










Operation	Keystroke	Function	Note
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 “ <i>delete-selection-mode</i> ” 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.
Smart Dash Mode	<ul style="list-style-type: none"> 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. <div>  Requires the smart-dash external package.  PEL activates it when pel-use-smart-dash is set to t. </div> <div>  To activate smart-dash-mode automatically: <ul style="list-style-type: none"> 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. </div>		
Toggle smart-dash mode See also: <ul style="list-style-type: none"> ☞ Num keypad ☞ Inserting Text ☞ Mode Line 	<f11> i -	(smart-dash-mode &optional ARG)	Toggle the smart-dash-mode on/off. <ul style="list-style-type: none"> 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. <div>  Normally when smart-dash-mode is active the numeric dash key (<kp-subtract>) acts as a smart-dash only. </div> <div>  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: <ul style="list-style-type: none"> In Numlock OFF: <ul style="list-style-type: none"> with no marked area: insert a dash. Numeric argument for multiple insertion is not supported. with an area marked: kill marked area with area marked with er/expand-region: kill marked area In Numlock ON: <ul style="list-style-type: none"> with no marked area: insert an underscore after letter, number or underscore, dash otherwise with area marked normally: Ignore the marked area; insert a dash at point 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 ☞ Num keypad. </div> <div>  With PEL type <f11> t m ? to display the status of text modes including dash-mode. </div> <div>  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. </div>
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 See also: ☞ Whitespace	<ul style="list-style-type: none"> <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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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) The argument, which is a single character and must be typed following the <f11> t w o , can be: <ul style="list-style-type: none"> 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. 	<ul style="list-style-type: none"> 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.
Enriched Mode See also: <ul style="list-style-type: none"> ☞ Enriched Text ☞ Filling/Justification Use it to store coloured logs!	enriched-mode is a buffer-local minor mode. When active text properties are used to modify the way the buffer renders text. <ul style="list-style-type: none"> 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. <div>  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! </div>		
Toggle enriched text mode	<ul style="list-style-type: none"> <f11> t e e <f11> t m e 	(enriched-mode &optional ARG)	Toggle the enriched-text minor mode for editing text/enriched files. <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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.

Operation	Keystroke	Function	Note
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: Σ Drawing	<ul style="list-style-type: none"> <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. <ul style="list-style-type: none"> 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	