Emacs Support for Javascript

		Linacs Support			
Description	<u>Keystroke</u>	Function	<u>Note</u>		
Javascript Support O Help & Customization Major mode selection Minor mode activation Comments Indentation Control Navigation Ouline/Fold/Show Compilation/Static Analysis Javascript REPL Debugging mode References	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: PEL activates the built-in classic js-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 and also activates the js2-minor-mode that provides js2 features to the other modes. with-js2-minor: PEL activates the built-in classic js-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. PEL activates the external js3-mode. A package that has not been maintained for some time. PEL also provides: Speedbar support for Javascript files listing functions and types when pel-use-speedbar is on. SiMenu is supported by all javascript modes Speedbar support provided when pel-use-js-comint user-option is turned on (set to t). Speedbar support provided when pel-use-js2-closure user-option is turned on (set to t). Speedbar support provided when pel-use-js2-closure user-option is turned on (set to t).				
Last updated on:	2025-10-23				
Open this PDF file. See also: <u>Nelp/Info</u>	<f11> SPC i <f1></f1></f11>	(pel-help-pdf &optional OPEN- WEB-PAGE)	Open the <u>\$\mathbb{M}\ccirc\$ - Java</u> 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.		
	<f12> <f1></f1></f12>	WESTAGE			
Customize PEL Javascript support	<f11> SPC i <f2> <f12> <f2></f2></f12></f2></f11>	(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.		
∑ Customize Emacs	<f11> SPC i <f3></f3></f11>	(pel-customize-library	Customize Emacs Java support: js, js2		
Javascript support	<f12> <f3></f3></f12>	&optional OTHER-WINDOW)	• 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	<f11> SPC i ?</f11>	(pel-js-setup-info &optional	Display Javascript setup information inside a *pel-js-info* buffer with buttons providing quick		
Javascript	<f12> ?</f12>	APPEND)	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 <u>∑ Tree Sitter</u>	<f11> C-t C-t</f11>	(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.</f5>		
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 Search/Replace • flyspell-prog-mode See Spell Checking				
	• smart-dash-mode See <u>∑ Text Modes</u> • which-function-mode See <u>∑ Mode Line</u>				
Toggle js2-minor-mode	<f11> SPC i 2 <f12> 2</f12></f11>	(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</f12></f11>	(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</f12></f11>	(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: Comments Also see the outline commands provided by js2-mode below; the C-c C-t folds comment blocks.				
Insert, realign, comment/ uncomment region	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		
With PEL: Comment the current line with M-0 M-;		(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> ; ;</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	<u>Keystroke</u>	Function	Note		
Indentation Control	-	ore commands related to indentation			
Show Indentation	• <f11> <tab> ?</tab></f11>	(pel-show-indent &optional	Indentation Width Control and Space/Tab Insertion Rendering from env.js Thursday, October 23, 2025		
settings Describe the user-options specific to the various Javascript major modes that control the indentation.	<fi><fi1> <tab>?</tab></fi1></fi>		* Indentation Control: - Under PEL, Javascript indentation level width is controlled entirely by the value of the 'pel-js-indent-width' user-option: - PEL stores its value inside the variables used by the js-mode and js-ts-mode to ensure consistency. If you want to use hard tabs for indentation, you should set the value 'tab-width' to the same value of 'pel-js-indent-width' and then you can control the visual rendering of indentation by changing the values of those two user-options: the content of the buffer and file does wont change but the indentation rendering will. Note, however, that other editors may not be able to do the same; the use of hard tabs in Javascript source code is not required as it is for Go, therefore this technique may not as well-spread as it is for Go. - pel-js-indent-width		
			quickly change the rendering of the indentation using that command.		
Indent current line (or region)	<tab></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></tab></f12></tab></f11>	(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</f12></f11>	(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 <u>Navigation</u>) the following commands exist. • See the <u>navigation commands across functions</u> .				
Skip comment and whiteface backward	<f11> SPC i <left> <f12> <left></left></f12></left></f11>	(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></right></f12></right></f11>	(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 <u>Foutline</u>) the <u>js2-mode</u> 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	С-с С-е	(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	The following functions e	,			
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.		
ess.	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 tak 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	<f11> z r i</f11>	(js-comint-repl &optional CMD)	Start a NodeJS REPL process. Optional CMD will override 'js-comint-program-command' and 'js-comint-program-		
See also: <u>Satart Shells/REPLs</u>	<f12> z</f12>		 Optional CWID 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 <u>js-comint</u> Emacs package and <u>node</u> installed. PEL activates this when <u>pel-use-js</u> is activated and <u>pel-use-js-comint</u> user option is set to the pel-use-js-comint user optio		
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	<f11> SPC i d .</f11>	(js2-node-name-at-point)	Prints the name of the node detected by the js2-mode parsing.		
point	<f12> d .</f12>		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 /</f12></f11>	(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