Text Modes

Operation	Keystroke	Function	Note
Text Modes See also: • ∑ **Search/Replace** • ∑ **Whitespace**	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 • As of Emacs 23.2 the CC Mode c-subword-mode is obsolete and has been replaced by the more general subword-mode.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-smart-dash-mode** Emacs also has: • several modes that deal with whitespace, • drawing modes that allow you to draw in text and navigate over 'void' space • different way of handling the concept of marking text in a buffer. The commands to control all those are listed in this table, starting with the commands to get help in Emacs and customize these features.		
Open this PDF file. See also: ∑ **Help/Info**	`<f11> t m <f1>`	**(pel-help-pdf** &optional OPEN-WEB-PAGE**)**	Open the ∑ **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>`	**(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. • If OTHER-WINDOW is non-nil (use `C-u`), display in other window.
∑ Customize Emacs text modes	`<f11> t m <f3>`	**(pel-customize-library** &optional OTHER-WINDOW**)**	Customize Emacs text mode group: **glasses**
Switch Insert/Overwrite mode	• `<insert>` • `<Esc><kp-0>` • `<f11> t o` • `<f11> ``	**(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. • ☞The `<insert>` key is not available in macOS keyboards. • For keyboards with numerical keyboards the `<Esc><kp-0>` is quite close since `<kp-0>` is often used as the insert key.
Binary file overwrite only mode	`<f11> t O`	**(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'. 👆 The nhexl mode must first be activated to edit the file in binary mode. ☞📦 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`	**(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 developing 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` • `<f12> M-b` • `<M-f12> M-b`	**(subword-mode** &optional ARG**)**	Toggle subword-mode: a minor mode that treats sections of camelCase and PascalCase as distinct words. • With a ARG>0: enable subword-mode mode , ARG<0 : disable it. • PEL provides the `<f12> M-b` key for the programming language modes where camelCase and PascalCase are popular.
Toggle superword-mode See also: ∑ **Search/Replace**	• `<f11> t m p` • `<f12> M-p` • `<M-f12> M-p`	**(superword-mode** &optional ARG**)**	Toggle superword-mode: a minor mode that treats snake_case as one word. In Lisp, '-' and '_' are 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 popular (Emacs Lisp, C, C++, Erlang, Python, etc…)

Toggle glasses-mode
 	
• See Making CamelCase Readable with Glasses-Mode
	• `<f11> t m g`	**(glasses-mode** &optional ARG	Minor mode for making identifiers likeThisLongCamelCaseName readable.
			• 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`	**(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 '`	**(electric-quote-local-mode** &optional ARG**)**	Toggle 'electric-quote-mode' only in this buffer. Useful to insert nicer-looking quote characters.
Show state of text modes	`<f11> t m ?`	**(pel-show-text-modes)**	Display the state of the various text modes in the mini buffer.
Mark and Region See also: ∑ **Cut & Paste** ∑ **Marking**	With most editors when you type over a "selected" region, the text in the selection is automatically replaced by the new text. By default Emacs does not behave like this; instead it allows typing text while there is an active a marked region. If you want Emacs behave like other editors and automatically replace the text activate the "*delete-selection-mode*" with the following command.		
Toggles delete selection mode	`<f11> t m d`	**(delete-selection-mode)**	Toggles delete selection-mode on/off. In delete selection-mode typing a character while a region is active replaces the entire region with what is typed. By default delete selection-mode is off.

Operation	Keystroke	Function	Note
Smart Dash Mode		📦 This uses the **smart-dash** external package. 🔲 PEL activates it when **pel-use-smart-dash** is set to **t**. 🔲 The **pel-modes-activating-smart-dash-mode** user option identifies the major modes where PEL automatically activates the 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 **author's page**.		
Toggle smart dash mode See also: • ∑ 🖩 **Numkeypad** • ∑ **Inserting Text**	**<f11> i -**	(**smart-dash-mode** &optional ARG)	Toggle the smart-dash-mode on/off.
		• When smart-dash-mode is active, it redefines the dash key to insert an underscore within C-style identifiers and a dash otherwise. This allows you to type all_lowercase_c_identifiers as comfortably as you would lisp-style-identifiers. • While Smart-Dash mode is active, you can type **C-q -** or use the minus key on the numeric keypad to override it and insert a dash after a C-style identifier character. You might need to do this if you want to type a cramped-looking expression like x-5. • If Smart-Dash mode is activated while in a C-like mode (c-mode, c++-mode, and objc-mode by default, customizable with '**smart-dash-c-modes**') it will also activate Smart-Dash-C mode, which translates "**_>**" into "**->**" and "**__**" into "**--**" automatically so that struct pointer member access and postfix-decrement aren't made more difficult by Smart-Dash mode's tendency to insert underscores at the tail ends of identifiers whether you want it to or not. Note that this will necessitate that you type literal underscores if you want more than one underscore in a row. 👆 With PEL, the keypad '-' is not affected, allowing the insertion of dash character as long as Emacs is operating in numlock ON: • in numlock ON mode the keypad '-' inserts a dash character, • in numlock OFF it kills the current line. For more information, see ∑ 🖩 **Numkeypad**. 👆 Also note that as soon as dash character is before point typing '-' from any key will produce a dash character. 👆 With PEL type **<f11> t m ?** to display the status of text modes including dash-mode.		
Text Whitespace Modes		The following Emacs command control how whitespace is shown or hidden. The following commands are also described in the ∑ **Whitespace** table.	
Toggle Whitespace Mode See also: ∑ **Whitespace**	• **<f11> t m w** • **<f11> t w m**	(**whitespace-mode** &optional ARG)	Toggle whitespace visualization (Whitespace mode). ARG>0: enable, ARG<0: disable. The kind of whitespace visualized is determined by the list variable **whitespace-style**, **whitespace-newline**.
Hide/Show trailing whitespaces	**<f11> t w T**	(**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**	(**pel-toggle-indicate-empty-lines**)	Toggle highlight of empty lines. • Toggles the value of the variable **indicate-empty-lines**.
Toggle individual elements of whitespace-style See also: ∑ **Whitespace**	**<f11> t w o**	(**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.
		The argument, which is a single character and must be typed following the **<f11> t w o**, can be: • **f** toggle face visualization • **t** toggle TAB visualization • **s** toggle SPACE and HARD SPACE visualization • **r** toggle trailing blanks visualization • **l** toggle "long lines" visualization • **L** toggle "long lines" tail visualization • **n** toggle NEWLINE visualization • **e** toggle empty line at bob and/or eob visualization • **C-i** toggle indentation SPACEs visualization (via 'indent-tabs-mode') • **I** toggle indentation SPACEs visualization • **i** toggle indentation TABs visualization • **C-t** toggle big indentation visualization • **C-a** toggle SPACEs after TAB visualization (via 'indent-tabs-mode') • **A** toggle SPACEs after TAB: SPACEs visualization • **a** toggle SPACEs after TAB: TABs visualization • **C-b** toggle SPACEs before TAB visualization (via 'indent-tabs-mode') • **B** toggle SPACEs before TAB: SPACEs visualization • **b** toggle SPACEs before TAB: TABs visualization • **?** Show the above list of options.		
Drawing ASCII in Emacs		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: • Artist Mode • 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.	
Toggle artist mode See also: ∑ **Drawing**	**<f11> D a**	(**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 *drawing* ASCII-art. This work well in both graphics and terminal mode.	
Enter picture mode See also: ∑ **Drawing**	• **<f11> D p** • **<f11> t p**	(**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. • Type **C-c C-c** to exit picture-mode and return to the mode previously used in the buffer.

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	