Document Legend

Symbol/ colour	Purpose	Description and Examples
Table Title	Table titles (and the equivalent PDF file titles showing in browser tabs) use a prefix symbol to identify the overall purpose of the table. Most table titles use a single symbol, some, like the tables related to Emacs Lisp, use more than one.	
➢	Document related	This table contains information related to the overall document only, such as this legend.
∑	Generic Emacs	The ∑ symbol is used in the table titles of tables that describe general Emacs feature set such as abbreviation, bookmarks, buffers, navigation, etc… In opposition to features specific to a specific major or minor mode.
𝔏	Emacs Lisp	This table contains information related to Emacs Lisp Development.
𝔓	Programming language - Major Mode	The two stylized PL letters are used in the table titles of tables that describe a specific set of Emacs major modes specifically designed to support the editing of files written in a specific programming language.
𝔐	Markup language - Major Mode	The stylized M letter is used in the table titles that describe a specific set of Emacs major modes specifically designed to support the editing a markup languages like reStructuredText, markdown, XML, HTML, etc…
ⓜ	Minor Mode	Information about a specific minor mode
𝕋	Templates	The table describes a text template system used to insert templated text quickly.
⌨	Keys/Keyboard	The table describes use of keys such as function keys, numeric keypad, modifier keys.
	macOS specific	The table describes macOS topics only.
※	Reference	The table contains references to web pages for documentation, examples, tutorial, etc…
☞	Uncompleted	This tab is not completed yet and needs further work.
🚧	Todo	Page in very early stage. Much more work left to do.
Colour Codes	The following colour codes describe the context, availability, of the key bindings described in the tables.	
Black	Standard global key binding	Bindings that are mostly always available in standard Emacs: global key bindings.

Underlined	Key chord keys	With PEL user option **pel-use-key-chord** set to **t**, PEL activates the key-chord external package along with global and mode-specific key chord bindings identified in the **pel-key-chords** user option.
		• A key chord is a group of 2 normal, non-modifier keys that must be typed simultaneously to activate the action identified in the key chord definition.
		  • This is **not** something like C-s, where the Control key and the s key are type together to do a CONTROL-S or where M-b represents using the Meta key and the b key together.
		• PEL default for **pel-key-chords** are identified in the tables of this document with the characters underlined.
		  • In some cases the key-chord is a simple binding to execute a command. In that case the 2 key-chord keys are shown in the keystroke column alone, simply underlined.
		  • In other cases, the key-chord inserts characters and execute commands. In such as case, the 2 key-chord keys are also shown in the keystroke column alone, but instead of describing the function in the function column, the cell shows the key-chord string which represent both the character inserted and the key code for the command.
		  • For example, the key-chord that consist of typing the **<** key and the **>** key together is represented as the **<>** key-chord and the expansion is show as **"<>\C-b"** . The effect is to insert both angle brackets and put point in between, since **C-b** is bound to to command backward-char.
		• The color of the key-chord corresponds to the availability of the commands used, if any. A key-chord that depends only on Emacs standard commands or simple characters is therefore shown in black.
		• The use of key-chords can be toggled by executing `M-x key-chord-mode`, which PEL binds to **<f11> M-k** when **pel-use-key-chord** is set to **t**.
		• For more information on PEL key-chords see the ∑ Key-Chord table.

Violet	Standard global key binding for Emacs execution	Same as above except that the keys are available to provide access to the various Emacs Lisp functions by name or execution of Emacs Lisp forms from the minibuffer.
Light Green	PEL key bindings	• When used to highlight key sequence: denotes a key sequence available in the PEL package. • When used to highlight an Emacs Lisp command, indicates that the code is implemented by the PEL package.
Dark Green	Context sensitive PEL bindings	Used to highlight key sequences available in the PEL package that are selected by the mode of the current buffer. These are special local mode bindings, but specific to the PEL system and which may be active while other modes are active. Most PEL specific bindings use function keys prefix to avoid clashes with currently popular bindings.
Light Teal	PEL prefix to external package key map	Several packages come with pre-defined key maps and prefix key suggestions. If the prefix key is different than the prefix suggested by the package but the remaining keys remain the same in the key map, this colour is used for the keys.
Teal	Hydra key	With PEL user option **pel-use-hydra** set to **t**, PEL activates the **hydra external package** and also creates several Hydra key sets. The keys that activate the hydra are identified using the teal color.
Red	Remapped Key	Identifies a key sequence normally bound by Emacs that PEL remaps for other purpose.
Orange	Key sequence not available in terminal mode	Key sequences highlighted in `orange` are not available in terminal (termcap) mode, inside a OS shell. They are only available when emacs in running in graphics mode.
Light Blue	Special behaviour in terminal mode	Highlight key sequences or notes that apply to the way the key binding operates when Emacs is running in a terminal (tty) frame.
Blue	Command using external package	Key sequence set-up by the PEL package that uses a command from an external package that must be installed and loaded. • The blue color is used if the key binding is defined by the external package itself. • If PEL defines it's own binding for the key then the key colouring is one of PEL's keybinding color (either light or dark green). • In all cases the description of the key includes a "requires external package" note using the 📦 symbol.
Dark Blue	Command using external package	Reserved. Consolidation of the colouring is underway in this document.
Coral	Command available only in special mode	Some commands, like debugger commands, are available through simple keys in the buffer where the mode is active, but are also available in other buffers, where the mode is not active. The commands that are available in the buffer where the mode is active are coloured in **coral** color. These constitute "local" bindings (for the specific mode) and may therefore conflict with the global binding other local binding of the same key chord.

Symbol/ colour	Purpose	Description and Examples
Grey	• Translated key • Using **`M-x`** • Using **`M-:`**	The grey colour is used for the following cases: • Emacs translates some key combinations to other keys. For example Shift-F5, <S-f5>, is translated to <f5> so any binding to <f5> is also accessible if the shift key is pressed while the F5 key is pressed. The original keys do not have a specific key binding. They could have one. Of course in some case you may want to keep it unchanged (for example because the Shift key may indicate something like starting or extending the mark in transient mark mode). In any case, when such a key is described in the various tables, the colour of that key is grey. • All Emacs commands (functions that are marked interactive) can be executed via the **`M-x`** command. Some of the commands have no other key bindings. If there is no other pertinent information to show for the command (such as marking the command orange because it is not available in Terminal/TTY mode), then the **`M-x`** binding is shown it is displayed in grey as a reminder that it's not a specific binding, just a use of the (execute-extended-command) command, which is bound to **`M-x`**. • Emacs also allow execution of any Emacs Lisp expression using the (eval-expression) command bound to the **`M-:`** key. Those are also coloured grey.
Light Grey	Unused key binding	A key binding that is not necessary and could be re-used for some other functionality.
Cell Colours	The background Colour of some cells is coloured to indicate various contexts. The description of these contexts follows.	
	Standard global key binding used by major/ minor mode that is same as another major/minor mode.	Bindings that are similar to the binding "mostly always available in standard Emacs", shown in black, (re) used in a specific major mode. The yellow cell background is only there as a reminder that the key binding and behaviour is also described in a Emacs Generic table (a table with a ∑ title prefix.) For those there is often a reference to the generic table inside the "Operation" column. The yellow background is also used to highlight the fact that the same key binding is also described inside another page. For example the follow-mode key binding is described in both scrolling and window pages.
	Key binding specific to some major modes	Several tables describe generic concepts such as navigation or handling comments. They mostly describe global key bindings. But they may also describe key bindings that are only available in some major modes, like a special key binding to comment lines of a particular set of programming languages. Those bindings use this background colour. For tables describing the bindings of major modes, this background colour is not used.
Notes		
★★	Powerful command	Used in the **Keystroke** column to highlight commands that are specially powerful in the sense that they integrate a relatively large and useful set of features. The description of these commands should be read carefully and fully understood.
📦	Requires External Package	This symbol identify features that depend on external packages. • Some of PEL's features require the use of external packages, packages that are not part of the standard Emacs installation and which must be installed separately. Most are Emacs Lisp packages, but some also require external applications. • PEL customization capabilities allow you to identify whether or not you want to use the features that depend on such external packages. PEL also attempt to use or implement code that can help you install the required package(s).
⚠️	Caution / Limitations	Describes surprising impacts or behaviours that might have important negative impacts. It is also used to highlight limitations.
✂	Key Binding Modification	The PEL package mostly tries to avoid modifying the binding of standard Emacs keys. But there are exceptions. This symbol is used to indicate such exception.
☝	Pointing up note	The text provides a specific explanation of the way the command works.
☛	General Note	A mention of something to remember.
💡	Idea	Identifies an interesting, useful, use of an Emacs feature.
📜	Historical Note	Describes key sequence bindings that were available in versions of Emacs older than version 26. Often contains a reference to a command, functions or variable alias still supported to permit the execution of code that still uses the old names.
❖	Windows identifier	Indicates a note that applies to the Windows OS implementation of Emacs.
🍎	Mac OS Identifier	Indicates a note that applies to the macOS implementation of Emacs or that the feature is only available when Emacs runs under macOS.
🎛	PEL Customizable	This feature is customizable via the **Pel** customization group or Inside Emacs there are several ways to access the customization system: • **`M-x customize`** : access Emacs top-level customization system • **`M-x customize-group`** : access to the group itself. For Pel, type: **`M-x customize-group Pel`** • **`M-x customize-option:`** customize a specific user option variable. ☝Read the ☝note below. It applies to PEL user option variables as well.
🎛	Customizable	This (non-PEL) feature is customizable through the list of user options described in the cell. ☝All user options variables can be set in Emacs customization (see commands in the row above) and stored in file that is loaded when Emacs starts. You can also set these variables in directory local files and also inside files (via the file local settings). So you can have a general setting for a variable, specialize it for a directory tree and specialize it further for a given file. Emacs allows you to fine tune these user options at the level you want. This is very powerful and flexible. See: • Emacs Customize • Per directory local variables • Per file local variables
🗂	Technical Detail - file locations	Several features store information in various locations. Notes describing where implementation files are stored are identified with this icon.
💻	Implementation detail	A note describing how the command is implemented.
👽	Special technical note	The document includes description of some boundary technical situations you may very well want to skip unless you are interested by internal details for the sake of technical interest. But for most people these will probably not be useful, might even look alien, and won't need to know that to use the Emacs feature or command.
🚧	Work in Progress	Identifies an incomplete area, more work is required to complete the information presented. Often accompanied with an explicit TODO note.
🐞	Bug	Identifies a bug detected in software or documentation. Normally used to identify problems in software or documentation used by PEL. In some cases, I have submitted a bug report (in which case the link to the bug report is included).
Keys	Most key binding description in the various tables use the standard Emacs key sequence notation like **`M-a`** (meaning Meta key and 'a' key down together). But sometimes, other keys are described with the following symbols.	
❖	Windows Key	Identifies the Windows key on a Windows OS PC.
⌘	macOS Command key	Identifies the macOS Command key, often used as the Emacs super (s-) key modifier.
⌥	macOS Option	Identifies the macOS Option key, often used as the Emacs meta key. ☝Note that inside macOS Terminal.app you can toggle the meaning of that key between macOs Option and Meta by typing **`⌘⌥o`**. While using Emacs in Terminal.app make sure that the key is set as Meta. You can use the **`⌘⌥o`** chord to quickly change it when typing accentuated letters or other symbols without having to change Emacs input method.
⇧	Shift	
^	Control	

Symbol/ colour	Purpose	Description and Examples
⌦	Delete forward	
⌫	Detete backward	
Fonts	Most of the text in the various tables follows a convention in the use of fonts. These conventions are listed here.	
Table Title	**Helvetica Neue - Bold- 20 pt.**	Used at the top of every table.
Section level 1	**Helvetica Neue - Bold- 15 pt.**	Top level section in most tables. Sometimes the first level is skipped and the second level is used first if the title contains too many letters and we want to reduce the vertical size of the row.
Section level 2	**Helvetica Neue - Bold- 13 pt.**	Identifies a secondary (sometimes a primary level - see above) section in the table.
Section level 3	**Helvetica Neue - Bold- 12 pt.**	Lowest grouping of lines.
Line title	**Helvetica Neue - Bold- 11 pt.**	What is shown in the first column of this line.
Main text	Helvetica Neue - 10 pt.	The section main text uses that font. If a key or code concept is included it uses the following fonts.
Keys	**`Courier - Bold - 11 pt.`**	To make key bindings, like **`C-M-u`**, stand out.
Code Concepts/ Keywords	**American Typewriter, bold, 10 pt.**	To make code concepts, keywords or lexical elements stand out like **`if`**, **`#define`**, **`#error`**, **`{`**, **`}`**, **`>>`**, etc…
Names	**PEL Code Naming Conventions**	
pel-	**'public' function or command**	A PEL function or interactive command that can be executed from anywhere.
pel--	**'private' function**	A PEL function that should not be used externally; these are meant to be used only by appropriate PEL code.
pel-∑-	**public' hydra command**	These are 'public' commands that can be used anywhere. The ∑ symbol is used instead of the word '**hydra**' in command names to shorten them a little. This helps showing a larger portion o the command name when the prefix key is typed and which-mode shows the list of commands in the minibuffer. • ☝Under macOS, typing ∑ is done using the **`⌥w`** key combination. • Note that when Emacs runs in terminal mode you can use **`⌘⌥o`** to switch the meaning of the ⌥ key between Option and Meta.