C-h d`** with the same input. ☝ Inside a *Help* buffer you can type type **`i`** or **`I`** to open the info node for the current topic, or **`s`** to open the source code and **`c`** to customize. ☝ Emacs >= 28.1: with **completions-detailed** minibuffer user-option non-nil, some commands provide more information with completion		
Describe a package See also: 𝕀 **Packages**	• **`C-h P`** • **`<f1> P`**	(**describe-package** PACKAGE)	Displays full documentation of PACKAGE (symbol). Prompts for package name, supports completion. Shows whether it is installed or not, its version, the features it implements & some extra notes. Accesses the elpa-compliant sites & downloads text file description.
Describe command (Emacs >= 28.1)	• **`C-h x`** • **`<f1> x`**	(**describe-command** COMMAND)	Display the full documentation of COMMAND (a symbol). When called from Lisp, COMMAND may also be a function object.
Describe a function	• **`C-h f`** • **`<f1> f`**	(**describe-function** FUNCTION)	Display the full documentation of FUNCTION (a symbol). • For example: **C-h f *-mode** : Get a completion list of all emacs modes • The buffer shown contains link to the implementation file, even if it is compressed.
Describe symbol ★★	• **`C-h o`** • **`<f1> o`**	(**describe-symbol** SYMBOL &optional BUFFER FRAME)	Display the full documentation of SYMBOL. Will show the info of SYMBOL as a function, variable, and/or face.
Show symbol value	**`<f11> ? S`**	(**pel-show-symbol** SYMBOL)	Prompt for a symbol (defaults with symbol at point) and prints a message showing name and value.
Describe variable	• **`C-h v`** • **`<f1> v`**	(**describe-variable** VARIABLE &optional BUFFER FRAME)	Prompt for Emacs Lisp variable and display information on it. • For example: **C-h v** load-path : shows the emacs lisp path. See: ref: variable current value.
Help on Input Method , as well as encoding & characters			
Help on Input Method See also: 𝕀 **Input Method**	• **`C-h I`** • **`<f1> I`** • **`C-h C-\`**	(**describe-input-method** INPUT-METHOD)	Provide information about the input method. Prompts for the name of an input method. See 𝕀 **Input Method** for more info.
Describe encoding system See also: 𝕀 **File Encoding**	• **`C-h C`** • **`<f1> C`** • **`<f11> ? d C`**	(**describe-coding-system** CODING-SYSTEM)	Display information about CODING-SYSTEM. • Prompts for coding system name. Supports completion. ☝ Type RET to describe current buffer encoding.

Description	Keystroke	Function	Note
Describe language environment See also: Σ **Input Method**	• **C-h L** • **<f1> L**	(**describe-language-environment** LANGUAGE-NAME)	Describe how Emacs supports language environment LANGUAGE-NAME. • Prompts for language name, proposing currently used as default. Supports completion.
Describe syntax-table of current major mode	• **C-h s** • **<f1> s**	(**describe-syntax** &optional BUFFER)	Describe the syntax specifications in the syntax table of BUFFER. The descriptions are inserted in a help buffer, which is then displayed. BUFFER defaults to the current buffer. See also: **Syntax Table @ Emacs Wiki**
Show character syntax info and text properties	**<f11> ? e .**	(pel-syntax-at-point)	Display complete information for character at point in a *Help* buffer to show extended character info **and** display text properties identified by the **pel-syntax-text-properties** user-option in the message area. Access with **<f11> ? <f2>**
Emacs Apropos	The following commands search, gather and open information in buffers using the info reader format. The info reader mode commands are shown after the command list. As with everything in Emacs you can always get help on the current mode, that applies to the info reader mode as well.		
Show information available about specified pattern ★★	**<f11> ? a a**	(**apropos** PATTERN &optional DO-ALL)	Show all meaningful Lisp symbols whose names match PATTERN. • Symbols are shown if they are defined as functions, variables, or faces, or if they have nonempty property lists.
	• PATTERN can be a word, list of words (separated by spaces), or regexp (using some regexp special characters). For a word, search for matches for that word as a substring. For a list of words, search for matches for any two (or more) of those words.		
Get a-propos info on command	• **C-h a** • **<f1> a** • **<f11> ? a c**	(**apropos-command** PATTERN &optional DO-ALL VAR-PREDICATE)	Show commands (interactively callable functions) that match PATTERN. • With **C-u** prefix, or if '**apropos-do-all**' is non-nil, also show non interactive functions. Old Emacs command name was: ***command-apropos***.
	• PATTERN can be a word, a list of words (separated by spaces), or a regexp (using some regexp special characters). If it is a word, search for matches for that word as a substring. If it is a list of words, search for matches for any two (or more) of those words. • Example: **<f1> a mode** : list all modes available in the Emacs session, showing their key bindings and a quick description.		
Look for topic in all info documents ★★	**<f11> ? i a**	(**info-apropos** STRING)	Prompts for a string and looks up for that string in all the indices of **all** the Info documents installed in the system. Opens an Apropos index menu with the links to the found topics. Use this to ***find the manual section(s) that describe a specific function or variable***.
Search for text in function and variables doc strings ★★	• **C-h d** • **<f1> d** • **<f11> ? a d**	(**apropos-documentation** PATTERN &optional DO-ALL)	Search for functions and variables whose documentation strings match the specified pattern and display the appropriate info pages. ⚠ Only searches in the functions predefined at Emacs startup. With **C-u** prefix, or if '**apropos-do-all**' is non-nil, it searches all currently defined documentation strings.
List variables and functions defined in Emacs Lisp file.	**<f11> ? a L**	(**apropos-library** FILE)	List the variables and functions defined by library FILE. • FILE should be one of the libraries currently loaded: should be found in 'load-history'.
Show buffer-local variables	**<f11> ? a l**	(**apropos-local-variable** PATTERN &optional BUFFER)	Show buffer-local variables that match PATTERN. Optional arg BUFFER (default: current buffer) is the buffer to check.
Show user option	**<f11> ? a o**	(**apropos-user-option** PATTERN &optional DO-ALL)	Show user options that match PATTERN. With **C-u** prefix, also show variables. PATTERN can be a word, a list of words (separated by spaces), or a regexp (using some regexp special characters). If it is a word, search for matches for that word as a substring. If it is a list of words, search for matches for any two (or more) of those words.
Show all symbols that have a specific value ★★	**<f11> ? a u**	(**apropos-value** PATTERN &optional DO-ALL)	Show all symbols whose value's printed representation matches PATTERN. • With **C-u** prefix, or if 'apropos-do-all' is non-nil, also looks at function definitions (arguments, documentation and body) and at the names and values of properties.
	PATTERN can be a word, a list of words (separated by spaces), or a regexp (using some regexp special characters). If it is a word, search for matches for that word as a substring. If it is a list of words, search for matches for any two (or more) of those words.		
Show variables that match a specific name pattern	**<f11> ? a v**	(**apropos-variable** PATTERN &optional DO-NOT-ALL)	Show variables that match PATTERN. • With the optional argument DO-NOT-ALL non-nil (or when called interactively with the prefix C-u), show user options only, i.e. behave like 'apropos-user-option'.
• **Key Sequence help**	Emacs has a large number of key bindings as these tables clearly show. Key strokes are extended in various ways and key prefixes is one of them. Following commands show available keys, help learning the key sequences, list the remaining available bindings, and list recent history of typed keys.		
List command history See also: Σ **Undo/Redo/Repeat/Arg**	**<f11> ? d H**	(**list-command-history**)	List history of commands that used the minibuffer. • Show list of commands in the *Command History* buffer as a list of Emacs Lisp forms.
Toggle which-key mode ☝ PEL activates it at startup when **pel-use-which-key** is t	**<f11> ? k K**	(**which-key-mode** &optional ARG)	Toggle which-key-mode: when enabled, as you type a prefix key, all keys bound following this prefix are shown in the mini buffer (if you wait long enough to let them display). Use **which-key** package (part of Emacs >= 30). PEL ***pel-use-which-key*** activates it.
Show top level bindings in the map of the current major mode ☝	**<f11> ? k k**	(**which-key-show-major-mode**)	Show top-level bindings in the map of the current major mode. Use **which-key** package (part of Emacs >= 30). PEL ***pel-use-which-key*** activates it. Also detect evil bindings made using 'evil-define-key' in this map. These bindings will depend on the current evil state.
	Once a list of key/commands is shown type C-h h to list these key-bindings in a help-mode buffer with their commands as links to their help.		
Show state of PEL numlock	**<f11> ? k #**	(pel-show-mac-numlock)	Display state of '***pel-mac-keypad-numlocked***' used to control the numeric keypad.
Show state of key-chord mode. See: Σ **Key-Chords**	• **<f11> <f5> k ?** • **<f11> ? k M-K**	(pel-key-chord-describe)	Show state of key-chord-mode. When key-chord mode is on, list key chord bindings in a help buffer.
Toggle keycast mode on/off	**<f11> ? k c**	(**keycast-mode** &optional ARG)	Show current command and its key binding in the mode line. Use it to create a screen cast to show how to use Emacs. Requires **keycast** available when the **pel-use-keycast** user option is set to **t**.
Show personal key bindings	**<f11> ? k b**	(**describe-personal-keybindings**)	Display all the personal keybindings defined by 'bind-key'.
Display free keys Requires the **free-keys** package PEL activates this when the **pel-use-free-keys** user option is **t**.	**<f11> ? k f**	(**free-keys** &optional PREFIX BUFFER)	Display free keys in current buffer. • A free key is a key without associated key-binding as determined by 'key-binding'.
	Keys on 'free-keys-keys' list with no prefix sequence are listed, possibly together with modifier keys from 'free-keys-modifiers'. To list other, in *Free-keys* buffer: • Type **p** to change the prefix sequence; type the prefix in format recognized by 'kbd', for example type the 3 characters composing "C-x" for that prefix. • Type **b** to change the buffer where the key sequences are applied.		
Display last few typed characters	• **C-h l** • **<f1> l** • **<f11> ? k l**	(**view-lossage**)	Display last few input keystrokes and the commands run. • To record all your input, use 'open-dribble-file'.
Record ALL typed characters to a file	**M-x open-dribble-file**	(**open-dribble-file** FILE)	Start writing all keyboard characters to a dribble file called FILE. If FILE is nil, close any open dribble file. The file will be closed when Emacs exits. ⚠ Be aware that this records **all** characters you type! Don't type passwords at that time!
Redo/edit last complex command executed See also: Σ **Undo/Redo/Repeat/Arg**	• **C-x Esc Esc** • **C-x M-Esc** • **C-x M-:**	(**repeat-complex-command** ARG)	Edit and re-evaluate last complex command, or ARGth from last. • A complex command is one which used the minibuffer. It is placed in the minibuffer as a Lisp form for editing. The result is executed, repeating the command as changed. • If the command has been changed or is not the most recent previous command it is added to the front of the command history. • Use minibuffer history **M-n** and **M-p** to get different commands to edit and resubmit.
Shortdoc (Emacs >= 28.1)	See: **Shortdoc: Emacs's Builtin Elisp Cheat Sheet :** short doc is organized in topic groups, listing functions, their arguments with usage. • You can define new documentation groups using the **define-short-documentation-group** in your code. Currently PEL does not define any. • Inside a Shortdoc buffer, use **n/p** to move to next/previous function, and **N/P** to move point to next/previous section		
Open a shortdoc buffer	**<f11> ? d d**	(**shortdoc** GROUP &optional FUNCTION SAME-WINDOW)	Pop to a buffer with short documentation summary for functions in GROUP. • If FUNCTION is non-nil, place point on the entry for FUNCTION (if any). • If SAME-WINDOW, don't pop to a new window.

Description	Keystroke	Function	Note
Emacs Info Reader • Install needed info packages if they are missing • **USRHOME project** help	Emacs has a powerful **info reader** built-in. • Emacs source repository has info directories that hold a large amount of Emacs related information. • Other software also have info directories with their manuals. • Emacs provide a very powerful environment to search and navigate this information. **Setting Up Emacs for Info:** ⚠️ You may need to **install the info directories for the package of interest** and update the **INFOPATH** environment variable to identify their locations, if it's not already done. • On Linux: • to check if a specific info package is installed, type `info -w PKG`, for example `info -w gdb` to see if the info for gdb is available. • If this prints *manpages* then the info for gdb is *not* installed. • Use your package manager to install the **gdb-doc** package. • For example: `sudo dnf install gdb-doc` or `sudo apt-get install gdb-doc` • To get Emacs-specific info pages, one way to get the files is to build Emacs from source, that create the info directories containing the info files. • On startup Emacs reads the INFOPATH value and sets the **Info-directory-list** variable from it. • My **USRHOME project** provides the **envfor-info** POSIX shell sourced script that builds the INFOPATH from a search of info directories. • Invoke it in a shell with **use-info**, or source it inside your USRHOME **usrcfg**/do-user.sh file to automatically activate it in your shell. • It uses the **find-dir** script to search the info directories in various places. • It also stores the found directories inside the ~/.infopath.txt file that acts as a cache for the information. • The script envfor-info must be sourced. USRHOME provides an alias command for sourcing it: **use-info**. • In a shell where envfor-info has been sourced, the INFOPATH environment variable is set. • Open and independent Emacs process from that shell. You could use the **e** or **ge** commands **PEL provides**. ☝ Inside a *Help* buffer describing a command or function, you can type **`i`** or **`I`** to open the info node for the current topic.			
Open the Info Reader on specific topic	• **`C-h i`** • **`<f1> i`** • **`<f11> ? i i`** • **`⌘-?`**	(**info** &optional FILE-OR-NODE BUFFER)	Open the *info* buffer if already opened. If not, open the info reader for the top node. • A non-numeric prefix argument (**C-u**) directs this command to read a file name from the minibuffer. It is possible to open a compressed .info.gz file directly! Emacs will uncompress it and open it. • A ***numeric prefix*** argument of N selects an Info buffer named "*info*<N>". • Called from a program, or from **M-:**, FILE-OR-NODE may specify an Info node of the form "(FILENAME)NODENAME". • See the **Info Reader Mode Keys** table below for the following actions available once emacs is in the Info Reader Mode.
Open Emacs Manual describing a specified command function	• **`C-h F`** • **`<f1> F`**	(**Info-goto-emacs-command-node** COMMAND)	Go to the Info node in the Emacs manual for command COMMAND. • The command is found by looking up in Emacs manual's indices or in another manual found via COMMAND's 'info-file' property or the variable 'Info-file-list-for-emacs'. COMMAND must be a symbol or string.
Open emacs manual	• **`C-h r`** • **`<f1> r`**	(**info-emacs-manual**)	Display the Emacs manual in Info mode. • It can also be invoked from the menu: Help → Read the Emacs Manual
Open specified info manual	• **`C-h R`** • **`<f1> R`** • **`<f11> ? i m`**	(**info-display-manual** MANUAL)	Prompt for a specific Info manual to open in a buffer. Supports tab completion. • Type return to open a list of all manual. For example: • `<f1> R info` to open the Info manual, • `<f1> R eintr` to open Introduction to Emacs Lisp, • `<f1> R elisp` to open the Emacs Lisp manual, • `<f1> R gdb` to open the gdb manual. ☝ This last one will work only if the info package for gdb is installed and the info directory that holds the gdb info is listed in the INFOPATH variable.
Find specified function function or variable in info	• **`C-h S`** • **`<f1> S`**	(**info-lookup-symbol** SYMBOL &optional MODE)	Display the definition of SYMBOL, as found in the relevant **info** manual. • When this command is called interactively, it reads SYMBOL from the minibuffer. In the minibuffer, use M-n to yank the default argument value into the minibuffer so you can edit it. The default symbol is the one found at point. • With prefix arg MODE a query for the symbol help mode is offered.
Info reader mode Emacs keys See also: • Unix info @ wikipedia • GNU standalone info manual • HTML page/node manual • Emacs Info: An Introduction • **`<f1> R intro <RET>`** • Advanced Info Commands • Info-mode variables	The keys that can be typed in the *Info* buffers and their meanings include the following: **`?`** : Get Info help **`SPC`** : Page down into the node text, move to following text/node if already at end **`<Page Down>`** : Page Down inside the node text (Does not move to other node) **`<Del>`** : Page up into the node text, move to previous text/node if already at top **`<Page Up>`** : Page up into the node text. (Does not move to other node) **`t`** : Move to the top of the Info document **`n`** : Next node in the current level **`o`** : With **ace-link external package** activated when the **pel-use-ace-link:** highlight each target with a target key. **`p`** : Previous node in the current level **`]`** : Next Node (any level) **`[`** : Previous Node (any level) **`u`** : Move to the Upper node (in the menu tree) **`l`** : Info History: visit last (lowercase 'L') **`r`** : Info History: visit history forward **`L`** : Info History: Create Virtual Node of all last visited **`m`** : Menu - Open a node's sub-menu entry. Emacs prompts for the menu text. Abbreviation is supported. Tab completion also supported. **`<RET>`** : Menu - enter nodes' sub-menu (at cursor position) **`1-9`** : Menu - enter nodes' sub-menu (at cursor position) - Type a number between 1 to 9 to select the corresponding menu entry. 1 := first. - Menu asterisk for menu entry 3, 6 and 9 are coloured in red to help identify them. **`f`** *`crossref-text`* : Cross Reference - follow node's cross reference. - To get all cross references, type: **f?** **`<tab>`** : Menu/Cross-Reference - Move cursor to nodes' next sub-menu/cross-reference link **`M-<tab>`** : Menu/Cross-Reference - Move cursor to nodes' previous sub-menu/cross-reference link **`s`** : Search Info - search entire info file for a string. After typing 's' type the string to search and <RET> To repeat search type 's' followed by <RET> **`i`** : Search Info - search index. Search the index for a specific topic. Prompts for the topic. Tab shows list of indices. If several are found, the ',' character can be used to display each one in turn. Access any section of any manual with the ***Info Reader*** by doing this: 1. **C-h i** to open the Info Reader at the top 2. **m** to open the ***menu prompt*** in the menu buffer 3. type topic name and RET **`I`** : (Search Info - construct a virtual info node displaying results of an index search. Runs the command (**Info-virtual-index** TOPIC) **`g`** : Goto a node by name. Topic is a node name: abbreviation is not supported, but **completion with TAB is supported**. Also allows going into another file using the syntax: 'g(*filename*)*Topic*<RET>' Topic may be '*' : means: open the whole file in the buffer. **`<f11> ? i a`** : M-x **info-apropos** : Search Info - search in all Info files installed in the computer **`M-n`** : Create New Independent Info Buffer • opens a new, independent, Info buffer, that at first contains the same Info, but can be managed independently from original. • This can also be done using: • **C-u m** : Move to menu entry into new Info buffer • **C-u g** : Go to topic in new Info buffer • **C-u *number* C-h i** : Open an info topic into a 'Info<#>' buffer (for the identified number) creating it if necessary.		

Description	Keystroke	Function	Note
Helpful - extended help for Emacs with more contextual information	📦 **helpful** external package 🔄 PEL installs and activates it when the **pel-use-helpful** user-option is set. It provides the same Emacs help information provided with more contextual information and extra links. • Use it to debug, trace, look at references, etc…		
Help for function/macro/ special form	`<f1> <f2> a`	(**helpful-callable** SYMBOL)	Show help for function, macro or special form named SYMBOL.
Help for command	`<f1> <f2> c`	(**helpful-command** SYMBOL)	Show help for interactive function named SYMBOL.
Help for function	`<f1> <f2> f`	(**helpful-function** SYMBOL)	Show help for function named SYMBOL.
Help for key	`<f1> <f2> k`	(**helpful-key** KEY-SEQUENCE)	Show help for interactive command bound to KEY-SEQUENCE.
Help for macro	`<f1> <f2> m`	(**helpful-macro** SYMBOL)	Show help for macro named SYMBOL.
Help for symbol	`<f1> <f2> o`	(**helpful-symbol** SYMBOL)	Show help for SYMBOL, a variable, function or macro.
Help for variable	`<f1> <f2> v`	(**helpful-variable** SYMBOL)	Show help for variable named SYMBOL.
Help for symbol at point	`<f1> <f2> .`	(**helpful-at-point**)	Show help for the symbol at point.
Log keys & commands 🚧 interaction-log author is moving to Gnu alpa. MELPA installation works.	PEL provides access to two different packages you can use to show the commands and their key bindings as you type them • These can be used to show what you type during a presentation to other users, or for documentation purpose. The following 2 external packages are supported: 📦 The **command-log-mode** external package. 🔄 PEL activates it when the **pel-use-command-log-mode** user option is turned on (set to **t**). 📦 The **interaction-log** external package. 🔄 PEL activates it when the **pel-use-interaction-log-mode** user option is turned on (set to **t**).		
• **Command Log Mode**	The command-log-mode open a dedicated window that shows the log of all key sequence and mouse events and the executed command name. The information is similar to what is available with view-lossage, but in a nicely formatted way, much easier to use. • See the Σ Windows table for commands that can be used to toggle the dedicated state of the window allowing you to move the window. 📦 This requires the command-log-mode.el file from the **command-log-mode external package**. • 🔄 PEL installs the latest version of that file when the **pel-use-command-log-mode** user option is turned on (set to **t**). • PEL saves it inside your ./emacs/utils directory. To get the latest version, erase that file and its .elc from ./emacs/utils and execute pel-init or restart Emacs. PEL installs it this way because the official project doesn't seem maintained. • With PEL you can customize command-log-mode by typing `<f11> ? <f3>` to access its **command-log** customization group. ☝ The first 2 commands listed below, common-log-mode and global-command-log-mode are available at startup to activate the logging. • Once logging has been activated once the other 3 commands and their bindings are available.		
Toggle command logging for current buffer	`<f11> ? k c c`	(**command-log-mode** &optional ARG)	Toggle command logging: command-log-mode in the current buffer. • The command-log lighter is shown on the mode line while the minor mode is active.
Toggle command logging for all buffers	`<f11> ? k c C`	(**global-command-log-mode** &optional ARG)	Toggle command logging globally: for all buffers. • The command-log lighter is shown on the mode line while the minor mode is active.
Open Command Log buffer	`<f11> ? k c o`	(**clm/open-command-log-buffer** &optional ARG)	Opens (and creates, if non-existant) a buffer used for logging keyboard commands. • With any prefix argument, the existing command log buffer is cleared.
Close Command Log buffer	`<f11> ? k c .`	(**clm/close-command-log-buffer**)	Close the command log window. • Logging continues while the window is closed.
Toggle log of all commands	`<f11> ? k c /`	(**clm/toggle-log-all**)	Toggle the logging of all commands: activate/de-activate common command filtering. • command-log-mode either logs all commands or filter some often used ones like the cursor and character movements. The default setting is controlled by the **clm/log-all**. • The list of non-logged commands is controlled by **clm/non-logged-commands**.
• **Interaction Log Mode**	The **interaction-log** external package is similar to the command-log-mode shown above, but more powerful. It shows the key bindings, the Emacs Lisp command names, the inserted text and other information in different colours. • It supports outputs inside a separate Emacs frame allowing you to continue showing information even after using C-x 1 to maximize the current window. • See **Youtube presentation of interaction-log-mode** by its author: Torstein Krause Johansen. 📦 The **interaction-log** external package. 🔄 PEL activates it when the **pel-use-interaction-log-mode** user option is turned on (set to **t**).		
Start/stop interaction log mode	`<f11> ? k i i`	(**interaction-log-mode** &optional ARG)	Global minor mode logging keys, commands, file loads and messages. • Logged information goes to the *Emacs Log* buffer. • On first invocation the buffer is created but not shown. Select it or use the command **pel-interaction-log-buffer** to show it.
Show interaction log buffer	`<f11> ? k i b`	(**pel-interaction-log-buffer**)	Show interaction log buffer.
Display interaction log in a separate frame.	`<f11> ? k i f`	(**ilog-show-in-new-frame**)	Display log in a pop up frame. Customize 'ilog-new-frame-parameters' to specify parameters of the newly created frame.
Toggle display of buffer names in the interaction log	`<f11> ? k i n`	(**ilog-toggle-display-buffer-names**)	Toggle display of buffers in log buffer for each key event. • This command must be issued inside the interactive log buffer only.
Toggle interaction log view	`<f11> ? k i v`	(**ilog-toggle-view**)	Toggle between different view states: showing only messages, only commands, only file loads, and everything. • This command must be issued inside the interactive log buffer only.
Programming Help	PEL has bindings for the following commands that are useful when editing source code, markup files or any file that has a mode that supports imenu.		
Show what completion mode is currently used.	• `<f11> ? M-c` • `<f11> M-c ?`	(**pel-show-active-completion-mode**)	Display the completion mode currently used, and the Ido prompt geometry when appropriate. • Show key bindings for changing other aspects of input completion.
Show function at point See also: Σ **Inserting Text**	`<f11> ? F`	(**pel-show-function** &optional INSERT-IT)	Display the name of the current "*function*" at point in the mini-buffer. • With any argument, like `C-u`, also insert the *"function"* name at point.

Toggle which-function-mode to display name of current function at point
 	
See also:
• Σ **Menus**
• Σ **Mode Line**
 	
☝ The concept of "*function*" is major mode specific. For example, in C++ mode, if point is inside a class definition it shows the name of the class.
	• `<f11> ? f`
	• `<f11> M-d f`
		(**which-function-mode** &optional ARG)	Toggle mode line display of current function (Which Function mode).
			• With a prefix argument ARG, enable Which Function mode if ARG is positive, and disable it otherwise.
			• The **which-function-mode** is a global minor mode. When enabled, the current function name is continuously displayed in the mode line.
			⚠ Detection of functions and variables depend on the **imenu** functionality. If you modify the content of a buffer, you need to force a menu rescan to get proper results. You can force a rescan with **pel-imenu-rescan**, bound to `<f11> <f10> r`.
			🎛 Identify major modes that automatically active the mode with **which-function-mode** user-option.
			• Use `M-x customize-option which-function-mode` to open the relevant customization buffer.
			• With PEL you can use:
			  • `<f11> ? <f3>` to access the which-func customization group. It will provide access to the customization group even when the feature has not yet been loaded, something that Emacs does not do by default.
			  • `<f11> <f2> o which-function-mode RET` to access the user-option directly.

Show syntax of char at point	`<f11> ? d s`	(**pel-show-char-syntax**)	Display a message showing the character syntax of character at point.
Extra Descriptions	PEL implements a set of extra commands and bindings to built-in Emacs commands to display other the following extra information.		
Show symbols of currently active major mode	`<f11> ? ?`	(**pel-show-major-mode**)	Display the symbol of the currently active major mode.
Show which search tool is currently used	`<f1> ? s`	(**pel-show-active-search-tool**)	Display the currently used search tool.
Show available colours	`<f11> ? d c`	(**list-colors-display** &optional LIST BUFFER-NAME CALLBACK	Display names of defined colors, and show what they look like.

Description	Keystroke	Function	Note
Show encoding of file visited in current buffer • See also: Σ **Help/Info**	`<f11> ? d e`	(**pel-show-buffer-file-encoding**)	Show coding system of file in current buffer. • Open a *Help* buffer and show the value of the buffer-file-coding-system variable.
List all available faces	`<f11> ? d F`	(**list-faces-display** &optional REGEXP)	List all faces, using the same sample text in each.
Show buffer and file name	`<f11> ? d f`	(**pel-show-window-filename-or-buffer-name**)	Show the (full path) name of the file or buffer of current window.
Show information about an input method	`<f11> ? d i`	(**list-input-methods**)	Display information about all input methods.
Display content of kill ring	`<f11> ? d k`	(**pel-show-kill-ring**)	Display content of 'kill-ring' in *Help* buffer.
Print current buffer line # (and narrowed line #)	`<f11> ? d l`	(**what-line**)	Print the current buffer line number and narrowed line number of point.
Query info about point **Show information about current character.**	• `C-x =` • `<f11> ? d p`	(**what-cursor-position** &optional DETAIL)	Displays information about character at point in the echo area: position, character, encoding.
	• ☝ With any prefix argument opens a *Help* buffer and show the complete information of character at point with all properties, face, encoding, etc. • Type: `C-u C-x =` With PEL, you can also type: `C-- C-x =`		
	`<f11> ? d P`	(**pel-what-cursor-position**)	Same as above but always display the complete information.
Show window info See Σ **Windows Hydra**	• `<f11> ? D w` • `<f11> w d ?` ✱ `<f7> I`	(**pel-show-window-info**)	Show information about window in minibuffer: #, buffer, size, dedicated, etc…
Display ASCII table See also: Σ **Input Method**	`<f11> ? A`	(**ascii-table**)	Show an interactive ASCII table in the other (next) window.
	📦 Requires the **ascii-table** package. PEL activates this when the **pel-use-ascii-table** user option is **t**.		
About Emacs	Information about Emacs, its environment and configuration is available through a set of commands listed below		
Display Emacs version	`<f11> ? e v`	(**emacs-version**)	Display Emacs version
Display Emacs uptime	`<f11> ? e u`	(**emacs-uptime** &optional FORMAT)	Display a string giving the uptime of this instance of Emacs in the echo area.
Display Emacs Config features	`<f11> ? e C`	(**pel-emacs-config-features**)	Print the names of all Emacs configured compilation features. It also prints whether Emacs was compiled with or without native compilation.
Open local copy of Emacs PDF reference card	`<f11> ? e r`	(**pel-open-emacs-refcard**)	Prompt for an Emacs REFCARD and open it. Supports tab completion
	• Attempts to find the directory where the Emacs PDF reference card files are stored. Failing to detect them, it uses the directory identified by the **pel-emacs-refcard-dirpath** user option. Access custom group with **<f11> ? <f2>**		
Open info about Emacs bug	`<f11> ? e B`	(**pel-emacs-bug-info** &optional in-browser)	Prompt for Emacs bug number. Open the bug discussion email stream in a Gnus buffer. • With optional arg, open in system buffer instead.
Show buffer stats & current window last visited buffer	• `<f11> b ?` • `<f11> ? e b`	(**pel-emacs-buffer-stats**)	Show buffer statistics: total count of buffers, # of buffer visiting files, # of special buffers and # of internal buffers. Also show the name of previous buffer used in the current window.
Show number of available and key bound commands	`<f11> ? e c`	(**pel-emacs-command-stats**)	Display number of available commands and the number of those that have key bindings in the echo area, and the number of bindings in the global map.
Show loaded files & features	`<f11> ? e l`	(**pel-emacs-load-stats** &optional WITH_DETAILS)	Display the number of loaded files and the number of features currently loaded. • With `C-u` prefix print features in a buffer. With `C-u C-u`, also print load information, with symbols displayed as clickable buttons that open a help buffer describing it.
Display Memory Usage	`<f11> ? e m`	(**pel-emacs-mem-stats**)	Display a short Emacs memory statistics inside an *emacs-mem-stats* buffer.
Display Memory Report (Emacs >= 28.1)	`<f11> ? e M`	(memory-report)	Generate a report of how Emacs is using memory. • This report is approximate, and will commonly over-count memory usage by variables, because shared data structures will usually by counted more than once.
Check/display list of shadowed Emacs Lisp files	`<f11> ? e s`	(**list-load-path-shadows** &optional STRINGP)	Display a list of Emacs Lisp files that shadow other files • Shows any shadows in a '*Shadows*' buffer
Print imenu controlling variables See also: Σ **Menus**	`<f11> ? e i`	(**pel-imenu-print-vars**)	Print the value of the imenu variables used to control the imenu functionality for the current buffer. Symbols are clickable buttons to help on the symbol. • Print this information in a *imenu-dbg* buffer. • Use to investigate the imenu support for a major mode.
Print value of outline controlling variables See also: Σ **Outline**	`<f11> ? e o`	(**pel-outline-print-vars**)	Print the current buffer specific values of outline controlling variables. Use this to learn possible how to control the outline minor mode.
See Emacs executable path	`<f11> ? e x`	(**pel-emacs-executable**)	Display Emacs executable path in echo area.
Display load-path	`<f11> ? e p`	(**pel-emacs-load-path** &optional N)	Show the current load-path inside a new *load-path* buffer. Open the buffer in the current window or the one identified by N, with the display-line-number-mode on.
	• If a buffer with the name *load-path* already exists, creates a new buffer name that contains the string *load-path*. • **Window selection**: If N is not specified, nil or 1: open buffer in current window. • If N is negative, create a new window and open buffer inside it. • If N is 0: : open buffer in other window • If N in [2,8] range, open buffer in window identified by the direction corresponding to the cursor in a numeric keypad: 8 := 'up 4 := 'left 5 := 'current 6 := 'right 2 := 'down • If N is 9 or larger: search in window below.		
Display Emacs initialization time with benchmark information if available • 📦 Uses the **benchmark-init** library to measure time of the various loaded modules. • See installation notes ➡	• `<f11> ? e t` • `M-S-<f9>`	(**pel-show-init-time**)	Display benchmark startup time. Display the benchmark initialization and duration tree in 2 buffers if the benchmark-init library is installed and loaded in the init.el file. It also display the Emacs startup time inside the echo area.

	Use `M-x list-package`, select benchmark-init and install it. Then update your **init.el** file, place the following lines as close as possible to the top of the file:
	`;; Setup Benchmark Measurement`
	`;; --------------------------`
	`;; Load benchmark soon to measure as much as possible.`
	`;; CAUTION: Modify the path when a new version is available.`
	`(require 'benchmark-init`
	`(expand-file-name`
	`"~/.emacs.d/elpa/benchmark-init-20150905.938/benchmark-init"))`
	`(add-hook 'after-init-hook 'benchmark-init/deactivate)`
	• Update the path in this code if necessary. When using PEL, this is already in the commented-out section OPTION C of the init.el example.		

List processes See also: Σ **Shells**	• `<f11> ? e C-p` • `<f11> z ?`	(**list-processes** &optional QUERY-ONLY BUFFER)	Display a list of all processes that are Emacs sub-processes in the *Process List* buffer. With non-nil optional argument, only processes with the query-on-exit flag set are listed. Any process listed as exited or signalled is actually eliminated after the listing is made.
Print process tree	`<f11> ? e M-p`	(**pel-process-tree**)	Print the process tree of the inferior process of the current buffer if any, otherwise print the process tree of Emacs itself. ⚠ Requires the **pstree** command. It generates an error if it is not available.
Print value of Emacs lisp.el control variables	`<f11> ? e a l`	(**pel-show-lisp-control-variables** & optional APPEND)	Print the values of all user-options and variables used by Emacs lisp.el file; that file controls the behaviour of important navigation and marking functions. Use this command to print their values used for a major-mode. With prefix argument append new information to existing buffer.

Description	Keystroke	Function	Note
ESUP - Emacs Start Up Profiler	`<f1> ? e P`	(**esup** &optional INIT-FILE &rest ARGS)	Profile the startup time of Emacs in the background. • If INIT-FILE is non-nil, profile that instead of USER-INIT-FILE. • ARGS is a list of extra command line arguments to pass to Emacs.
	📦 Requires the **esup** external package. PEL activates it when the **pel-use-esup** customization variable is set to **t**.		
	⚠️ The esup profiler has several limitations: 1) it only supports Emacs running in graphics mode. 2) esup steps into `require' and `load' forms at the top level of a file but not if they are enclosed in any other statements. This limits its usefulness when conditional loading is located in the init.el file and when the use-package macros are used. Both of these techniques are used by PEL to reduce init time.		
Using Man inside Emacs See also: • **𝛑 - Erlang** • **Σ Customize**	Emacs provide 2 main commands to display **man pages** inside buffers. • Both of these are much more powerful than the usual man reader available on the shell allowing navigation across man pages & opening hyperlinks. • The man command uses the system man utility, while woman is a complete implementation which has some formatting limitations compared to man but it's very useful in systems where man is not available. • The man command will find pages that the system's man can find. This can be extended or modified by setting the MANPATH environment variable. Inside Emacs you can also customize the Emacs **Man-switches** user option to provide extra configuration including a different MANPATH by using the -M switch. For an example see how to add Erlang man pages in the 𝛑 - Erlang table.		
Open a man page inside an Emacs buffer • On Unix/Linux, use it to display help about C/C++ functions, types.	• `<f11> ? m` • `M-<f8>` • `⌘-M`	(**man** MAN-ARGS)	Open a Man page inside an Emacs window.
	Using man pages inside emacs is even better than using it from the shell because: • The links are active and can be followed. When the man page describes a directory or file, emacs will open the file or the directory (in direct mode) when pressing **<RET>** over the link. • You can navigate easily between sections (n/p will move to the next/previous section). You can use any of the searches. • You can use any of the options to the man command at the prompt, like the -a option to access all man pages of the same name. Then use **M-n** and **M-p** to move from one to the other page, inside the same buffer. • See all keys available in mode, with **<f1> m** or `<f11> ? k m`. ☝ The man command prompts, using the word at point as the default. 🎛 PEL key sequence to customize man: `<f11> <f2> E m` ☝ The man command provides completion at prompt. However, if you set up a MANPATH to isolate on directory to get only the list of commands in a specified set of man pages (eg. for Erlang commands only), the completion will only work if the man directory contains a whatsis database file. See my description on how to create whatis file for local man directory.		
Use Emacs as a man viewer from the shell	☝ You may want to use *Emacs as your man pager directly from the shell*. I have written shell code to do this: launch Emacs to open the requested man page when you type man from the shell. See my **USRHOME** project: **use-emacs-for-man**.		
Open man page for item at point	`M-S-<f8>`	(**pel-man-at-point**)	Open a man page for the topic at point if any, otherwise prompts for topic. • Man page section controlled by user option named pel-%s-man-section, where '%s' is replaced by the major mode. ☝ Useful for modes like Tcl where section name differs.
Open a man page without external man process: woman	• `<f11> ? w` • `C-<f8>`	(**woman** &optional TOPIC RE-CACHE)	Open a man page file in Emacs using the woman mode, completely implemented in Emacs Lisp (and therefore without using the external 'man' process).
	That can be very useful under environments where man is not available (such as basic Microsoft Windows ®). 🎛 PEL key sequence to customize woman: `<f11> <f2> E w` ☝ With **ace-link external package** activated when the **pel-use-ace-link** user option is set to **t**., the following key is activated: o : Quick navigation: highlight each target with a target key.		
Emacs Bug Reports See also: • **EmacsBugTracker @ Emacs Wiki** • **Emacs Bug triaging article**	• Emacs bugs are managed by the **GNU Bug Tracker** which is an instance of **Debian bug tracker: debbugs.** • The **GNU Bug Tracker** is used as a bug tracker for several GNU project. See the list of **Gnu software packages using this bug tracker.** • More info is available in the **GNU Bug Tracker Documentation**. This information can also be accessed directly within Emacs by using the 📦 **debbugs** external package. PEL activates it when the **pel-use-debbugs** user option is turned on (set to **t**). PEL also binds the **debbugs** commands to the following keys. With PEL access the **debbugs** customization group via the `<f11> ? <f3>` key sequence.		
List all outstanding Emacs bugs	`<f11> ? b a`	(**debbugs-gnu** SEVERITIES &optional PACKAGES ARCHIVEDP SUPPRESS TAGS)	List all outstanding bugs.
Search for Emacs bugs	`<f11> ? b s`	(**debbugs-gnu-search** PHRASE &optional QUERY SEVERITIES PACKAGES ARCHIVEDP)	Search for Emacs bugs interactively. • Search arguments are requested interactively. The "search phrase" is used for full text search in the bugs database. • Further key-value pairs are requested until an empty key is returned. If a key cannot be queried by a SOAP request, it is marked as "client-side filter". • When using interactively, use C-x M-: after this command for reusing the argument list. • Be careful in editing the arguments, because the allowed attributes for QUERY depend on PHRASE being a string, or nil. • See Info node '(debbugs-ug) Searching Bugs'.
List all users tags	`<f11> ? b u`	(**debbugs-gnu-usertags** &rest USERS)	List all user tags for USERS, which is ("emacs") by default.
List bug reports that contain a patch	`<f11> ? b p`	(**debbugs-gnu-patches**)	List the bug reports that have been marked as containing a patch.
List all bugs or specified bugs	`<f11> ? b b`	(**debbugs-gnu-bugs** &rest BUGS)	List all BUGS, a list of bug numbers. • In interactive calls, prompt for a comma separated list of bugs or bug ranges, with default to 'debbugs-gnu-default-bug-number-list'. • This accepts a single bug number, a comma separated list of bug numbers as well as dash separated range of bug numbers.
List bugs tags locally	`<f11> ? b t`	(**debbugs-gnu-tagged**)	List the bug reports that have been tagged locally.
List all outstanding Emacs bugs in Org-mode format	`<f11> ? b A`	(**debbugs-org**)	List all outstanding bugs using an Org-mode format.
Search for Emacs bugs, list bugs in Org-mode format	`<f11> ? b S`	(**debbugs-org-search**)	Search for bugs interactively. List bugs in Org-mode format. • Search arguments are requested interactively. The "search phrase" is used for full text search in the bugs database. • Further key-value pairs are requested until an empty key is returned. If a key cannot be queried by a SOAP request, it is marked as "client-side filter".
List bug reports that contain a patch, list bugs in Org-mode format	`<f11> ? b P`	(**debbugs-org-patches**)	List the bug reports that have been marked as containing a patch. List bugs in Org-mode format.
List all bugs or specified bugs in Org-mode format	`<f11> ? b B`	(**debbugs-org-bugs**)	List all bugs, a list of bug numbers. List bugs in Org-mode format. • In interactive calls, prompt for a comma separated list of bugs or bug ranges, with default to 'debbugs-gnu-default-bug-number-list'.
List bugs tags locally in Org-mode format	`<f11> ? b T`	(**debbugs-org-tagged**)	List the bug reports that have been tagged locally. List bugs in Org-mode format.
More Help			
Open Emacs Tutorial	• `C-h t` • `<f1> t`	(**help-with-tutorial** &optional ARG DONT-ASK-FOR-REVERT)	Open an Emacs Tutorial. Restore location if used before (after prompt).
Find Elisp Package See also: **Σ Packages**	• `C-h p` • `<f1> p`	(**finder-by-keyword**)	Find packages matching a given keyword. Useful to search for packages supporting a specific concept.
Open Emacs FAQ	• `C-h C-f` • `<f1> C-f`	(**view-emacs-FAQ**)	Display the Emacs Frequently Asked Questions (FAQ) file.

Description	Keystroke	Function	Note
Emacs news	• **C-h n** • **<f1> n**	(**view-emacs-news** &optional VERSION)	Display info on recent changes to Emacs. With argument, display info only for the selected version. Includes code modifications of each version of Emacs.
Display local help in echo area	**<f1> .** **C-h .** **C-c ! H**	(**display-local-help** &optional ARG)	Display local help in the echo area. • This displays a short help message, namely the string produced by the 'kbd-help' property at point. If 'kbd-help' does not produce a string, but the 'help-echo' property does, then that string is printed instead. • A numeric argument ARG prevents display of a message in case there is no help. While ARG can be used interactively, it is mainly meant for use from Lisp.
Emacs + PEL specifics	The following commands provide more information about Emacs and how PEL uses it.		
Show PEL user option and package info See also: **Σ Customize**	**<f11> ? e ?**	(**pel-package-info** &optional FULL-REPORT ON-STDOUT	Display the following information inside a *pel-user-options* buffer: • name of custom file, package-user-dir, the number of PEL user-options, and the number of them that are active, number of loaded files, and features. • The number of Elpa packages active: the count of the ones directly installed because of active PEL user-options and the count of them installed as dependencies of the first group. • The number of Emacs Lisp files stored in the ~/.emacs.d/utils (or equivalent directory) as a result of PEL user options. • The number of elpa-compliant packages that have a newer version and could be updated. • With optional argument, like **C-u**, generates a full report with more details.
Display name of customization file. Show whether PEL dual independent customization is used or not. See also: **Σ Customize**	• **<f11> ? e <f2>** • **<f11> <f2> ?**	(**pel-setup-info-dual-environment**)	Display current PEL customization setup. • Check two independent customization files for terminal/tty and graphics mode are requested and if so check if they are setup properly. • Report an error and list problems if there are any, otherwise display the current setup. ⚠ After executing that command you will have to edit your init.el file and set the pel-use-graphic-specific-custom-file-p symbol to **t**.
Display current Emacs Startup configuration setup See also: **Σ Fast Startup**	• **<f11> ? e M-S** • **<f11> M-S ?**	(**pel-setup-info**)	Display current state of PEL setup: whether Emacs startup is used in normal or in fast startup operation mode.
Open PEL PDF Help File See also: **≻Legend**	PEL includes a large set of help PDF (like this one) that are hosted on GitHub and located in your local PEL installation. The **pel-help-pdf** command supports prefix commands that control how to open the file and , for some context, open a main topic or secondary topic file. User-options also control the behaviour. This is described at the top of this PDF.		
Open this PDF file.	**<f11> ? <f1>**	(**pel-help-pdf** &optional N)	Open the **Σ Help/Info** local PDF.
Select and Open a PEL PDF file	• **<f11> ? p** • **<f11> p**	(**pel-help-pdf-select** &optional OPEN-WEB-PAGE)	Prompt for a PEL PDF and open it. • Supports tab completion.
Open a Dired Buffer for PEL PDF files.	**<f11> ? P**	(**pel-help-pdfs-dir**)	Open a Dired buffer on the PEL PDF directory. Inside Dired you can open a PDF file by typing 'z' over the file name. You can also select several and type 'z' to open them all.
≻Index	**<f11> <f1>**	Open **≻Index** PDF file, a quick index with links to all other PEL PDF files.	
Σ Abbreviations	**<f11> a <f1>**	Open **Σ Abbreviations** PDF file.	
Σ Align	**<f11> t a <f1>**	Open **Σ Align** PDF file.	
Σ Auto-Completion	**<f11> , <f1>**	Open **Σ Auto-Completion** PDF file.	
Σ Bookmarks	**<f11> ' <f1>**	Open **Σ Bookmarks** PDF file.	
Σ Buffers	**<f11> b <f1>**	Open **Σ Buffers** PDF file.	
Σ Case Conversions	**<f11> t <f1> 1**	Open **Σ Case Conversions** PDF file.	
Σ Comments	**<f11> ; <f1>**	Open **Σ Comments** PDF file.	
Σ Cut & Paste	• **<f11> = <f1>** • **<f11> - <f1>**	Open **Σ Cut & Paste** PDF file.	
Σ Counting	**<f11> c <f1>**	Open **Σ Counting** PDF file.	
Σ Cursor	**<f11> m <f1>**	Open **Σ Cursor** PDF file.	
Σ Customize	**<f11> <f2> <f1>**	Open **Σ Customize** PDF file.	
Σ Diff & Merge	**<f11> d <f1>**	Open **Σ Diff & Merge** PDF file.	
Σ Dired	**<f11> f <f1> 2**	Open **Σ Dired** PDF file.	
Σ Drawing	**<f11> D <f1>**	Open **Σ Drawing** PDF file.	
Σ Enriched Text	**<f11> t e <f1>**	Open **Σ Enriched Text** PDF file.	
Σ Fast Startup	**<f11> <f2> S <f1>**	Open the **Σ Fast Startup** PDF file.	
Σ File-mngt	**<f11> f <f1> 1**	Open **Σ File-mngt** PDF file.	
Σ File/Directory Variables	**<f11> f v <f1>**	Open **Σ File/Directory Variables** PDF file.	
Σ Filling/Justification	• **<f11> t f <f1>** • **<f11> t j <f1>**	Open **Σ Filling/Justification** PDF file.	
Σ Frames	**<f11> F <f1>**	Open **Σ Frames** PDF file.	
Σ Grep	**<f11> g <f1>**	Open **Σ Grep** PDF file.	
Σ Help/Info	**<f11> ? <f1>**	Open **Σ Help/Info** PDF file.	
Σ Hide/Show	**<f11> M-/ <f1>**	Open **Σ Hide/Show** PDF file.	
Σ Highlight	**<f11> h <f1>**	Open **Σ Highlight** PDF file.	
Σ Indentation	**<f11> TAB <f1>**	Open **Σ Indentation** PDF file.	
Σ Input Method	**<f11> t <f1> 2**	Open **Σ Input Method** PDF file.	
Σ Inserting Text	• **<f11> i <f1>** • **<f11> y <f1>** • **<f11> _ <f1>**	Open **Σ Inserting Text** PDF file.	
Σ Keyboard Macros	**<f11> k <f1>**	Open **Σ Keyboard Macros** PDF file.	
Σ Key-Chords	**<f11> <f5> k <f1>**	Open the **Σ Key-Chords** PDF file.	
Line management. **Σ Display - Lines**	**<f11> l <f1>**	Open **Σ Display - Lines** PDF file.	
Σ Marking	**<f11> . <f1>**	Open **Σ Marking** PDF file.	
Σ Mode Line	**<f11> M-d <f1>**	Open **Σ Mode Line** PDF file.	
Σ Menus	**<f11> <f10> <f1>**	Open **Σ Menus** PDF file.	
Σ Outline	**<f11> M-l <f1>**	Open **Σ Outline** PDF file.	
Σ Projectile	• **<f11> <f8> <f1>** • **<f8> <f1>**	Open **Σ Projectile** PDF file. • The key sequence **<f8> <f1>** is available when the projectile mode is activated.	

Description	Keystroke	Function	Note
Σ Registers	<f11> r <f1>	Open Σ Registers PDF file.	
Σ Scrolling	<f11> | <f1>	Open Σ Scrolling PDF file.	
Σ Search/Replace	<f11> s <f1>	Open Σ Search/Replace PDF file.	
Σ Sessions	<f11> S <f1>	Open Σ Sessions PDF file.	
Σ Shells	<f11> z <f1>	Open Σ Shells PDF file. Information about how to launch shell, process and applications.	
Σ Sorting	<f11> o <f1>	Open Σ Sorting PDF file (o for ordering).	
Σ Speedbar	<f11> M-s <f1>	Open Σ Speedbar PDF file.	
Σ Spell Checking	<f11> $ <f1>	Open Σ Spell Checking PDF file.	
Σ Text Modes	• <f11> t <f1> 3 • <f11> t m <f1>	Open Σ Text Modes PDF file.	
Σ Time Tracking	<f11> T <f1>	Open Σ Time Tracking PDF file.	
Σ Transpose	<f11> t t <f1>	Open Σ Transpose PDF file.	
Σ Whitespace	<f11> t w <f1>	Open Σ Whitespace PDF file.	
Σ Undo/Redo/Repeat/Arg	<f11> u <f1>	Open Σ Undo/Redo/Repeat/Arg PDF file.	
Σ VCS-Mercurial	<f11> v <f1>	Open Σ VCS-Mercurial PDF file.	
Σ Web	<f11> f <f1> 3	Open Σ Web PDF file.	
Σ Windows	<f11> w <f1>	Open Σ Windows PDF file.	
Σ Xref	<f11> X <f1>	Open Σ Xref PDF file.	
Specialized Minor Modes	Extending the capabilities for specific programming languages		
𝔓𝔏- Lispy	PEL does not provide a global key binding for Lispy. This is available for the Lisp family languages as well as Julia and Python.		
Mode Specific PDF Help: • **Programming Languages**	PEL PDF files for specific major modes can be opened using the <f12> <f1> key from a buffer in that mode. • The longer key sequence that starts with <f11> SPC is available globally, allowing you to open it from any buffer. All commands support prefix arguments that allows control to open the local PDF with the PDF viewer or open the GitHub hosted raw PDF in your browser. That is more flexible allowing you to browse quickly through all PDF files. See description of arguments at the beginning of the section.		
𝔓𝔏- AppleScript	<f11> SPC a <f1>	Open 𝔓𝔏- AppleScript PDF	
	<f12> <f1>		
𝔓𝔏 - C	<f11> SPC c <f1>	Open 𝔓𝔏 - C PDF	
	<f12> <f1>		
𝔓𝔏 - C++	<f11> SPC C <f1>	Open 𝔓𝔏 - C++ PDF	
	<f12> <f1>		
𝔓𝔏 - Clojure	<f11> SPC C-j <f1>	Open 𝔓𝔏 - Clojure PDF	
	<f12> <f1>		
𝔓𝔏 - D	<f11> SPC D <f1>	Open 𝔓𝔏 - D PDF	
	<f12> <f1>		
𝔓𝔏 - Erlang	<f11> SPC e <f1>	Open 𝔓𝔏 - Erlang PDF	
	<f12> <f1>		
𝔓𝔏 - Elixir	<f11> SPC x <f1>	Open 𝔓𝔏 - Elixir PDF	
	<f12> <f1>		
𝔓𝔏 - Forth	<f11> SPC f <f1>	Open 𝔓𝔏 - Forth PDF	
	<f12> <f1>		
𝔓𝔏 - Go	<f11> SPC g <f1>	Open 𝔓𝔏 - Go PDF	
	<f12> <f1>		
𝔓𝔏 - Hy	<f11> SPC C-h <f1>	Open 𝔓𝔏 - Hy PDF	
	<f12> <f1>		
𝔓𝔏 - Gleam	<f11> SPC M-G <f1>	Open the 𝔓𝔏 - Gleam PDF	
	<f12> <f1>		
𝔓𝔏 - Javascript	<f11> SPC i <f1>	Open 𝔓𝔏 - Hy PDF	
	<f12> <f1>		
𝔓𝔏 - Julia	<f11> SPC j <f1>	Open 𝔓𝔏 - Julia PDF	
	<f12> <f1>		
𝔓𝔏 - Janet	<f11> SPC T <f1>	Open the 𝔓𝔏 - Janet PDF	
	<f12> <f1>		
𝔣𝔓𝔏 - Emacs Lisp	<f11> SPC l <f1>	Open 𝔣𝔓𝔏 - Emacs Lisp PDF	
	<f12> <f1>		
𝔓𝔏 - Common Lisp	<f11> SPC L <f1>	Open 𝔓𝔏 - Common Lisp PDF	
	<f12> <f1>		
𝔓𝔏 - LFE	<f11> SPC C-l <f1>	Open 𝔓𝔏 - LFE PDF	
	<f12> <f1>		
𝔓𝔏 - NetRexx	<f11> SPC N <f1>	Open 𝔓𝔏 - NetRexx PDF	
	<f12> <f1>		
𝔓𝔏 - Python	<f11> SPC p <f1>	Open 𝔓𝔏 - Python PDF	
	<f12> <f1>		
𝔓𝔏 - REXX	<f11> SPC R <f1>	Open 𝔓𝔏 - REXX PDF	
	<f12> <f1>		

Description	Keystroke	Function	Note
𝔓𝔏 - Rust	<f11> SPC r <f1>	Open 𝔓𝔏 - Rust PDF	
	<f12> <f1>		
𝔓𝔏 - Scheme	<f11> SPC C-s <f1>	Open 𝔓𝔏 - Scheme PDF	
	<f12> <f1>		
𝔓𝔏 - UNIX Shell	<f11> SPC Z <f1>	Open 𝔓𝔏 - UNIX Shell PDF. Describes the major mode used for editing Unix shell scripts.	
	<f12> <f1>		
𝔓𝔏 - V 🚧	<f11> SPC v <f1>	Open 𝔓𝔏 - V PDF	
	<f12> <f1>		
• Build Tools			
𝔓𝔏 - Make	<f11> SPC M <f1>	Open 𝔓𝔏 - Make	
	<f12> <f1>		
• Markup languages			
Ⓜ Graphviz Dot	<f11> SPC M-g <f1>	Open Ⓜ Graphviz Dot PDF	
	<f12> <f1>		
Ⓜ Outline/Org-Mode	<f11> SPC M-o <f1>	Open Ⓜ Outline/Org-Mode PDF	
	<f12> <f1>		
Ⓜ PlantUML	• <f11> D u <f1> • <f11> SPC M-u <f1>	Open Ⓜ PlantUML PDF	
	<f12> <f1>		
Ⓜ Markdown	<f11> SPC M-m <f1>	Open Ⓜ Markdown PDF	
	<f12> <f1>		
Ⓜ reStructuredText	<f11> SPC M-r <f1>	Open Ⓜ reStructuredText PDF	
	<f12> <f1>		
• Using Shells			
Σ shell-mode	<f12> <f1>	Open the Σ shell-mode PDF which describes the commands available in shell-mode.	
Σ term-mode	<f12> <f1>	Open the Σ term-mode PDF which describes the commands available in term-mode.	
Σ eat-mode	<f12> <f1>	Open the Σ eat-mode PDF which describes the commands available in eat-mode.	
Σ vterm-mode	<f12> <f1>	Open the Σ vterm-mode PDF which describes the commands available in vterm-mode.	

Help — References

Topic & Link	Description
Emacs Help	
GNU Emacs Manuals Online	The page with the list of all available online GNU Emacs manuals.
GNU Emacs Manual - Help	Emacs manual - Help chapter
Gnu Emacs Manual - Help Mode	Describes the command and key bindings that can be used in the Help-mode buffer window, which shows the help information.
Emacs Manuals	Note that **all** Emacs manuals are available ***inside*** *of Emacs. It's better to test, investigate code, etc..*
GNU Emacs Manuals Online	Lists all GNU Emacs manuals, reference cards, etc…
GNU Emacs Manual	Points to different formats of the manual. The format where all is inside one HTML file is useful to search. There's also the PDF formats.
GNU Reference Cards	This is accessible via the first link.
Emacs Papers	
EMACS: The Extensible, Customizable Display Editor	This paper was written by Richard Stallman in 1981 and delivered in the ACM Conference on Text Processing.
Emacs Tutorials	
A Guided Tour of Emacs	The official Emacs Tutorial. Part of Emacs. Best used ***inside*** Emacs. A good starting point. Use the others to get different point of views.
Absolute Beginner's Guide to Emacs	
A Tutorial Introduction to GNU Emacs	
Practical Emacs Tutorial @ ErgoEmacs	
Emacs Cheat Sheet / Keystroke Lists	Note, however, that Emacs itself and PEL provides similar information.
Emacs Videos	
Emacs Rocks - home	A collection of Youtube homed videos about various Emacs features. Well documented with keystrokes showing on the screen cast. Worth watching slowly to catch what is being done.
Emacs and Man files	
How to create a local whatis file	Show how to create a missing whatis file for a set of man pages and the philosophy behind apropos, whatis and makewhatis.