Text Modes

| <u>Operation</u> | Keystroke | Function | Note | | |
|---|---|---|---|--|--|
| Text Modes | - | | and symbols, and the concept of "words" can be modified to include or exclude underscores and | | |
| See also: | hyphens. It supports | the following special mode: | by hyphen and underscores as a single entity, useful for programming languages using snake case | | |
| ∑ Search/Replace ∑ Whitespace | * subeword-mode that treats words separated by hypiter and direct scores as a single entity, useful for programming languages using shake 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 As of Emacs 23.2 the CC Mode c-subword-mode is obsolete and has been replaced by the more general subword-mode.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_SymbolsLikeThis_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, * 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: <u>▼ Help/Info</u> | <f11> t m <f1></f1></f11> | (pel-help-pdf &optional OPEN-WEB-PAGE) | Open the <u>Nature</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. • If OTHER-WINDOW is non-nil (use C-u), display in other window. | | |
| <u>∑ Customize</u> Emacs text modes | <f11> t <f3></f3></f11> | (pel-customize-library &optional OTHER-WINDOW) | Customize Emacs text mode related groups: glasses, whitespace. | | |
| <u>Customize</u> Emacs text modes | <f11> t m <f3></f3></f11> | (pel-customize-library &optional OTHER-WINDOW) | Customize Emacs text mode group: glasses | | |
| <u>© Customize</u> Emacs whitespace management control | <f11> t w <f3></f3></f11> | (pel-customize-library &optional OTHER-WINDOW) | Customize Emacs handling of whitespace . | | |
| Switch Insert/Overwrite mode | <pre>• <esc><insert> • <esc><kp-0> • <f11> t o • <f11> `</f11></f11></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. • 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> | | |
| Binary file overwrite only mode | <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'. 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</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 developing info files, where some characters are not visible by default. Same in Orgmode (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. | | |
| | • <f12> M-b • <m-f12> M-b</m-f12></f12> | | 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 See also: Search/Replace | • <f11> t m p</f11> | (superword-mode &optional ARG) | Toggle superword-mode: a minor mode that treats <u>snake_case</u> as one word. In Lisp, '-' and '_' are treated part of words. • With a ARG>0: enable superword mode, ARG<0: disable it. | | |
| | • <f12> M-p • <m-f12> M-p</m-f12></f12> | - | PEL provides the <f12> M-p key for the programming language modes where snake case is popular (Emacs Lisp, C, C++, Erlang, Python, etc)</f12> | | |
| See Making CamelCase Readable with Glasses- Mode | • <f11> t m g</f11> | (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</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 (backwardsentence) 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. | | |
| Show state of text modes whether hard tabs are used for indentation, tabwidth electric-quote-mode delete-selection-mode enriched-mode overwrite-mode case folding subword, superword, glass modes visible-mode, smart-dashmode paragraph definition See also: Indentation | tabs or spaces are ins When indent-tabs- a number of hard ta The number of he the amount of ch indentation and t If indent-tabs-mode inserts only space of PEL provides user-of mode>-use-tabs | (pel-show-text-modes) pecially useful to see if hard- erted for indentation: mode is active, Emacs inserts ibs and spaces. ard tabs instead depends on iaracters required for the tab-width. e is not active, then Emacs | Display the state of the various text modes in the mini buffer. -UU-:Fl keymaps.c Top (1,0) Git-master (C/*la WK Anzu Modes status: - Local indent-tabs-mode : off: use spaces. Tab width = 8 - Local electric-quote-mode: off. - delete-selection-mode : off. - enriched-mode : not loaded. - overwrite mode : off. - case-fold-search: on, sort-fold-case: not loaded. - subword mode: off, superword mode: on, glass-mode: not loaded. - visible-mode: off, smart-dash-mode: on. - Sentences end with 2 space characters. - paragraph-start : "[*\\(//+\\\\\\\\\\) []*\$\\\^^L" paragraph-separate: "[*\\(//+\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ | | |

| <u>Operation</u> | <u>Keystroke</u> | Function | <u>Note</u> | | |
|---|---|--|--|--|--|
| Mark and Region See also: <u>∑ Cut & Paste</u> <u>∑ Marking</u> | behave like this; instea | | gion, the text in the selection is automatically replaced by the new text. By default Emacs does not ere is an active a marked region. If you want Emacs behave like other editors and automatically with the following command. | | |
| Toggles delete selection mode | <f11> t m d</f11> | (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. | | |
| 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. Requires the smart-dash external package. PEL activates it when pel-use-smart-dash is set to t. To activate smart-dash-mode automatically: for major modes supported by PEL, add smart-dash-mode to the pel- MODE>activates-minor-mode user-option for the specific mode. for other modes, add the mode name to the pel-modes-activating-smart-dash-mode user-option. | | | | |
| Toggle smart-dash mode | <f11> i -</f11> | (smart-dash-mode &optional ARG) | Toggle the smart-dash-mode on/off. | | |
| See also: • | 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. A Normally when smart-dash-mode is active the numeric dash key (<kp-subtract>) acts as a smart-dash only.</kp-subtract> However, with PEL, the behaviour of the keypad '-' is only partly affected when the smart-dash-mode is active and it depends on the Numlock state: insert a dash. Numeric argument for multiple insertion is not supported. kill marked area insert a dash. Numeric argument for multiple insertion is not supported. kill marked area with area marked with er/expand-region: Normally in Numlock OR: with area marked ormally: insert an underscore after letter, number or underscore, dash otherwise lignore the marked area; insert a dash at point with area marked ormally: with area marked with er/expand-region: Reduces the marked area semantically as controlled by er/expand-region For more information on the Numlock control and key support, see \(\sum \) Numkeypad. With PEL, when pel-use-delight is turned on, a short lighter of a green da | | | | |
| 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, lines that are too long. | | | | |
| Toggle Whitespace Mode | • <f11> t m w • <f11> t w m</f11></f11> | (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 <i>whitespace-style</i> , | | |
| See also: Whitespace Hide/Show trailing whitespaces | <f11> t w T</f11> | (pel-toggle-show-trailing- whitespace) | whitespace-newline. 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: • f toggle face visualization • t toggle TAB visualization • s toggle SPACE and HARD SPACE visualization • r toggle trailing blanks visualization • I toggle "long lines" visualization • I 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 • C-a toggle SPACEs after TAB visualization • C-b toggle SPACEs after TAB: SPACEs visualization • a toggle SPACEs after TAB: TABs visualization • a toggle SPACEs before TAB visualization • b toggle SPACEs before TAB visualization • b toggle SPACEs before TAB: TABs visualization • C-b toggle SPACEs before TAB: TABs visualization • B toggle SPACEs before TAB: TABs visualization • C-b toggle SPACEs before TAB: TABs visualization • B toggle SPACEs before TAB: TABs visualization • C-b toggle SPACEs before TAB: TABs visualization • B toggle SPACEs before TAB: TABs visualization</f11> | | | |
| Enriched Mode See also: • | 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. | | | | |
| Toggle enriched text mode | Later you can • <f11> t e e</f11> | re-open the file with Emacs an (enriched-mode & optional | d see the same fonts and colours! Toggle the enriched-text minor mode for editing text/enriched files. | | |
| - gg.c surrous text mode | • <f11> t m e</f11> | ARG) | These are files with embedded formatting information in the 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'. | | |
| 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: <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. | | |

| <u>Operation</u> | <u>Keystroke</u> | Function | <u>Note</u> |
|--------------------------|--|----------------|---|
| 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 | • <f11> D p</f11> | (picture-mode) | Switch to Picture mode, in which a quarter-plane screen model is used. |
| See also: Drawing | • <f11> t p</f11> | | 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 | |