↖ Topic Index

Emacs Tree Sitter Support 🚧

Operation	Keystroke	Function	Note
Tree-Sitter Support ○ **Help & Customization** • Exploring Syntaxes ○ **Combobulate** • **References** See also: 🚦**Tree-sitter** feature table. Location of Tree-Sitter Language Grammar files ➽ Setup Emacs for Tree-Sitter: ➽ Emacs design choice limitation: ➽ PEL solution for more flexible major mode selection ➽	Emacs provides built-in Tree-Sitter syntax parsing since Emacs 29 but version 30 has a breaking change in the way Tree-Sitter modes are handled. • Because of that change **PEL supports Tree-sitter on Emacs >= 30 only** and when 🤖 **pel-use-tree-sitter** customizable user option is set to **t**. • To use these features Emacs must be built with tree-sitter support, and the Tree-Sitter language grammar files must be installed. **Package providing Tree-Sitter Language Grammars:** • There are now several Emacs packages implementing tree-sitter grammar for various major modes. • They are accessible via the Emacs package-list-packages command (**`M-x package-list-packages`**) See **Σ Packages** • As PEL explicitly incorporate Tree-Sitter support for various major modes, PEL will add automatic installation of some of those packages. • The 📦 **tree-sitter-langs** external package is a language bundle that holds the pre-compiled grammars for several languages a support Linux, macOS and Windows. 🤖 PEL installs it automatically when the **pel-use-tree-sitter** customizable user option is set to **t**. **Directory holding Language Grammar:** • Emacs searches for the Tree-Sitter grammar implementation dynamically loadable libraries in the directories identified by the **treesit-extra-load-path** variable. This is variable, not a customizable user option. • PEL provides the **pel-treesit-load-path** customizable user-option, which holds a list of directories that hold Tree-Sitter grammar dynamically loadable library files. On startup PEL appends its content to the **treesit-extra-load-path**, providing a customizable way to identify the location of the Tree-Sitter grammar dynamically loadable library files. The default value is an empty list. ☝**Recommendation**: follow the instructions in **Using Tree-Sitter with Emacs and PEL** to setup your Emacs environment for Tree-Sitter. **Controlling whether the classic or tree-sitter major mode should be used** • As described in the Emacs News note identified by the above link, the content of the **major-mode-remap-alist** determines whether loading the tree-sitter aware mode once should force Emacs to continue using the tree-sitter aware mode when opening the same type of files later. ⚠️Unfortunately it was never Emacs designers intent to allow using both the classic and the tree-sitter based major mode for a given type of file. Their intent was that as soon as a command executing a tree-sitter major mode was issued, every buffer of the same type would be opened using the tree-sitter based major mode. That's the way Emacs works as of 30.2 and it will most probably continue to behave this way in Emacs 31. • That's not the way I wanted Emacs under PEL to behave: I want to be able to configure support for a type of file to use either a classic major mode or a tree-sitter major mode and that would be how Emacs would open the buffer by default. I wanted the user to be able to switch to a classic major mode or to tree-sitter major mode for the current buffer at any time, without any impact on which mode subsequent buffer opening would use. • PEL has extra logic that ensures a more flexible behaviour: it associates mode dispatcher functions to control which major mode is used and, when required, executes a fixer function after modes are loaded to repair the auto-mode-alist to ensure that the PEL mode dispatcher remains used. • For each type of file where PEL supports tree-sitter major modes, you select whether you want to use the classic major mode or the tree-sitter major mode with the value set to the customizable user-option that activates PEL support for the type of file. **For example**, PEL activates support for Go (**Φ - Go**) when the **pel-use-go** customizable user-option is turned on with 2 choices: • **t:** request using the classic major mode (**go-mode** for Go) • Each time you open a Go file, Emacs will use **go-mode**. • Once the file is opened, you can explicitly activate the tree-sitter **go-ts-mode** by typing **`M-x go-ts-mode`** or execute the **pel-treesit-toggle-mode** command by typing **`<f11> C-t C-t`**. It only affects the current current buffer. If you open another Go file, it will use the **go-mode** as identified by the value of **pel-use-go**. • **with-tree-sitter**: request using the tree-sitter aware mode (**go-ts-mode** for Go) • Each time you open a Go file, Emacs will use **go-ts-mode**. • You can explicitly activate the classic **go-mode** by typing **`M-x go-mode`** or type **`<f11> C-t C-t`**. It only affects the current buffer. But if you open another Go file, it will use the **go-ts-mode**.		
Last updated on:	2025-12-28	PEL provides extra key binding for operations related to tree sitter. They are described below.	
Open this PDF file. See also: **Σ Help/Info**	**`<f11> C-t <f1>`**	(**pel-treesit-help** &optional OPEN-WEB-PAGE)	Open the **Σ Tree-Sitter** 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.
Σ Customize PEL tree-sitter control	**`<f11> C-t <f2>`**	(**pel-treesit-customize** &optional OTHER-WINDOW)	Customize PEL support for: open the pel-pkg-for-tree-sitter customization group buffer in current window. • If OTHER-WINDOW is non-nil (use **`C-u`**), display in other window.
Σ Customize Emacs tree-sitter control	**`<f11> C-t <f3>`**	(**pel-treesit-emacs-customize** &optional OTHER-WINDOW)	Customize Emacs tree-sitter support customization group: treesit, combobulate. • If OTHER-WINDOW is non-nil (use **`C-u`**), display in other window. ☝ **Tree-Sitter based highlighting** has 4 level of highlighting, controlled by the value of the customizable **treesit-font-lock-level** user-option. The levels are: • Level 1 : comment, definition • Level 2 : keyword, preprocessor, string, type • Level 3 : attribute, assignment, constant, control, function, number, operator • Level 4 : bracket, delimiter, error, label
Check validity of Tree-Sitter language grammar setup	**`<f11> C-t ?`**	(**pel-treesit-check-setup**)	Check the Emacs Tree-Sitter environment and report problems as warnings with an error count printed as message. Prints only a message if all is OK.
Toggle between classic-mode and tree-sitter based mode	For several major modes, PEL supports the ability to select the classic or the Tree-Sitter based mode as the default mode for buffers of the given type. That selection can be changed without impacting the ability to use the other mode. • This capability is normally not available under Emacs as described at the top of this page. PEL supports it for several major modes. • For this modes you can switch between the classic mode to the Tree-sitter based mode with the following command.		
Toggle between classic and Tree-Sitter major mode	**`<f11> C-t C-t`**	(**pel-treesit-toggle-mode**)	Toggle the major mode between the classic mode and the Tree-Sitter based mode. • If the other major mode is not available the command signals a user error. ☝Use the repeat command (bound to **`<f5>`** under PEL) to quickly toggle from the classic to the Tree-Sitter major mode and compare the impact of syntax highlighting.
Exploring Syntaxes	Use the following commands to explore the Tree-Sitter control syntax elements of a buffer using a tree-sitter based major mode.		
Toggle the tree-sitter explore minor mode.	**`<f11> C-t e`**	(**treesit-explore-mode** &optional ARG)	Enable exploring the current buffer's syntax tree. • Pops up a window showing the syntax tree of the source in the current buffer in real time. The corresponding node enclosing the text in the active region is highlighted in the explorer window. • This is a minor mode. If called interactively, toggle the 'Treesit-Explore mode' mode. If the prefix argument is positive, enable the mode, and if it is zero or negative, disable the mode. • If called from Lisp, toggle the mode if ARG is 'toggle'. Enable the mode if ARG is nil, omitted, or is a positive number. Disable the mode if ARG is a negative number.
Toggle tree-sitter inspection mode	**`<f11> C-t i`**	(**treesit-inspect-mode** &optional ARG)	Minor mode that displays in the mode-line the node which starts at point. • When this mode is enabled, the mode-line displays PARENT FIELD-NAME: (NODE FIELD-NAME: (CHILD (...))) where NODE, CHILD, etc, are nodes which begin at point. • PARENT is the parent of NODE. NODE is displayed in bold typeface. • FIELD-NAMEs are field names of NODE and CHILD, etc (see Info node '(elisp)Language Grammar', heading "Field names"). • If no node starts at point, i.e., point is in the middle of a node, then the mode line displays the earliest node that spans point, and its immediate parent. • This minor mode doesn't create parsers on its own. It uses the first parser in 'treesit-parser-list'. • This is a minor mode. If called interactively, toggle the 'Treesit-Inspect mode' mode. If the prefix argument is positive, enable the mode, and if it is zero or negative, disable the mode.

Operation	Keystroke	Function	Note
Tree-Sitter based packages	Some packages are using the tree-sitter parsing to implement features not specific to major modes. These packages are describe below. All of these packages require tree-sitter support. Therefore PEL support them on **Emacs >= 30** only.		
Combobulate 🚧	📦 The **combobulate** external package is used by PEL when the **pel-use-combobulate** user-option is set to t. • Since combobulate is still under early development it is not available on MELPA. PEL installs it via **quelpa** and installs **quelpa** first if it's not already present. **quelpa** creates a combobulate elpa-compliant package from combobulate Github git repo and stores it inside the ~/.emacs.d/elpa directory. • Once installed if you want to upgrade combobulate to what is now available in the combobulate Git repo, execute the **quelpa-upgrade combobulate** command		

Tree Sitter and Emacs— References

Topic & Link	Notes
Tree Sitter	• Tree-Sitter @ Wikipedia • Tree-Sitter @ home • Tree-Sitter @ Github

GNU Emacs Lisp Manual	• Using Tree Sitter Parser
	  • Tree-Sitter Language Grammar
	  • Retrieving Nodes
	  • Accessing Node Information
	  • Pattern Matching Tree-Sitter Nodes
	  • User-defined "Things" and Navigation
	  • Parsing text in multiple languages
	  • Developing major modes with Tree-Sitter
	  • Tree-Sitter C API Correspondence

Emacs and Tree-Sitter	• Emacs Tree-Sitter @ Github , a collection of tree-sitter Emacs projects

Articles about Emacs and Tree-Sitter	• Articles from Mickey Petersen:
	  • How to get Started with Tree-Sitter
	  • Tree Sitter and the Complications of Parsing Languages
	  • Let's Write a Tree-sitter Major Mode

Packages using Tree-Sitter	• **combobulate** @ Github
	• Articles from Mickey Petersen, combobulate author:
	  • Combobulate: Structured Movement and Editing with Tree-Sitter,
	  • Combobulate: Intuitive, Structured Navigation with Tree-Sitter,
	  • Combobulate: Editing and Searching with the new Query Builder,
	  • Combobulate: Interactive Node Editing with Tree-Sitter
	  • Combobulate: Bulk Editing Tree-Sitter Nodes with Multiple Cursors
	• Combobulate @ reddit

Emacs, PEL and Tree-sitter	• Using tree-sitter with Emacs and PEL describes how PEL currently deals with tree-sitter and how to setup the environment for using tree-sitter with Emacs.