Text Modes

lext Modes				
<u>Operation</u>	Keystroke	Function	<u>Note</u>	
Text Modes O Help & Customize Show state of text modes Snow state of text modes Toggle special text modes Smart-dash mode text whitespace modes text enriched mode Drawing in ASCII References	Emacs support navigation and also search for words and symbols, and the concept of "words" can be modified to include or exclude underscores and hyphens. It supports the following special mode: • superword-mode that treats words separated by hyphen and underscores as a single entity, useful for programming languages using snake case like C, C++, Erlang. • subword-mode that treats sections of camelCase and PascalCase as distinct words. That is useful when editing portions of these longer symbols. • As of Emacs 23.2 the CC Mode c-subword-mode is obsolete and has been replaced by the more general subword-mode. • glasses-mode that transforms the way unreadableSymbolsLikeThisOneUsingCamelCase popular in some programming languages into something that some people find more visually pleasing like unreadable_Symbol_like_This_One_Using_Camel_Case using underscores virtual separators. You can use other characters as separators through customization of the glasses customization group. • smart-dash-mode (see smart-dash-mode section below).			
	PEL provides the ability to activate these modes automatically for various major modes by identifying the major modes in the following user option lists: • pel-modes-activating-superword-mode • pel-modes-activating-subword-mode • pel-modes-activating-subword-mode pel-modes-activating-glasses-mode pel-modes-activating-glasses-m			
Last updated on:	2025-05-01		<u>▼ Whitespace</u> . For drawing, see also: <u>M Graphviz Dot, M PlantUML</u>	
Open this PDF file. See also: <u>∑ Help/Info</u>	<f11> t m <f1></f1></f11>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the <u>S</u> Text Modes 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.	
 ∑ Customize Text Mode support See also: ∑ Inserting Text ∑ Filling/Justification 	<f11> t m <f2></f2></f11>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL cross-reference support: what modes automatically activate superword-mode, subword-mode, auto-fill-mode and smart-dash-mode. Also control keypad special binding. • If OTHER-WINDOW is non-nil (use C-u), display in other window.	
∑ Customize Emacs text modes	• <f11> t <f3> • <f11> t m <f3></f3></f11></f3></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs text mode related groups: editing-basics, glasses, whitespace.	
∑ Customize Emacs whitespace management	<f11> t w <f3></f3></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs handling of whitespace.	
• Are hard tabs used for indentation, tab-width	<f11> t m ?</f11>	(pel-show-text-modes)	Display the state of the various text modes in the mini buffer. Specially useful to see if hard-tabs or spaces are inserted for indentation.	
 electric-quote-mode delete-selection-mode enriched-mode overwrite-mode 	 When indent-tabs-mode is active, Emacs inserts a number of hard tabs and spaces. The number of hard tabs instead depends on the amount of characters required for indentation and the tab-width. If indent-tabs-mode is not active, then Emacs inserts only space characters. PEL provides user-options of the form pel- Pel-provides user-options of the form pel- 			
 case folding subword, superword, glass modes visible-mode, smart-dash- 	-UU: F1 *scratch* All (4,0) (Lisp Interaction WK Anzu LY Fly/ 2 ElDoc Fill) 12:16 1.42 - Local newline does align: void . Automatically activated by modes (<f11> t a <f2>): (c-mode c++-mode sh-mode) - Local electric-quote-mode: off , electric-quote-local-mode: not loaded. - whitespace-mode : not loaded, show-trailing-whitespace: off , indicate-empty-lines: off. - enriched-mode : not loaded.</f2></f11>			
mode • paragraph definition See also: <u>Indentation</u>	- overwrite mode - case-fold-sear - subword mode - visible-mode - Sentences end - paragraph-star - paragraph-sepa	delete-selection-mode : off. sort-fold-case : not loaded. superword mode : on , glass-mode: not loaded. smart-dash-mode : not loaded.		
Switch Insert/Overwrite mode	<pre> <esc><insert> <esc><kp-0> </kp-0></esc></insert></esc></pre>	(overwrite-mode &optional ARG)	Toggles the overwrite mode on/off • With a prefix argument ARG, enable Overwrite mode if ARG is positive, and disable it otherwise.	
See also: Numkeypad	• <f11> t o • <f11> `</f11></f11>		 The <insert> key is not available in macOS keyboards where it registers as <kp-0> instead, but it's the same key on the numeric keypad: the numeric keypad 0 key.</kp-0></insert> In some situations, like when when using Emacs on a Linux host accessed through ssh, the <esc><kp-0> may not work. Try setting pel-keypad-esc-0-is-overwrite-mode to t.</kp-0></esc> 	
Binary file overwrite only mode. so nhext mode must be activated to edit in binary.	<f11> t 0</f11>	(nhexl-overwrite-only-mode &optional ARG)	Minor mode where text is only overwritten: Insertion/deletion is avoided where possible and replaced by overwriting existing text, if needed with 'nhexl-overwrite-clear-byte'. Requires the nhexl-mode package. PEL activates it if pel-use-nhexl user option is t.	
Make Info control text visible/invisible (toggle visible mode)	<f11> t m v</f11>	(visible-mode &optional ARG)	Toggle making all invisible text temporarily visible (Visible mode). With a prefix argument ARG, enable Visible mode if ARG is positive, and disable it otherwise.	
	 Useful for editing info files, where some characters are not visible by default. Same in Org-mode (for example to show everything, or to show the syntax of links without expanding anything). 			
Toggle subword-mode See also: ∑ Search/Replace	• <f11> t m b</f11>	(subword-mode &optional ARG)	Toggle subword-mode: a minor mode that treats sections of <u>camelCase</u> and <u>PascalCase</u> as distinct words. With a ARG > 0: enable subword-mode mode, ARG < 0: disable it.	
Toggle superword-mode	• <f12> M-b • M-<f12> M-b • <f11> t m p</f11></f12></f12>	(superword-mode &optional	 PEL provides the <f12> M-b key for the programming language modes where <u>camelCase</u> and <u>PascalCase</u> are popular.</f12> Toggle superword-mode: a minor mode that treats <u>snake case</u> as one word. In Lisp, '-' and '_' are 	
See also: Search/Replace	• <f12> M-p</f12>	ARG)	treated part of words. With a ARG > 0: enable superword mode, ARG < 0: disable it. • PEL provides the <f12> M-p key for the programming language modes where snake case is</f12>	
Toggle glasses-mode	• M- <f12> M-p • <f11> t m g</f11></f12>	(glasses-mode &optional	popular (Emacs Lisp, C, C++, Erlang, Python, etc) Minor mode for making identifiers likeThisLongCamelCaseName readable.	
See <u>Making CamelCase</u> Readable with Glasses- <u>Mode</u>	 With a prefix argument ARG, enable the mode if ARG is positive, and disable it otherwise. When glasses-mode is active, it tries to add virtual separators (like underscores) at places they belong to. It does not changes the content of the buffer, just the way the text looks. If you type the separator, it is inserted into the buffer. By default it uses underscore as separator. It can be modified through customization of the user-options in the glasses customization group. 			
Toggle sentence separators between 1 or 2 spaces	<f11> t m s</f11>	(pel-toggle-sentence-end)	Toggle definition of end of sentence between 2 and 1 space character (to help text filling). This has an impact on the commands that deal with sentences (navigation such as (backward-sentence) and (forward-sentence), kill such as (kill-sentence) and (backward-kill-sentence).	
Toggle local electric quote mode	<f11> t m '</f11>	(electric-quote-local-mode &optional ARG)	Toggle 'electric-quote-mode' only in this buffer. Useful to insert nicer-looking quote characters. • To toggle the mode inside all buffers use: M-x electric-quote-mode	
Toggle prettify-symbols-mode More info: here (on SO) and	<f11> t m y</f11>	(prettify-symbols-mode &optional ARG)	Toggle Prettify Symbols mode. • When Prettify Symbols mode and font-locking are enabled, symbols are prettified (displayed as composed characters) according to the rules in 'prettify-symbols-alist' (which see), which are locally defined by major modes supporting prettifying. To add further customizations for a given major	
here (on Reddit).			mode, you can modify 'prettify-symbols-alist'.	
Mark and Region See also: <u>© Cut & Paste</u> <u>© Marking</u>	behave like this; instead		on, the text in the selection is automatically replaced by the new text. By default Emacs does not is an active a marked region. If you want Emacs behave like other editors and automatically replace the	

<u>Operation</u>	Keystroke	Function	<u>Note</u>		
Smart Dash Mode	 Anyone that has been writing Lisp code for a while knows that using dash as word separator instead of underscore is more natural and faster to type. Unfortunately most programming languages (all non-Lisp?) have restrictions on the characters available in identifiers and underscore is often used. Typing underscore requires hitting the Shift key and it annoys some people that enjoyed writing Lisp code. This is where the smart-dash-mode helps. You can insert underscore in text by typing the dash key without hitting the Shift key! A very useful mode. More information is available in the <u>author's page</u>. Requires the <u>smart-dash</u> external package. PEL activates it when <u>pel-use-smart-dash</u> is set to t. To activate smart-dash-mode automatically: for major modes supported by PEL, add smart-dash-mode to the <u>pel-<mode>activates-minor-mode</mode></u> user-option for the specific mode. for other modes, add the mode name to the <u>pel-modes-activating-smart-dash-mode</u> user-option. 				
Toggle smart-dash mode	<f11> i -</f11>	(smart-dash-mode	Toggle the smart-dash-mode on/off.		
See also: • ∑ Numkeypad • ∑ Inserting Text • ∑ Mode Line	all_lowercase_c_ident While Smart-Dash mode character. You might If Smart-Dash mode is also activate Smart-D decrement aren't mad that this will necessita Normally when smar However, with PEL, tt In Numlock OFF: with no marked with an area man with area marked In Numlock ON: with no marked with area marked with area marked with area marked	tifiers as comfortably as you woode is active, you can type C-q need to do this if you want to ty is activated while in a C-like more difficult by Smart-Dash ate that you type literal undersont-dash-mode is active the numerical he behaviour of the keypad '-' is area: inser rked: kill mid with er/expand-region: kill mid area: inser dinormally: lignor	— or use the minus key on the numeric keypad to override it and insert a dash after a C-style identifier type a cramped-looking expression like x-5. de (c-mode, c++-mode, and objc-mode by default, customizable with 'smart-dash-c-modes') it will "_>" into "->" and "" into "" automatically so that struct pointer member access and postfix-remode's tendency to insert underscores at the tail ends of identifiers whether you want it to or not. Note ores if you want more than one underscore in a row. eric dash key (<kp-subtract>) acts as a smart-dash only. sonly partly affected when the smart-dash-mode is active and it depends on the Numlock state: rt a dash. Numeric argument for multiple insertion is not supported.</kp-subtract>		
	For more information on the NumLock control and key support, see <u>See Numkeypad</u> .				
	with PEL type <f11> t m? to display the status of text modes including dash-mode. With PEL, when pel-use-delight is turned on, a short lighter of a green dash is showing in the mode line when smart-dash-mode is active.</f11>				
Text Whitespace Modes	The following Emacs command control how whitespace is shown or hidden. The following commands are also described in the Whitespace table. The whitespace mode can be used to highlight: space characters, hard tabs, end of line and lines that are too long.				
Toggle Whitespace Mode	• <f11> t m w</f11>	(whitespace-mode	Toggle whitespace visualization (Whitespace mode). ARG>0: enable, ARG<0: disable.		
See also: <u>▼ Whitespace</u>	• <f11> t w m</f11>	&optional ARG)	The kind of whitespace visualized is determined by the list variable whitespace-style , whitespace-newline .		
Hide/Show trailing whitespaces	<f11> t w T</f11>	(pel-toggle-show-trailing- whitespace)	Toggle highlight of the trailing whitespaces in current buffer. • Toggles the value of the variable show-trailing-whitespace .		
Hide/Show trailing empty lines	<f11> t w e</f11>	(pel-toggle-indicate-empty- lines)	Toggle highlight of empty lines. • Toggles the value of the variable <i>indicate-empty-lines</i> .		
Toggle individual elements of whitespace-style	<f11> t w o</f11>	(whitespace-toggle- options ARG)	If local whitespace-mode is off, toggle the ARG option and turn on local whitespace-mode. If local whitespace-mode is on, toggle ARG option and restart local whitespace-mode.		
See also: Whitespace	The argument, which is a single character and must be typed following the <f11> t w o, can be:</f11>				
Enriched Mode See also: •	The enriched-mode is a buffer-local minor mode. When active text properties are used to modify the way the buffer renders text. • When saving a buffer with enriched-mode to a file, the is saved in 'text/enriched' format. Several text properties are saved to the file, including the fonts, colours, indentation and justification. If Emacs opens the file later it can automatically re-active the enriched-mode to display these properties. • If Emacs fails to recognize that the file is in 'text/enriched' format, execute M-x format-decode-buffer. • This can be quite useful to store the content of a log file that uses different fonts and colours to indicate commands and errors in a file. Later you can re-open the file with Emacs and see the same fonts and colours!				
Toggle enriched text mode	• <f11> t e e • <f11> t m e</f11></f11>	(enriched-mode &optional ARG)	Toggle the enriched-text minor mode for editing text/enriched files. These are files with embedded formatting information in MIME standard text/enriched format. With a prefix argument ARG, enable the mode if ARG is positive, and disable it otherwise. If called from Lisp, enable the mode if ARG is omitted or nil. Turning the mode on or off runs 'enriched-mode-hook'.		
GNU Screen Log File	GNU Screen log files co	ntain ANSI escape codes norm	ally interpreted by the shell but not by Emacs. Use the following command to render the codes.		
Fix the rendering of the log file created by GNU Screen See also:	<f11> t s</f11>	(pel-screen-log-fix- rendering)	 Fix rendering of buffer created by the GNU Screen log. It converts the marked area of a buffer, if it is marked, otherwise it processes the entire or the narrowed portion of the buffer. It renders the escape codes, converts the line endings to Unix-style line endings. In some cases the log file created by GNU Screen injects one extra line per line. If the 'pel-screen-log-delete-all-consecutive-blank-lines' user-option is set, the function removes them. Some artifact main remain after these transformations. To fix them automatically, identify a set of regular expression/replacement string pairs in the 'pel-screen-log-fix-regexp-pairs' user-option. ⚠ This removes the escape codes from the buffer but renders them using color and other attributes. If you save the file these will no longer be visible unless you first activates the enriched text mode in the buffer to encode the text attributes in a way Emacs will be able to interpret later. 		
Drawing ASCII in Emacs See also: M Graphviz Dot M PlantUML	Emacs provides the picture-mode and artist-mode to draw ASCII-based pictures. Both are available when Emacs runs in graphics and terminal mode. However, I have not been able to use artist-mode with the mouse, even with xterm-mouse-mode active: each click just prints an ANSI sequence code. Two modes are available: 1) Artist Mode 2) Picture Mode The Picture mode can be very useful to type text in vertical fashion when for example, writing reStructuredText table or writing code in tabular fashion.				
Artist Mode	Although you can get some commands to work in terminal mode, it's best to use artist-mode when running Emacs in graphics mode. See <u>demo</u> .				
Toggle artist mode See also: <u>▼ Drawing</u>	<f11> D a</f11>	(artist-mode &optional ARG)	Toggle Artist mode. With argument ARG, turn Artist mode on if ARG is positive. Artist lets you draw lines, squares, rectangles and poly-lines, ellipses and circles with your mouse and/or keyboard.		
Picture Mode	Emacs supports the picture mode that allow you to move your cursor freely anywhere inside the window, which greatly simplify creating rectangular shapes for tables or even <i>drawing</i> ASCII-art. This work well in both graphics and terminal mode.				
Enter picture mode See also: Drawing	• <f11> D p • <f11> t p</f11></f11>	(picture-mode)	Switch to Picture mode, in which a quarter-plane screen model is used. Very useful to type text in vertical fashion when for example, writing reStructuredText table.		

Text Modes — References

Topic & Link	Notes
GNU Emacs Manual: Text - Words	
GNU Emacs Manual: Text - Sentences	
GNU Emacs Manual: Text - Paragraphs	
GNU Emacs Manual: Text - Quotation Marks	
GNU Emacs Manual: Modes - Minor Modes	
GNU Emacs Manual: Programs - Other Features Useful for Editing Programs	
GNU Info Manual: Getting Started - Invisible text in Emacs Info	