↖ Topic Index

Topic & link	Description
Last updated on:	2026-01-21
Books	
Writing GNU Emacs Extensions - O'Reilly by Bob Glickstein, July 2010	A good book that provides insight on how to use the various facilities to write good Emacs Lisp code. Emacs has evolved since the book was written but almost everything in the book still applies as of Emacs version 26.
Emacs Lisp Coding Guidelines	• Emacs Lisp Coding Thoughts provides several ideas.
Lisp Style	
Lisp Indentation Style @ Wikipedia	The Lisp Style is shown for some Common Lisp code but also applied to C and happens to be also very similar to the Python style (although in Python the blocks are simply indented; no parens character is used).
Lisp Editing - Parenthesis Highlighting	Several Emacs packages have been written to help highlight the parens. Emacs packages and modes include show-paren-mode, rainbow-delimiters and paren-face. PEL uses show-paren-mode and rainbow-delimiters
show-paren mode @ Emacs Manual	The paren.el is part of Emacs and implements the show-paren mode, which highlights the parens that matches the one before or after point.
rainbow-delimiters @ GitHub	The rainbow-delimiters mode allows colouring rareness according to their depth. When Emacs is used in Graphics mode it's also possible to assign different sizes as shown by Xah Lee in the ErgoEmacs Colored Nested Brackets page. The EmacsWiki Rainbow Delimiters page describes how to setup hooks that activate the mode automatically for some files.
paren-face @ GitHub	Defines a face named parenthesis used for the parentheses character, with the intention of dimming the parentheses to help show the real structure of Lisp code via indentation. The parinfer mode does something similar (if dims the closing parentheses).
Lisp Editing - Parenthesis Management	Several Emacs packages have been written to help the editing process. These include the following listed packages: adjust-parens, lispy, paredit, paxedit, parinfer, smartparens and probably several others.
Lisp Editing @ WikEmacs	This WikEmacs page describes several of those packages with editing scenarios
ParInfer	The parinfer package provides modes that infer the parenthesis.
ParInfer Documentation	The documentation allows live interaction
ParInfer Mode Implementation for Emacs (in Emacs Lisp)	Emacs Lisp code for ParInfer for Emacs. Describes how to install and configure ParInfer.
Highlighting Emacs Lisp Code	The default emacs-lisp-mode highlights the Emacs Lisp code available in the buffer. Emacs Lisp is a Lisp-2; so a symbol can be a variable and/or a function: each symbol has a link to variable definition, function definition and a property alist. Furthermore, there are different *kind* of functions: lambda, compiled-byte functions (autoloaded or not), macros (autoloaded or not), primitive (written in C), special forms (primitive written in C that treat the list differently). And there can be indirection and advices. There's also variation in the "kind" of variables: there's global variables, local variables, closures, etc… The standard highlighting does not show all of this information; the designers considered that it would be too distracting; just some of the information is available via highlighting. Some have different views and developed modes that highlight Emacs Lisp code differently. These modes are listed here.
highlight-defined @ MELPA	The highlight-defined package provides the highlight-defined-mode, a minor mode that highlights defined symbols. It has the ability to highlights differently different "*kind*" of function symbols. • Unfortunately it does not consider the semantic of the code enough in the selection of the highlighting. For example if you define a macro named while-n, the face you specify for macros won't be used for code that invokes the macro in a macro call form, however it will use that face if you specify a symbol like 'while-n in any list position except the first one. That mean it will be highlighted in the argument list (but not if the symbol is the first argument). • I would prefer highlighting to follow the code semantics, and perhaps have a customization option to colonize the arguments & variables that use the same name as functions. It might be difficult to do this in a minor mode. I'll have to investigate more.
The Emacs Lisp Mode Syntax Coloring Problem — Xah Lee	Xah Lee describes the problem he saw in the colouring. He tried to request changes to the Emacs developers, create a bug report and that was closed. So he wrote his own code. It's a new major mode: xah-elisp-mode @ MELPA
Emacs Lisp Bytecode	
GNU Emacs Lisp Bytecode Reference Manual	Describes the Emacs Lisp byte code and LAP, Lisp Assembly Program. • From the elisp-bytecode GitHub project.
Inspecting Emacs Lisp	
The helpful package	The helpful package provides alternate help commands for commands, functions, variables, etc. It provides much more information than Emacs standard help and presents it in buffer with lots of buttons that you can use to activate several things like the debugger, tracing. You can de-assemble functions for example. Worth using.
The elisp-refs package	Very powerful reference detection package but detects only references in code **already loaded**. See also: Searching A Million Lines Of Lisp which provides some description of the package.
Debugging Emacs Lisp	
An Introduction to Programming in Emacs Lisp - Debugging	A gentle introduction/overview of debugging Emacs Lisp with both debug and edebug, with examples.
GNU Emacs Lisp Manual: Debugging Lisp Programs	Extensive description of both debug and edebug.
How to debug elisp? @ stackOverflow	A discussion on debugging Emacs Lisp for a very quick oveview. Contribution from Drew Adams, Trey Jackson and Artur Malabarba.
Debugging Basics - Nic Ferrier's Youtube video	A 11 minute video showing a simple debugging session with edebug. Aside from the keyboard noise I find annoying, this video gives a good introduction of what can be done with EDebug, and also covers debugging of macros using macrostep to expand the macro before debugging to be able to see the execution inside the macro code.
Profiling Emacs Lisp	
GNU Emacs Lisp Manual: Profiling	Brief description of the built-in profiler and the elp package.
EmacsWiki - Emacs Native Profiler	List more functions than the GNU manual…
EmacsWiki - Emacs Lisp Profiler	Better description of the elp profiler.
Test Coverage	
Interesting Emacs Lisp Libraries / Files	The following describe useful Emacs Lisp libraries that might be useful writing more Emacs Lisp code.
memoize.el — Chris Wellons Elisp Memoize @ null program	Provides function and macros to memoize Emacs Lisp functions, caching values of time-intensive computations. See **Memoization @ Wikipedia** for an overview of this technique.