PEL Topics Index

Emacs Reference Cards

☝With PEL you can access these via the `<f11> ? e r` key sequence.
See ∑ Help/Info

These are links to the PDF version of official English version of the quick reference cards for GNU Emacs and popular external packages. PEL documents Emacs key bindings as well, these cards provide useful complement to what PEL provides.

Emacs	Calc	Gnus	Magit Cheatsheet	Org	Viper
Emacs survival card	Dired	Gnus booklet	Magit Ref-card		VIP

➢ PEL Overview

- PEL repo
- PEL Readme
- PEL Manual
- PEL NEWS

This table holds links to the PEL file tables. Each cell holds a hyperlink to the GitHub hosted raw PDF table.

☝For the best user experience, use a browser that can render PDF directly instead of downloading.
- Mozilla Firefox (version > 78) does that perfectly. You may need to activate a plug-in for other browsers.
- With that in place, you can browse through all the PDFs quickly and reach a vast amount of information quickly.

☝From within Emacs open this topic index PDF by typing the `<f11> ? <f1>` key sequence. More help topics with `<f11> ? p` keys.

☝The symbols, colour coding and various other conventions are described in the ➢Legend PDF.

- General Information.
- Development Information
- Migration Guide

➢Legend	➢Recommended Emacs User Option		➢Themes		
➢PEL	iMenu/Speedbar support		PEL Naming Conventions		
➢CRiSP ⇏ Emacs					

OS Desktop Key Bindings

(Bindings that don't clash with PEL)

	macOS Keys	Ubuntu 16.04 Desktop Keys			
	terminal settings	Mint 20 Desktop Keys			

Feature Comparisons

Completion Modes Compatibility | Speedbar/iMenu Mode Compatibility | Shells/Terminals Comparisons

Key Prefixes & Suffixes

∑ Modifier Keys | ∑ Numkeypad | ➢PEL | Keys - Fn | Keys - F11

∑ Emacs Features

See a Guided Tour of Emacs.

The PEL tables named at right ☛ describe the Emacs commands and key bindings for generic Emacs concepts and features.

Emacs commands can be executed by name or bound to key sequences. The commands may have ***arguments*** and keys can express them.
See:
- Emacs Keys
- Numeric Arguments

You can also:
- Run Command by Name

Emacs uses a concept of modes.
See:
- Emacs Major and Minor Modes
  - Major Modes
  - Minor Modes
  - Choosing Modes

PEL provides several key sequences to toggle minor modes, described in the relevant PDFs.

The links that start with only ∑ Emacs generic features, the blue links are external packages. The green links are mostly PEL extensions.

∑ Abbreviations	∑ Cursor	∑ Filling/ Justification	𝔓𝔩- Lispy	∑ Scrolling	∑ Time Tracking
∑ Align	∑ Customize	∑ Frames	∑ Marking	∑ Search/Replace	∑ Transpose
∑ Auto-Completion	∑ Cut & Paste	∑ Grep	∑ Menus	∑ Semantic	∑𝔵 Treemacs
∑ Autosave/Backup	∑ Diff & Merge	∑ Help/Info	∑ Mode Line	∑ Sessions	∑ Undo/Redo/ Repeat/Arg
∑ Bookmarks	∑ Dired	∑ Hide/Show	∑ Mouse	∑ Shells, REPLs & terminal emulators	∑ VCS-Git 𝔵Magit
∑ Buffers	∑ Display - Lines	∑ Highlight (colors)	∑ Narrowing	∑𝔵 Smartparens	∑ VCS-Mercurial
∑ Case Conversions	∑ Drawing	∑ ibuffer-mode	∑ Navigation	∑ Sorting	∑ VCS-Subversion
∑ Closing/ Suspending	∑ Enriched Text	∑ Indentation	∑ Outline	∑ Speedbar	∑ Web
∑ Comments	∑ Faces/Fonts	∑ Input Method	∑ Packages	∑ Spell Checking	∑ Whitespace
∑ Completion/Input	∑P Fast Startup	∑ Inserting Text	∑𝔵 Projectile	∑ SyntaxCheck	∑ Windows
∑ Counting	∑ File-mngt	∑ Key-Chords	∑ Rectangles	T Templates	∑ Xref - Cross References
∑M CUA	∑ File/Directory Variables	∑ Keyboard Macros	∑ Registers	∑ Text Modes	

𝔵𝔓𝔩 - Emacs Lisp concepts & tools

𝔵 ERT (Emacs Lisp Regression Testing) | 𝔵 Hooks | 𝔵※ - Emacs Lisp Types

XRef - Cross Reference Tools

See also: ∑ Xref

Emacs supports various cross reference mechanisms described in the ∑ Xref table. These mechanisms take advantage of various external tools and integrate with them. Notes about those tools are available in the tables listed in this section. 🚧 This is work in progress.

Xref-Support | Xref-Backend

Build Tools & Preprocessor

PEL supports installation and partial setup of the following tools: ☛

PEL has support for several build tools but they are not all documented in a page.
- Nix 📦 Requires nix-mode external package ↪ activated when **pel-use-nix-mode** user-option is tuned on.
- Tup 📦 Requires tup-mode external package ↪ activated when **pel-use-tup** user-option is tuned on.

𝔓𝔩 - M4	𝔓𝔩 - Make

Data Serialization

Ⓓ CWL | Ⓓ YAML

Data Modelling/ Specification

Ⓢ ASN.1 asn1-mode | Ⓢ MIB snmp-mode | Ⓢ YANG

Hardware Description Languages

Verilog 🚧future | VHDL 🚧future

Text Markup Languages

M↓ AsciiDoc | M↓ Markdown | M↓ Org-Mode | M↓ reStructuredText

• Graphics Markup

M↓ Graphviz Dot | M↓ MscGen | M↓ PlantUML

Programming Languages

Main Paradigm of Programming Language Families
- ***Actor Model:*** Ⓐ
- ***Concatenative*** Ⓚ
- ***Concurrent:*** ⓒ
- ***Functional:*** ⓕ ***Pure:*** Ⓕ
- ***Imperative:*** ⓘ ***or no token***
- ***Has Syntactic Macros:*** ⓜ

- The programming languages supported by PEL are listed here in alphabetical order.
- PEL also provides basic support for other programming languages not listed here.
- Emacs supports other programming languages directly, not listed here.

Future support for Crystal, Elm, Kotlin, Lua, Purescript, ReasonML, Seed7, Typescript, Zig and documentation of support for Ada, Fortran, Javascript, Java, Modula, Pascal (based on my need for them or requests (if any)).

Emacs has major mode support for several programming languages. PEL currently adds extra support for some of them, listed below.

BEAM Programming Languages	*Functional Languages*	Javascript target	Lisp Family Languages	Lisp-like Languages	Command Line Scripting Languages
Curly Bracket Languages	Java Virtual Machine Languages	ML Family Languages	Scheme Language Dialects	Stack Based Languages	OS App Control Scripting Languages

The following lists the programming languages in alphabetical order.
- The cell colours give a coarse indication of the programming language family(ies).

𝔓𝔩- AppleScript	Common Lisp ⓕⓜ	𝔓𝔩 - Forth Ⓚ	𝔓𝔩 - Janet ⓘⓕⓜ	𝔓𝔩 - Nim ⓜ	𝔓𝔩 - Ruby
Ada 🚧future	Crystal 🚧future	Fortran 🚧future	Java 🚧future	𝔓𝔩 - OCaml ⓘⓕ	𝔓𝔩 - Rust
𝔓𝔩 - Arc ⓕⓜ	𝔓𝔩 - D ⓘⓕⒶ	𝔓𝔩 - Gambit ⓕⓜ	𝔓𝔩 - Javascript 🚧	Pascal 🚧future	𝔓𝔩 - Scheme ⓕⓜ
𝔓𝔩 - C	Eiffel 🚧future	𝔓𝔩 - Gerbil ⓕⓜⒶ	𝔓𝔩 - Julia ⓜ	𝔓𝔩 - Perl	Seed7 🚧future
𝔓𝔩 - C++	𝔓𝔩 - Elm 🚧future Ⓕ	𝔓𝔩 - GNU Guile ⓕⓜ	Kotlin 🚧future	𝔓𝔩 - Python	𝔓𝔩 - Tcl 🚧future ⓕⓘ
𝔓𝔩 - Chez ⓕⓜ	𝔓𝔩 - Elixir ⓒⓜⓕⒶ	𝔓𝔩 - Gleam	𝔓𝔩 - LFE ⓒⓜⓕⒶ	𝔓𝔩 - Purescript Ⓕ	𝔓𝔩 - Typescript 🚧
𝔓𝔩 - Chibi ⓕⓜ	𝔵𝔓𝔩 - Emacs Lisp	𝔓𝔩 - Go	Lua 🚧future	𝔓𝔩 - Racket ⓕⓜ	𝔓𝔩 - UNIX Shell
𝔓𝔩 - Chicken ⓕⓜ	𝔓𝔩 - Erlang ⓒⓕⒶ	𝔓𝔩 - Haskell Ⓕ	Modula 🚧future	𝔓𝔩 - ReasonML 🚧	𝔓𝔩 - V
𝔓𝔩 - Clojure ⓕⓜ	Factor 🚧future Ⓚ	𝔓𝔩 - Hy *(python)* ⓜ	𝔓𝔩 - NetRexx	𝔓𝔩 - REXX	Zig 🚧future