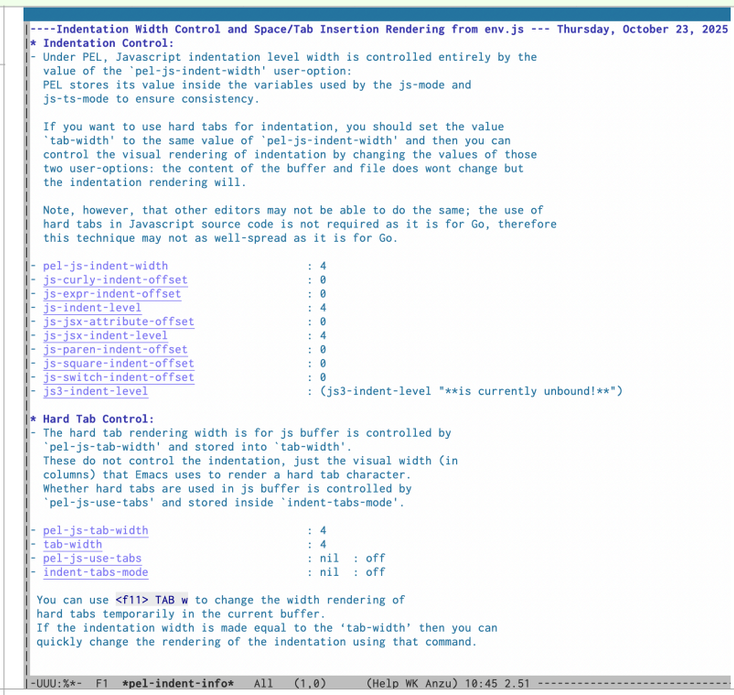
↖ Topic Index

Emacs Support for Javascript 🚧

Description	Keystroke	Function	Note
Javascript Support ○ **Help & Customization** • Major mode selection • Minor mode activation • Comments ○ Indentation Control • Navigation • Ouline/Fold/Show • Compilation/Static Analysis • Javascript REPL • Debugging mode ○ References Last updated on:	Emacs has built-in support for Javascript via the **js-mode** and **js-ts-mode** (when tree-sitter is activated). PEL supports Javascript when **pel-use-js** user options is turned on (non-nil), depending on its value: See Σ **Tree Sitter** and **Tree-sitter** • **t:** PEL activates the built-in classic j**s-mode** • **with-tree-sitter**: PEL activates the Tree-sitter based internal **js-ts-mode**. Requires: **Emacs >= 30**, and when **pel-use-tree-sitter** is set to t. • **js2-mode**: PEL activates the classic 📦 external **js2-mode** . It provides a **js2-minor-mode** that provides js2 features to the other modes. • **with-js2-minor**: PEL activates the built-in classic j**s-mode** and also activates the **js2-minor-mode** • **with-ts-js2-minor**: PEL activates the Tree-sitter based internal **js-ts-mode** and also activates the **js2-minor-mode.** ☝ Probably the most useful for most. • **js3-mode**: PEL activates the 📦 external **js3-mode**. A package that has not been maintained for some time. PEL also provides: • Σ **Speedbar** support for Javascript files listing functions and types when **pel-use-speedbar** is on. • Σ**iMenu** is supported by all javascript modes • 📦 **js-comint** support provided when **pel-use-js-comint** user-option is turned on (set to **t**). • 📦 **js2-closure** support provided when **pel-use-js2-closure** user-option is turned on (set to **t**). • 📦 **flow-js2-mode** support provided when **pel-use-flow-js2-mode** user-option is turned on (set to **t**). • 📦 **xref-js2** support provided when **pel-use-xref-js2** user-option is turned on (set to **t**). 🚧 Not yet integrated seamlessly in PEL. 2025-10-23		
Open this PDF file. See also: Σ **Help/Info**	`<f11> SPC i <f1>` `<f12> <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **PDF - Java** 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 Javascript support	`<f11> SPC i <f2>` `<f12> <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Javascript support: pel-use-javascript • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Σ **Customize** Emacs Javascript support	`<f11> SPC i <f3>` `<f12> <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs Java support: js, js2 • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Environment Help	Use the following command to verify your Javascript environment.		
Show PEL setup for Javascript	`<f11> SPC i ?` `<f12> ?`	(**pel-js-setup-info** &optional APPEND)	Display Javascript setup information inside a *pel-js-info* buffer with buttons providing quick access to the customization buffer of each variable shown. The information shown includes the value and interpretation of: • pel-use-js (whether the classic or tree-sitter based major mode is used), • activation of minor modes for the 3 set of major modes supported, • the user options controlling indentation and hard tab width rendering. To append information in the buffer instead of clearing the previous content type any prefix argument (such as **C-u**) before the command keystroke.
Toggle between classic-mode and tree-sitter based mode	PEL supports the ability to switch between a classic and the Tree-Sitter based mode. • For Javascript, the switch can be done between the **js-mode** and the **js-ts-mode**. There is no Tree-Sitter-based mode for the **js2-mode** or the **js3-mode**, • However, you can use **js-mode** and **js-ts-mode** with the **js2-minor-mode** which gives you the ability to use A Tree-Sitter based mode with the extra functionality provided by the **js2-mode** package. The selection of the major mode used by default is controlled by the **pel-use-js** user-option as described at the top of the page. • That selection can be changed without the following command without impacting the mode used when you open new JavaScript buffers.		
Toggle between classic and Tree-Sitter major mode See Σ **Tree Sitter**	`<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.
Minor Mode activation	PEL provides 3 user-options you can use to identify minor modes that are activated when the mode, selected by **pel-use-js**, starts. These user-options are: • **pel-js-activates-minor-modes** : identify the functions that activate minor modes in the js-mode and js-ts-mode buffers. • **pel-js3-activates-minor-modes** : identify the functions that activate minor modes in the js2-mode buffers (when **pel-use-js** is set to js2-mode). • **pel-js3-activates-minor-modes** : identify the functions that activate minor modes in the js3-mode buffers (when **pel-use-js** is set to js3-mode). It is also possible to manually activate or deactivate the minor mode. The following are minor modes related to the js2-mode functionality. Other minor modes can be quite useful for Javascript editing, such as the smart-dash mode, various modes for completion, search, cross reference, etc. Refer to other PEL PDF for more information on those. For example: • **Anzu**-mode See Σ **Search/Replace** • **smart-dash-mode** See Σ **Text Modes** • **which-function-mode** See Σ **Mode Line** • **flyspell-prog-mode** See Σ **Spell Checking**		
Toggle js2-minor-mode	`<f11> SPC i 2` `<f12> 2`	(**js2-minor-mode** &optional ARG)	Minor mode for running js2 as a background linter. • This allows you to use a different major mode for JavaScript editing, such as 'js-mode', while retaining the asynchronous error/warning highlighting features of 'js2-mode'. • This is a minor mode. If called interactively, toggle the 'Js2 minor mode' mode. • If the prefix argument is positive, enable the mode, if it is zero or negative, disable the mode. 📦 Requires external **js2-mode** . Activated by **pel-use-js**. See above.
Toggle editing Javascript with flow type annotation.	`<f11> SPC i M-f` `<f12> M-f`	(**flow-js2-mode** &optional ARG)	Toggle minor mode for editing JS files with **flow** type annotations. • If called interactively, toggle the 'Flow-Js2 mode' mode. • If the prefix argument is positive, enable the mode, if it is zero or negative, disable the mode.
Toggle Js2 refactor mode	`<f11> SPC i M-r` `<f12> M-r`	(**js2-refactor-mode** &optional ARG)	Toggle minor mode providing JavaScript refactorings. • If called interactively, toggle the 'Js2-Refactor mode' mode. • If the prefix argument is positive, enable the mode, if it is zero or negative, disable the mode.
Comments	Global comment commands can be used in markdown buffers to comment or un-comment lines and regions. See also: Σ **Comments** • Also see the outline commands provided by js2-mode below; the `C-c C-t` folds comment blocks.		
Insert, realign, comment/ uncomment region **With PEL**: Comment the current line with `M-0 M-;`	`M-;`	(**comment-dwim** ARG)	Insert or realign comment on current line (or region if a region is active). • On a single line, the comment is placed *after* the code. • `C-u M-;` executes comment-kill
		(**pel-comment-dwim** ARG)	Same as **comment-dwim** but comments the current line with a numeric ARG or 0.
Toggle display of comments in buffer or active region	`<f11> ; ;`	(**hide/show-comments-toggle** &optional START END)	Toggle hiding/showing of comments in the active region or whole buffer. • If the region is active then toggle in the region. Otherwise, in the whole buffer.

Description	Keystroke	Function	Note
Indentation Control	See also **Indentation** for more commands related to indentation and tab width control.		
Show Indentation settings Describe the user-options specific to the various Javascript major modes that control the indentation.	• `<f11> <tab> ?` • `<f11> ? <tab>`	(**pel-show-indent** &optional APPEND)	 This command opens a *pel-indent-info* buffer that describes what controls the indentation of Javascript code under the various major modes supported by PEL. It lists the customizable user-options that control various aspects of the indentation and describes that PEL sets some of the user-options from the one the PEL provides. If a user-option variable is not available because it is defined inside a major mode file that has not been loaded yet (because you don't use it), then the value shown states that it is unbound. The name of the user-options that are underlined are buttons that open the customization buffer providing more information and the ability to change its value in the current Emacs session and also for all future sessions. The buffer also describes the user-options that control the rendered width of hard tabs in Javascript buffers.
Indent current line (or region)	`<tab>`	(**indent-for-tab-command** &optional ARG)	Indent the current line or region, or insert a tab, as appropriate. • This is the indentation command that should be used. It can be issued from anywhere on the line and indents the line at the requited position.
Indent current line	`<f11> SPC i <tab>` `<f12> <tab>`	(**js2-indent-bounce** &optional BACKWARD)	Indent the current line, bouncing between several positions. • This is an older indentation command. You can issue it several time consecutively and it will reposition the line at several potential locations, but several of these locations might be invalid.
Print path of JSON value at point. Copy in kill ring.	`<f11> SPC i j` `<f12> j`	(**js2-print-json-path** &optional HARDCODED-ARRAY-INDEX)	Print the path to the JSON value under point, and save it in the kill ring. • If HARDCODED-ARRAY-INDEX provided, array index in JSON path is replaced with it.
Specialized Navigation	Aside from several generic navigation commands provided by Emacs and PEL (described in **Navigation**) the following commands exist. • See the navigation commands across functions.		
Skip comment and whiteface backward	`<f11> SPC i <left>` `<f12> <left>`	(**js2-backward-sws**)	Move backward through whitespace and comments. Available only when **pel-use-js** is set to js2-mode or a mode that activates js2-minor-mode .
Skip comment and whiteface forward	`<f11> SPC i <right>` `<f12> <right>`	(**js2-forward-sws**)	Move forward through whitespace and comments. Available only when **pel-use-js** is set to js2-mode or a mode that activates js2-minor-mode
Outline Commands	Aside from the generic outline commands provided by the outline-minor mode (see **Outline**) the **js2-mode** js2-mode and js2-minor-mode provide the following specialized outline commands:		
Show all text	`C-c C-a`	(**js2-mode-show-all**)	Show all of the text in the buffer.
Hide current element	`C-c C-e`	(**js2-mode-hide-element**)	Fold/hide contents of a block, showing ellipses.
Hide all functions	`C-c C-f`	(**js2-mode-toggle-hide-functions**)	Fold/hide the body of all functions in the buffer.
Hide/show element at point.	`C-c C-o`	(**js2-mode-toggle-element**)	Hide or show the foldable element at the point.
Show element at point	`C-c C-s`	(**js2-mode-show-element**)	Show the hidden element at current point.
Fold all comment blocks	`C-c C-t`	(**js2-mode-toggle-hide-comments**)	Folds all block comments in the buffer. • Use js2-mode-show-all to reveal them, or js2-mode-show-element to open an individual entry.
Compilation/Static Analysis	The following functions exist. More also exist depending on the major mode used. The documentation of this section is not complete. Will complete this once LSP support is documented.		
Toggle warning/error display	`C-c C-w`	(**js2-mode-toggle-warnings-and-errors**)	Toggle the display of warnings and errors. • Some users don't like having warnings/errors reported while they type.
	`C-c C-c`	(**js2-closure-fix**)	Fix the 'goog.require' statements in the current buffer. • This function assumes that all the requires are in one place and sorted, without indentation or blank lines. If you don't have any requires, they'll be added after your provide statements. If you don't have those, then this routine will fail. • Effort was also made to avoid needlessly modifying the buffer, since syntax coloring might take some time to kick back in. • This will automatically load 'js2-closure-provides-file' into memory if it was modified or not yet loaded.
Start Javascript REPL See also: **start Shells/REPLs**	`<f11> z r i` `<f12> z`	(**js-comint-repl** &optional CMD)	Start a NodeJS REPL process. • Optional CMD will override 'js-comint-program-command' and 'js-comint-program-arguments', as well as any nvm setting. • When called interactively use a universal prefix to set CMD. Requires the **js-comint** Emacs package and **node** installed. PEL activates this when **pel-use-js** is activated and **pel-use-js-comint** user option is set to **t**.
js2-mode debugging	The following functions are only available when the js2-mode or js2-minot mode is active and when the **js2-mode-dev-mode-p** variable is t. PEL activates this when the **pel-use-js** user-options is set to a value that activates them and when **pel-js2-activates-development-mode** is set to **t**.		
Show syntax name at point	`<f11> SPC i d .` `<f12> d .`	(**js2-node-name-at-point**)	Prints the name of the node detected by the js2-mode parsing. • Use this to debug the js2-mode logic. Also useful to identify the Javascript syntax elements.
Show the syntax node hierarchy at point	`<f11> SPC i d /` `<f12> d /`	(**js2-find-node-at-point**)	Print a list of syntax node elements, as used by the js2-mode code. • Only useful when investigating js2-mode code.

Javascript & Emacs Reference

Javascript Language	• Javascript @ Wikipedia • ECMAScript home . Has links to the standard.

JSX : Javascript XML	• JSX @ Wikipedia
	  • React @ Wikipedia
	  •