↖ Topic Index

Getting Help / Apropos / Descriptions / Info Manuals / Queries

Description	Keystroke	Function	Note
Getting Help	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 supplements Emacs help with a **large set of topic-specific PDF files** such as this one (identified as ∑ **Help/Info**). • See the ➢**Index** PDF with has links to all PEL PDFs. • These PDFs have a large set of hyper links to each other, to Emacs manual and external package home and description sites. • Type the **<f12> <f1>** key sequence to open help for several major modes or **<f1> ? p** for help on a specific topic. • By default PEL opens the local PDF file using the local PDF rendering application. You can force PEL to launch your default browser and open the Github raw PDF instead by changing the values of **pel-flip-help-pdf-arg** or and/or **pel-open-pdf-method** user-options. • Type **<f1> ? <f2>** to open their customization group and change those..		
Open this PDF file.	**<f1> ? <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the ∑ **Help/Info** local PDF. If the prefix argument (like **C-u** or **M--**) is used, then it opens the remote GitHub hosted raw PDF instead. If the **pel-flip-help-pdf-arg** user-option is set it's the other way around.
∑ **Customize PEL Help Support**	**<f1> ? <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL help support. • If OTHER-WINDOW is non-nil (use **C-u**), display in other window.
∑ **Customize Emacs Help Support**	**<f1> ? <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs grep support. Groups: command-log, debbugs, helpful, which-func, which-key.
Emacs Reference Cards	Emacs has a set of short **PDF reference cards**. Access customization group with **<f1> ? <f2>** • PEL provides a command to open the local copy of these files if they are present. • 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	**<f1> ? 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.		
• **Prefix Keys**	Key sequences consist of either one keystroke like **C-a** or **M-b**, or a key sequence that starts with a prefix, like **C-x s**, where **C-x** is the key prefix.		
List all keys that belong to a prefix	• *<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**
• **Describe Help**	The following commands display a description of the item the command requests. The information is displayed in a read-only *Help* buffer.		
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, and their definitions. The keys are displayed in order of precedence.
Help on key binding	• **C-h k** *<keystroke>* • **<f1> k** *<keystroke>*	(**describe-key** &optional KEY UNTRANSLATED UP-EVENT)	Display documentation of the function invoked by KEY. • KEY can be any kind of a key sequence; it can include keyboard events, mouse events, and/or menu events. • Get binding for the typed <keystroke> in the current context. • Displays the name of the command function, it's description, it's bindings.
	☝ The PEL system comes with an extensive key binding system entered around a set of function keys like **<f11>** , some of these are bindings for commands that already have standard Emacs bindings and sometimes the standard Emacs bindings are easier to type. Using **C-h k** (or the equivalent **<f1> k**) binding to get help on a specific binding may help you discover other, more efficient key bindings for the same command.		
Open Info manual describing the command for the specific key	• **C-h K** *<keystroke>* • **<f1> K** *<keystroke>*	(**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'
Print name of function invoked by key	• **C-h c** *<keystroke>* • **<f1> c** *<keystroke>*	(**describe-key-briefly** &optional KEY INSERT UNTRANSLATED)	Print the name of the function KEY invokes. KEY is a string.
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.
Describe a package See also: ∑ **Packages**	• **C-h P** • **<f1> P**	(**describe-package** PACKAGE)	Display the full documentation of PACKAGE (a symbol). • Prompts for the package name, supports completion. • Shows whether it is installed or not, its version, the features it implements and some extra notes. Accesses the the elpa-compliant sites and downloads text file description.
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 file where the function is implemented. Following the link will open the file in a buffer, even if the file 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.
Describe variable	• **C-h v** • **<f1> v**	(**describe-variable** VARIABLE &optional BUFFER FRAME)	- For example: **C-h v** load-path : shows the emacs lisp path. - Reference: https://www.gnu.org/software/emacs/manual/html_node/eintr/See-variable-current-value.html
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. • If INSERT (the prefix arg) is non-nil, insert the message in the buffer
Help on Input Method See also: ∑ **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 **Input Method** section for more info.
Describe encoding system **Describe buffers 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.
Describe language environment	• **C-h L** • **<f1> L**	(**describe-language-environment** LANGUAGE-NAME)	Describe how Emacs supports language environment LANGUAGE-NAME. • Prompts for language name, proposing the currently used language as the default. • Supports completion.
Helpful - extended help for Emacs with more contextual information	The helpful external package provides the same help information provided by Emacs with more contextual information and extra links. 📦 This requires the **helpful** external package PEL installs and activates it when the **pel-use-helpful** user-option is set. ☝ These commands provide a lot more information than standard Emacs help. Use then 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.

Description	Keystroke	Function	Note
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.
• **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. The following commands show available keys, help learning the key sequences, list the remaining available bindings, and list recent history of the typed keys and commands.		
List command history See also: ∑ 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	`<f11> ? k K`	(**which-key-mode** &optional ARG)	Toggle which-key-mode. When which-key mode is enabled, and 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). 📦 Requires the **which-key** package. PEL activates it when ***pel-use-which-key*** is t.
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: ∑ 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.
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. This function will also detect evil bindings made using 'evil-define-key' in this map. These bindings will depend on the current evil state. 📦 Requires the **which-key** package. PEL activates it when ***pel-use-which-key*** is t.
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. 📦 This requires the **keycast external package** PEL makes 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	`<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'.
	• By default, keys on 'free-keys-keys' list with no prefix sequence are considered, possibly together with modifier keys from 'free-keys-modifiers'. You can change the prefix sequence by hitting 'p' in the *Free keys* buffer. Prefix is supplied in format recognized by 'kbd', for example "C-x". • 📦 Requires the package free-keys. PEL activates this when the **pel-use-free-keys** user option is **t**.		
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: ∑ 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. The command 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. • You can use the minibuffer history commands `M-n` and `M-p` to get different commands to edit and resubmit.
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 ∑ 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 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**.
Help with Emacs Help, Apropos, and Info.	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, 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.
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.		

Description	Keystroke	Function	Note
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***.
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.		
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.
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 and should thus 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. 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. • With **C-u** prefix, also show variables, not just user options.
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. 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. 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.
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'.
Open specified info manual	`<f11> ? i m`	(**info-display-manual** MANUAL)	Prompt for a specific Info manual to open in a buffer. • Example: "eintr" := Introduction to Emacs Lisp.
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.
Find specified function function or variable in info	• **C-h S** • **<f1> F**	(**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 keys

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 : Quick navigation: highlight each target with a target key.
This uses the **ace-link external package** activated when the **pel-use-ace-link** user option is set to **t**.
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
- M-n opens a new, independent, Info buffer, that at first contains the same Info, but can be managed independently from the 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.

Programming Help Utilities

PEL provides key bindings for the following commands that are useful when editing source code files.

Show what completion mode is currently used.	`<f11> M-c ?`	(**pel-show-active-completion-mode**)	Display the completion mode currently used.

Description	Keystroke	Function	Note
Toggle which-function-mode to display name of current function at point See also: • ∑ Menus • ∑ Mode Line	• **<f11> ? f** • **<f11> M-l 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, in certain major modes. ⚠ 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 the major modes where you want this activated in the which-function-mode user-option with **M-x customize-option which-function-mode**. • With PEL you can use: • **<f11> ? <f3>** to access the which-func customization group. Note that it will provide access to the customization group even when the feature has not yet been loaded. • **<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.
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 =**		
Show window dimension	**<f11> ? d w**	(**pel-show-window-sizes**)	Show the height & width of the current window.
Display ASCII table See also: ∑ 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		
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>**		
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.
Display Memory Usage	**<f11> ? e m**	(**pel-emacs-mem-stats**)	Display Emacs memory statistics inside an *emacs-mem-stats* buffer.
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. The buffer is NOT committed to a file. • 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.		
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
Display Emacs initialization time with benchmark information if available	• **<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.
	• 📦 Uses the benchmark-init library to measure time of the various loaded modules.
	• Use **M-x list-package,** select benchmark-init and install it.
	• Then update your **init.el** file and 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.		

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 version	**<f11> ? e v**	(**emacs-version**)	Display Emacs version
Display Emacs executable path	**<f11> ? e x**	(**pel-emacs-executable**)	Display Emacs executable path in echo area.

Description	Keystroke	Function	Note
Print imenu controlling variables See also: ⅀ Menus	`<f11> ? e i`	(**pel-imenu-dbg-print-vars**)	Print the value of the imenu variables used to control the imenu functionality for the current buffer. • Print this information in a *imenu-dbg* buffer. • Use this when investigating the imenu support for a major mode: use as a (currently primitive) Emacs development tool.
List processes See also: ⅀ Shells	• `<f11> ? e C-p` • `<f11> z ?`	(**list-processes** &optional QUERY-ONLY BUFFER)	Display a list of all processes that are Emacs sub-processes. If optional argument QUERY-ONLY is non-nil, 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.
ESUP - Emacs Start Up Profiler	`<f11> ? 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.		
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:⅀ Customize	`<f11> ? e ?`	(**pel-package-info** &optional FULL-REPORT)	Display the following information in the echo area: • The number of PEL user-options, and the number of them that are active. • 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. • With optional argument, generates a full report with much more details in a *pel-user-options* report buffer. Any key prefix works. `M- - <f11> ? e ?` for example.
Display name of customization file. Show whether PEL dual independent customization is used or not. See also:⅀ 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**. See the OPTION A inside the **init-5.el example file**.
Display current Emacs Startup configuration setup See also: ⅀ 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.
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: ⅀ 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.
Using Man inside Emacs See also: • **𝔓 - Erlang** • **∑ 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 and opening hyperlinks. • The man command uses the system man utility, while woman is a complete implementation. It 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 𝔓 - Erlang table.		
Open a man page inside an Emacs buffer	• **<f11> ? m** • **⌘-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.		
Open a man page without external man process: woman	**<f11> ? w**	(**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 Windows). 🎛 PEL key sequence to customize man: **<f11> <f2> E w** • text width, use word at point, etc… 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.		
Open PEL PDF Help File See also: **≻Legend**	PEL includes a list of help PDF files such as this one for several topics. • You can open a local or GitHub hosted raw copy of these help PDF using the PEL help commands bound to the **<f1>** key available inside several PEL key prefixes. • For example several major modes bind the **<f12><f1>** key sequence to PEL help. Other such as help for abbreviations is **<f11> a <f1>** • The complete list of these key bindings is listed below. • Unfortunately not all Help PDF files have key sequences for them. However, you can: • Open any PDF file with the **pel-help-pdf-select** command: it prompts for a topic with tab completion support: use **<f11> ? p** • Open a dired buffer on the local directory where all PDF files are stored with **<f11> ? P** . Select the file(s) and type **z** to open the selected file(s). ☝Each PEL PDF uses icons and colour conventions. These conventions are described in the **≻Legend** table. ☝When opening a GitHub hosted raw PDF you will want to use a browser that is capable of rendering the PDF directly instead of downloading the file to your file-system. **Firefox** does this by default. You may need to install a plugin for other browsers. 🎛 These PEL functions select the local PDF or the remote GitHub raw PDF via a combination of factors: • By default it opens the local PDF file unless a prefix argument is specified by typing **C-u** or **M--** before the command key sequence. • If the **pel-flip-help-pdf-arg** user-option is set to **t**, it's the other way around: it opens the remote GitHub raw PDF by default. • By default, when opening a remote GitHub raw PDF it uses the default browse method selected by the **browse-url-browser-function** user-option. • The default value of the **browse-url-browser-function** user-option is to use the system browser. • You can force another one by setting PEL's **pel-browser-used** user-option to 'firefox or 'chrome. • When you do that the new selected browser is also used for the local file. If you want to use this browser for remote files only and keep Emacs' ability to use a PDF viewer for local files, then set the **pel-open-pdf-method** user-option to 'pdf-viewer.		
Open this PDF file.	**<f11> ? <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **∑ Help/Info** local PDF. If the prefix argument (like **C-u** or **M--**) is used, then it opens the remote GitHub hosted raw PDF instead. If the **pel-flip-help-pdf-arg** user-option is set it's the other way around.
Select and Open a PEL PDF file	**<f11> ? p**	(**pel-help-pdf-select** &optional OPEN-WEB-PAGE)	Prompt for a PEL PDF and open it. • By default it opens the local PDF file, but if the OPEN-WEB-PAGE argument is non-nil it opens the web-based PDF copy hosted on Github. • 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.	
∑ Abbreviations	**<f11> a <f1>**	Open **∑ Abbreviations** PDF file.	
∑ Align	**<f11> t a <f1>**	Open **∑ Align** PDF file.	
∑ Auto-Completion	**<f11> , <f1>**	Open **∑ Auto-Completion** PDF file.	
∑ Bookmarks	**<f11> ' <f1>**	Open **∑ Bookmarks** PDF file.	
∑ Buffers	**<f11> b <f1>**	Open **∑ Buffers** PDF file.	
∑ Case Conversions	**<f11> t <f1> 1**	Open **∑ Case Conversions** PDF file.	
∑ Comments	**<f11> ; <f1>**	Open **∑ Comments** PDF file.	
∑ Cut & Paste	• **<f11> = <f1>** • **<f11> - <f1>**	Open **∑ Cut & Paste** PDF file.	
∑ Counting	**<f11> c <f1>**	Open **∑ Counting** PDF file.	
∑ Cursor	**<f11> m <f1>**	Open **∑ Cursor** PDF file.	
∑ Customize	**<f11> <f2> <f1>**	Open **∑ Customize** PDF file.	
∑ Diff & Merge	**<f11> d <f1>**	Open **∑ Diff & Merge** PDF file.	
∑ Dired	**<f11> f <f1> 2**	Open **∑ Dired** PDF file.	
∑ Drawing	**<f11> D <f1>**	Open **∑ Drawing** PDF file.	
∑ Enriched Text	**<f11> t e <f1>**	Open **∑ Enriched Text** PDF file.	

Description	Keystroke	Function	Note
∑ Fast Startup	<f11> <f2> S <f1>	Open the ∑ Fast Startup PDF file.	
∑ File-mngt	<f11> f <f1> 1	Open ∑ File-mngt PDF file.	
∑ File/Directory Variables	<f11> f v <f1>	Open ∑ File/Directory Variables PDF file.	
∑ Filling/Justification	• <f11> t f <f1> • <f11> t j <f1>	Open ∑ Filling/Justification PDF file.	
∑ Frames	<f11> F <f1>	Open ∑ Frames PDF file.	
∑ Grep	<f11> g <f1>	Open ∑ Grep PDF file.	
∑ Help/Info	<f11> ? <f1>	Open ∑ Help/Info PDF file.	
∑ Hide/Show	<f11> M-/ <f1>	Open ∑ Hide/Show PDF file.	
∑ Highlight	<f11> h <f1>	Open ∑ Highlight PDF file.	
∑ Indentation	<f11> TAB <f1>	Open ∑ Indentation PDF file.	
∑ Input Method	<f11> t <f1> 2	Open ∑ Input Method PDF file.	
∑ Inserting Text	• <f11> i <f1> • <f11> y <f1> • <f11> _ <f1>	Open ∑ Inserting Text PDF file.	
∑ Keyboard Macros	<f11> k <f1>	Open ∑ Keyboard Macros PDF file.	
∑ Key-Chords	<f11> <f5> k <f1>	Open the ∑ Key-Chords PDF file.	
Line management. ∑ Display - Lines	<f11> l <f1>	Open ∑ Display - Lines PDF file.	
∑ Marking	<f11> . <f1>	Open ∑ Marking PDF file.	
∑ Mode Line	<f11> M-l <f1>	Open ∑ Mode Line PDF file.	
∑ Menus	<f11> <f10> <f1>	Open ∑ Menus PDF file.	
∑ Projectile	• <f11> <f8> <f1> • <f8> <f1>	Open ∑ Projectile PDF file. • The key sequence <f8> <f1> is available when the projectile mode is activated.	
∑ Registers	<f11> r <f1>	Open ∑ Registers PDF file.	
∑ Scrolling	<f11> | <f1>	Open ∑ Scrolling PDF file.	
∑ Search/Replace	<f11> s <f1>	Open ∑ Search/Replace PDF file.	
∑ Sessions	<f11> S <f1>	Open ∑ Sessions PDF file.	
∑ Shells	<f11> z <f1>	Open ∑ Shells PDF file.	
∑ Sorting	<f11> o <f1>	Open ∑ Sorting PDF file (o for ordering).	
∑ Speedbar	<f11> M-s <f1>	Open ∑ Speedbar PDF file.	
∑ Spell Checking	<f11> $ <f1>	Open ∑ Spell Checking PDF file.	
∑ Text Modes	• <f11> t <f1> 3 • <f11> t m <f1>	Open ∑ Text Modes PDF file.	
∑ Transpose	<f11> t t <f1>	Open ∑ Transpose PDF file.	
∑ Whitespace	<f11> t w <f1>	Open ∑ Whitespace PDF file.	
∑ Undo/Redo/Repeat/Arg	<f11> u <f1>	Open ∑ Undo/Redo/Repeat/Arg PDF file.	
∑ VCS-Mercurial	<f11> v <f1>	Open ∑ VCS-Mercurial PDF file.	
∑ Web	<f11> f <f1> 3	Open ∑ Web PDF file.	
∑ Windows	<f11> w <f1>	Open ∑ Windows PDF file.	
∑ Xref	<f11> X <f1>	Open ∑ 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. Inside another mode the longer key sequence that starts with <f11> SPC is available.		
𝔓ℓ- AppleScript	<f11> SPC a <f1>	Open 𝔓ℓ- AppleScript PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓ℓ - C	<f11> SPC c <f1>	Open 𝔓ℓ - C PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓ℓ - C++	<f11> SPC C <f1>	Open 𝔓ℓ - C++ PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓ℓ - Clojure	<f11> SPC C-j <f1>	Open 𝔓ℓ - Clojure PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓ℓ - D	<f11> SPC D <f1>	Open 𝔓ℓ - D PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓ℓ - Erlang	<f11> SPC e <f1>	Open 𝔓ℓ - Erlang PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓ℓ - Elixir	<f11> SPC x <f1>	Open 𝔓ℓ - Elixir PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓ℓ - Forth	<f11> SPC f <f1>	Open 𝔓ℓ - Forth PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		

Description	Keystroke	Function	Note
𝔓 - Go	<f11> SPC g <f1>	Open 𝔓 - Go PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - Hy	<f11> SPC C-h <f1>	Open 𝔓 - Hy PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - Gleam	<f11> SPC M-G <f1>	Open the 𝔓 - Gleam PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - Javascript	<f11> SPC i <f1>	Open 𝔓 - Hy PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - Julia	<f11> SPC j <f1>	Open 𝔓 - Julia PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔈𝔓 - Emacs Lisp	<f11> SPC l <f1>	Open 𝔈𝔓 - Emacs Lisp PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - Common Lisp	<f11> SPC L <f1>	Open 𝔓 - Common Lisp PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - LFE	<f11> SPC C-l <f1>	Open 𝔓 - LFE PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - NetRexx	<f11> SPC N <f1>	Open 𝔓 - NetRexx PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - Python	<f11> SPC p <f1>	Open 𝔓 - Python PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - REXX	<f11> SPC R <f1>	Open 𝔓 - REXX PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - Rust	<f11> SPC r <f1>	Open 𝔓 - Rust PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - Scheme	<f11> SPC C-s <f1>	Open 𝔓 - Scheme PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝔓 - V 🚧	<f11> SPC v <f1>	Open 𝔓 - V PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
• **Build Tools**			
𝔓 - Make	<f11> SPC M <f1>	Open 𝔓 - Make PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
• **Markup languages**			
𝕄 Graphviz Dot	<f11> SPC M-g <f1>	Open 𝕄 Graphviz Dot **PDF using method specified by the pel-open-pdf-method user-option or the alternate one if a command prefix (like C-u) was used.**	
	<f12> <f1>		
𝕄 Outline/Org-Mode	<f11> SPC M-o <f1>	Open 𝕄 Outline/Org-Mode PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝕄 PlantUML	• <f11> D u <f1> • <f11> SPC M-u <f1>	Open 𝕄 PlantUML PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝕄 Markdown	<f11> SPC M-m <f1>	Open 𝕄 Markdown PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		
𝕄 reStructuredText	<f11> SPC M-r <f1>	Open 𝕄 reStructuredText PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **C-u**) was used.	
	<f12> <f1>		

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	

Topic & Link	Description
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.