# EMACS Cheatsheet

**For version 25**

## Command line options

`--visit=<filespec> | --file=<filespec> | <filespec>`
Open filespec into individual buffers for editing.

`+row[:column]`
Move point to line number row and (optional) horizontal position column in the file (default is +1:1).

`--insert <file>`
Insert file at the beginning of the buffer.

`--load <file> | -l <file>`
Execute the Emacs Lisp instructions in file.

`--g <dimensions> | --geometry <dimensions>`
Set the window's width, height, and position according to the given X window dimensions (the default is to make the window 80x40 characters).

`-nw | --no-windows`
In X, don't use an X client window, but open in the current terminal window instead. This option doesn't affect console sessions.

## Prefixes

`C-c`
Commands particular to the current editing mode

`C-x`
Commands for files and buffers

`C-h`
Help commands

`M-x`
Literal function name

`M-!` *shell-command*
Execute external shell command from within Emacs.The output from the shell command is displayed in the minibuffer or in a separate buffer, depending on the output size. When used with a prefix argument (e.g, `C-u M-!`), the shell-command output is inserted in the current buffer at point.

`M-|` *shell-command-on-region*
Provide the region text to the shell command as input. If you want the shell to replace the region text with the output from the shell command, use `C-u M-|`.

## Buffer and files functions

`C-x C-f` *find-file*
Visit a file.

`C-x C-s` *save-buffer*
Save current buffer to disk.

`C-x C-w` *write-file*
Ask for a filename and write the current buffer with that name. Like the command "Save as ..." on other editors.

`C-x s` *save-some-buffers*
Ask about saving all unsaved buffers to disk.

`C-x C-c` *save-buffers-kill-emacs*
Ask about saving all unsaved buffers to disk and exit Emacs.

`C-x C-z | C-z` *suspend-emacs*
Suspend Emacs and make it a background process (press `fg | fg %emacs` to *awake* it back.

`C-x C-b` *list-buffers*
List all buffers.

`C-x k` *kill-buffer*
Kill a buffer (the current buffer, by default).

`C-x C-q` *vc-toggle-read-only*
Toggle read-only status on the current buffer (and perform version control if applicable).

`C-x i` *insert-file*
Insert the contents of a file at point.

`C-x z` *repeat*
Repeat most recently executed command.

`C-x M-: | C-x M-ESC` *repeat-complex-command*
Edit and re-evaluate last complex command. A complex command is one which used the minibuffer. The command is placed in the minibuffer as a Lisp form for editing. The result is executed, repeating the command as changed.

## Movement and navigation

`C-p | UpArrow` *previous-line*
Move point up to the previous line.

`C-n | DownArrow` *next-line*
Move point down to the next line.

`C-f | RightArrow` *forward-char*
Move point forward to the next character.

`C-b | LeftArrow` *back-char*
Move point backward to the previous character.

`M-f | C-RightArrow` *forward-word*
Move point forward to the next word.

`M-b | C-LeftArrow` *backward-word*
Move point backward to the previous word.

`C-v | PgDn` *scroll-up*
Scroll the text upward by a screen.

`M-v | PgUp` *scroll-down*
Scroll the text downward by a screen.

`C-Home` *beginning-of-buffer*
Move point to the beginning of the buffer. (On some versions, this key is defined by default to move to the beginning of the current line.)

`C-End` *end-of-buffer*
Move point to the end of the buffer. (On some versions, this key is defined by default to move to the end of the current line.)

`Home | C-a` *beginning-of-line*
Move point to the beginning of the line.

`End | C-e` *end-of-line*
Move point to the end of the line.

`M-a` *beginning-of-sentence*
Move point to the beginning of the sentence.

`M-e` *end-of-sentence*
Move point to the end of the sentence.

`C-{` *beginning-of-paragraph*
Move point to the beginning of the paragraph.

`C-}` *end-of-paragraph*
Move point to the end of the paragraph.

## Navigating over balanced expressions

`C-M-n` *forward-list*
Move forward over a parenthetical group.

`C-M-p` *backward-list*

Move backward over a parenthetical group.

`C-M-f` *forward-sexp*
Move forward over a balanced expression.

`C-M-b` *backward-sexp*
Move backward over a balanced expression.

`C-M-k` *kill-sexp*
Kill balanced expression forward.

`C-M-SPC` *mark-sexp*
Put the mark at the end of the sexp.

### Common editing commands

`Ins` *overwrite-mode*
Toggle overwrite mode (default is off).

`Backspace | Del` *delete-backward-char*
Delete the character before point.

`C-d` *delete-char*
Delete the character at point.

`M-d` *kill-word*
Delete the characters from point forward to the end of the word.

`M-Backspace | M-Del` *backward-kill-word*
Delete the characters from point backward to the beginning of the word.

`C-_` *undo*
Undo your last typing or action.

`C-q[CHAR] or [NUM]` *quoted-insert*
Insert, at point, the literal character keypress or the character whose octal value is XXX.

`C-u[NUM] [COMMAND]` *universal-argument*
Execute command a total of number (default 4) times in succession.

### Functions for marking and killing text

`C-Space` *set-mark-command*
Set the mark at point.

`C-k` *kill-line*
Kill all text from point to the end of the line.

`C-w` *kill-region*
Kill the region.

`M-w` *kill-ring-save*
Save the region in the kill ring, but don't kill it.

`C-y` *yank*
Yank text from the kill ring.

### Commands for using rectangles

`C-space` *set-mark-command*
Marks one corner of a rectangle (point marks the opposite corner).

`C-x r k` *kill-rectangle*
Kills the current rectangle and saves it in a special rectangle buffer.

`C-x r d` *delete-rectangle*
Deletes the current rectangle and doesn't save it for yanking.

`C-x r c` *clear-rectangle*
Clears the current rectangle, replacing the entire area with whitespace.

`C-x r o` *open-rectangle*
Opens the current rectangle, filling the entire area with whitespace and moving all text from the rectangle to the right.

`C-x r y` *yank-rectangle*
Yanks the contents of the last-killed rectangle at point, moving all existing text to the right.

### Advanced mark and selection commands

`C-u C-space` *pop-to-mark-command*
Moves to the previous mark in the mark ring.

`C-x C-x` *exchange-point-and-mark*
Swaps the location of point and the mark.

`M-@` *mark-word*
Marks all text from point to the end of the current word.

`M-h` *mark-paragraph*
Marks the current paragraph, regardless of the location of point.

*transient-mark-mode*
Toggles Transient Mark mode.

`C-x h` *mark-whole-buffer*
Marks the entire buffer, regardless of the location of point.

### Advanced kill and yank commands

`[NUM] C-k` *kill-line*
Kills integer number of lines. If 0, kills from point to the beginning of the line; if negative, kills in reverse (not sure for version 25).

`M-k` *kill-sentence*
Kills from point to the end of the sentence.

`M-z` *zap-to-char*
Zaps all text from point to the specified character.

`M-y` *yank-pop*
Moves to the next slot in the kill ring.

`[NUM] C-y` *yank*
Yanks the specified slot in the kill ring.

### Text mode key bindings

`Esc`
Prefix for mode-specific commands

`Esc Tab | M-Tab` *ispell-complete-word*

`Esc S | M-S` *center-paragraph*

`Esc s | M-s` *center-line*

### Text manipulation commands

`C-x C-i | C-x Tab` *indent-rigidly*
This command indents lines in the region (or at point).

*fill-region*
This command fills all paragraphs in the region.

`M-q` *fill-paragraph*
This command fills the single paragraph at point.

`M-\` *delete-horizontal-space*
This command removes any horizontal space to the right and left of point.

`C-o` *open-line*
This command opens a new line of vertical space below point, without moving point.

`C-t` *transpose-chars*
This command transposes the single characters to the right and left of point.

`M-t` *transpose-words*
This command transposes the single words to the right and left of point.

`C-x C-t` *transpose-lines*
This command transposes the line at point with the line before it.

`M-^` *delete-indentation*
This command joins the line at point with the previous line. Preface with `C-1` to join the line at point with the next line.

`M-u` *uppercase-word*
This command converts the text at point to the end of the word to uppercase

letters.

`M-l` *downcase-word*
This command converts the text at point to the end of the word to lowercase letters.

`C-x C-l` *downcase-region*
This command converts the region to lowercase letters.

`C-x C-u` *upcase-region*
This command converts the region to uppercase letters.

### Search and replace commands

`C-s [STRING] [C-w]` `[C-y]` *isearch-forward*
Incrementally search forward through the buffer for string (default is the last search string you gave, if any); `C-w` uses the text from point forward to the end of the word and `C-y` uses everything from point to the end of the line.

`C-r [STRING] [C-w]` `[C-y]` *isearch-backward*
Incrementally search backward through the buffer for string (default is the last search string you gave, if any); `C-w` uses the text from point forward to the end of the word, and `C-y` uses everything from point to the end of the line.

`C-s Enter C-w [WORD OR PHRASE]` *word-search-forward*
Search forward through the buffer for the given word or phrase, regardless of spacing.

`C-r Enter C-w [WORD OR PHRASE]` *word-search-backward*
Search backward through the buffer for the given word or phrase, regardless of spacing.

`C-M-s` *isearch-forward-regexp*
Incrementally search forward through the buffer for a given regular expression.

`C-M-r` *isearch-backward-regexp*
Incrementally search backward through the buffer for a given regular expression.

*replace-string*
Search for a given string from point to the end of the buffer and replace it with a given string.

*replace-regexp*
Search for a given regular expression from point to the end of the buffer and replace it with a given string.

`M-%` *query-replace*
Search for a given string from point to the end of the buffer and, in each instance, query to replace it with a given string.

`C-M-%` *query-replace-regexp*
Search for a given regular expression from point to the end of the buffer and, in each instance, query to replace it with a given string.

### Regular expressions

`.` *any character (but newline)*
`*` *previous character or group, repeated 0 or more time*
`+` *previous character or group, repeated 1 or more time*
`?` *previous character or group, repeated 0 or 1 time*
`^` *start of line*
`$` *end of line*
`[...]` *any character between brackets*
`[^..]` *any character not in the brackets*
`[a-z]` *any character between a and z*
`\` *prevents interpretation of following special char*
`\| or \w` *word constituent*
`\b` *word boundary*
`\sc` *character with c syntax (e.g. \s- for whitespace char)*
`\( \)` *start\end of group*
`\<` *start\end of word (faulty rendering:*
`\>` *backslash + less-than and backslash + greater-than)*
`\_< \_>` *start\end of symbol*
`\‘ \’` *start\end of buffer\string*
`\1` *string matched by the first group*
`\n` *string matched by the nth group*
`\{3\}` *previous character or group, repeated 3 times*
`\{3,\}` *previous character or group, repeated 3 or more times*
`\{3,6\}` *previous character or group, repeated 3 to 6 times*
`\=` *match succeeds if it is located at point*
`*?, +?, and ??` *non-greedy versions of *, +, and ?*
`\ca` *ascii character*
`\Ca` *non-ascii character (newline included)*
`\cl` *latin character*
`\cg` *greek character*
`[:digit:]` *a digit, same as [0-9] (\d is not supported)*
`[:alpha:]` *a letter (an alphabetic character)*
`[:alnum:]` *a letter or a digit (an alphanumeric character)*
`[:upper:]` *a letter in uppercase*
`[:lower:]` *a letter in lowercase*
`[:graph:]` *a visible character*
`[:print:]` *a visible character plus the space character*
`[:space:]` *a whitespace character, as defined by the syntax table, but typically [ \t\r\n\v\f], which includes the newline character*
`[:blank:]` *a space or tab character*
`[:xdigit:]` *an hexadecimal digit*
`[:cntrl:]` *a control character*
`[:ascii:]` *an ascii character*
`\s-` *whitespace character*
`\sw` *word constituent*
`\s_` *symbol constituent*
`\s.` *punctuation character*
`\s(` *open delimiter character*
`\s)` *close delimiter character*
`\s"` *string quote character*
`\s\` *escape character*
`\s/` *character quote character*
`\s$` *paired delimiter*
`\s’` *expression prefix*
`\s<` *comment starter*
`\s>` *comment ender*
`\s!` *generic comment delimiter*
`\s|` *generic string delimiter*

### Regex examples

`[-+[:digit:]]` *digit or + or - sign*
`\(\+\|-\)?[0-9]+` *decimal number*
`\(\.[0-9]+\)?` *(-2 or 1.5 but not .2 or 1.)*
`\<\(\w+\)+\1\>` *two consecutive, identical words*

`\<[[:upper:]]\w*` *word starting with an uppercase letter*

`+$` *trailing whitespaces (note the starting SPC)*

`\w\{20,\}` *word with 20 letters or more*

`\w+phony\>` *word ending by phony*

`\(19\|20` *year*

`\)[0-9]\{2\}` *1900-2099*

`^.\{6,\}` *at least 6 symbols*

`^[a-zA-Z0-9_]\{3,16\}$` *decent string for a user name*

`<tag[^>` `C-q` `C-j` `]*>\(.*?\)</tag>` *html tag*

### Registers commands

`C-x r space X` *point-to-register*
Save point to register X.

`C-x r s X` *copy-to-register*
Save the region to register X.

`C-x r r X` *copy-rectangle-to-register*
Save the selected rectangle to register X.

*view-register*
View the contents of a given register.

`C-x r j X` *jump-to-register*
Move point to the location given in register X.

`C-x r i X` *insert-register*
Insert the contents of register X at point.

### Abbreviations

`(setq-default abbrev-mode t)`
Write this into `.emacs` to switch *abbrev* minor mode on at start-up.

*abbrev-mode*
Toggles *Abbrev* mode; with a numeric argument, it turns *Abbrev* mode on if the argument is positive, off otherwise.

`C-x a g` *add-global-abbrev*
Define an abbrev, using one or more words before point as its expansion.

`C-x a l` *add-mode-abbrev*
Similar, but define an abbrev specific to the current major mode.

`C-x a i g` *inverse-add-global-abbrev*
Define a word in the buffer as an abbrev.

`C-x a i l` *inverse-add-mode-abbrev*
Define a word in the buffer as a mode-specific abbrev.

`M-'` *abbrev-prefix-mark*
Separate a prefix from a following abbrev to be expanded.

`C-x a e` *expand-abbrev*
Expand the abbrev before point. This is effective even when *Abbrev* mode is not enabled.

*expand-region-abbrevs*
Expand some or all abbrevs found in the region.

*list-abbrevs*
Display a list of all abbrev definitions. With a numeric argument, list only local abbrevs.

*edit-abbrevs*
Edit a list of abbrevs; you can add, alter or remove definitions.

`M-/` *dabbrev-expand*
Expand the word in the buffer before point as a dynamic abbrev, by searching for words starting with that abbreviation.

`C-M-/` *dabbrev-completion*
Complete the word before point as a dynamic abbrev.

### Bookmarks commands

`C-x r m Bookmark` *bookmark-set*
Set a bookmark named Bookmark.

`C-x r l` *bookmarks-bmenu-list*
List all saved bookmarks.

*bookmark-delete*
Delete a bookmark.

`C-x r b Bookmark` *bookmark-jump*
Jump to the location set in the bookmark named Bookmark.

*bookmark-save*
Save all bookmarks to the bookmark file, ~/.emacs.bmk.

### Window-manipulation commands

`C-x 2` *split-window-vertically*
Split the current window in half across the middle, stacking the new buffers vertically.

`C-x 4 b` *switch-to-buffer-other-window*
Split the current window in half vertically, prompting for the buffer to use the bottom window and making that the active window.

`C-x 4 C-o` *display-buffer*
Display a buffer in another window, prompting for the buffer to use the other window but keeping the current window active. (If only one window exists, then split the window vertically to display the other buffer.)

`C-x 4 f` *find-file-other-window*
Open a new file in a new buffer, drawing it in a new vertical window.

`C-x 4 r` *find-file-read-only-other-window*
Open a new file in a new read-only buffer, drawing it in a new vertical window.

`C-M-v` *scroll-other-window*
Scroll to the window that would be the next one to switch to with `C-x o`.

*scroll-all*
Toggle the scroll-all minor mode. When it's on, all windows displaying the buffer in the current window are scrolled simultaneously and in equal, relative amounts.

`C-x o` *other-window*
Move the cursor to the next window, and make it the active window.

`C-x 0` *delete-window*
Delete the current window, and move the cursor to the window that would be the next one to switch to with `C-x o`.

`C-x 1` *delete-other-windows*
Delete all windows except the current window.

`C-x 4 0` *kill-buffer-and-window*
Delete the current window, and kill its buffer.

`C-x 3` *split-window-horizontally*
Split the current window in half down the middle, stacking the new buffers horizontally.

*follow-mode*
Toggle follow, a minor mode. When it's on in a buffer, all windows displaying the buffer are connected into a large virtual window.

`C-x ^` *enlarge-window*
Make the current window taller by a line; preceded by a negative, this makes the current window shorter by a line.

`C-x }` *shrink-window-horizontally*
Make the current active window thinner by a single column.

`C-x {` *enlarge-window-horizontally*
Make the current active window wider by a single column.

`C-x -` *shrink-window-if-larger-than-buffer*
Reduce the current active window to the smallest possible size for the buffer it contains.

`C-x +` *balance-windows*
Balance the size of all windows, making them approximately equal.

*compare-windows*
Compare the current window with the next window, beginning with point in both windows and moving point in both buffers to the first character that differs until reaching the end of the buffer.

### Shell commands

`M-! <cmd>` *shell-command*
Run the shell command line `cmd` and display the output.

`M-| <cmd>` *shell-command-on-region*
Run the shell command line `cmd` with region contents as input; optionally replace the region with the output.

*shell*
Run a subshell with input and output through an Emacs buffer. You can then give commands interactively.

*term*
Run a subshell with input and output through an Emacs buffer. You can then give commands interactively. Full terminal emulation is available.

*eshell*
Start the Emacs shell.

### Interactive Highlighting

`C-x w h [regexp] <RET> [face] <RET>` *highlight-regexp*
Highlight text that matches regexp using `face` face. The highlighting will remain as long as the buffer is loaded.

`C-x w r [regexp] <RET>` *unhighlight-regexp*
Unhighlight regexp.

`C-x w l [regexp] <RET> [face] <RET>` *highlight-lines-matching-regexp*
Highlight entire lines containing a match for regexp, using `face` face.

### nXML mode

`C-c C-n` *rng-next-error*
Move to the next location where the document structure is not valid.

`C-c C-v` *rng-validate-mode*
Turn validation on or off. If validation is turned on, in the status line's mode area you will see either "nXML Valid" or "nXML Invalid". If validation is turned off, neither word will appear after "nXML" in the mode line.

`tab` *indent-for-tab-command*
Indent the current line according to the level of nested block tags. The indentation is two spaces per level.

`M-C-\` *indent-region*
Indent all the lines in the region using the same process as for tab.

`C-c C-f` *nxml-finish-element*
Insert an end tag for whatever element the cursor is in.

`C-c C-i` *nxml-balanced-close-start-tag-inline*
This command adds the closing ">" and an end tag, and then places the cursor between the tags so you can type the content.

`C-c C-b` *nxml-balanced-close-start-tag-block*
Adds the closing ">", then a blank line, then an end tag on yet another separate line. The cursor is left indented at the proper level on the central blank line.

`M-q` *fill-paragraph*
Reformat the paragraph containing the cursor.

`C-c C-x` *nxml-insert-xml-declaration*
Inserts an XML processing instruction at the top of the file.

`M-C-f` *forward-sexp*
Move forward over tag.

`M-C-b` *backward-sexp*
Move backward over tag.

`M-C-n` *nxml-forward-element*
Move the cursor to the end of the next element.

`M-C-p` *nxml-backward-element*
Move the cursor before the previous element.

`M-C-d` *nxml-down-element*
Move the cursor to the next included element after point, to a position just after the start tag; d is for "down."

`M-C-u` *nxml-backward-up-element*
Move the cursor to a position just before the start tag of the element containing point; u is for "up."

`C-c C-o C-d` *nxml-hide-subheadings*
Hide the children of the current element, as in emacs outline-mode.

`C-c C-o C-s` *nxml-hide-subheadings*
Reverses the action of `C-c C-o C-d`, revealing the children of the current element.

### Managing variables

*set-variable*
Ask for a variable to change and for the needed value.

*auto-mode-alist*
If set to *nil*, automatic selection of major mode based on file name extension is turned off. Its default value is a list of file name extensions and corresponding modes.

*auto-save-default*
If not set to *nil*, Emacs automatically saves a changed buffer to its corresponding file at preset intervals. Its default value is t.

*auto-save-interval*
Contains the number of character changes after which Auto-save mode, if true, is invoked; the default value is 300.

*calendar-latitude*
Contains the latitude value for the location of the user's workstation, in degrees; the default value is *nil*.

*calendar-longitude*
Contains the longitude value for the location of the user's workstation, in degrees; the default value is *nil*.

*calendar-location-name*
Contains the value for the location name (such as city, state, and country) for the location of the user's workstation; the default value is *nil*.

*colon-double-space*
If not set to *nil*, commands for filling

text insert two spaces after a colon instead of one. The default value is nil.

*command-line-args*
Contains the list of arguments used in the command line that executed the current Emacs session.

*command-line-default-directory*
Contains the path name of the directory from which the current Emacs session was executed.

*compare-ignore-case*
If not set to *nil*, Emacs ignores differences in uppercase and lowercase letters when running the compare-windows function, as described in fifth installment of this series (see Resources). The default value is *nil*.

*confirm-kill-emacs*
If set to *nil*, Emacs doesn't ask for a confirmation when exiting; otherwise, the exit verification might be customized as an Emacs Lisp function such as *y-or-n-p*. The default value is *nil*.

*default-justification*
Sets the default justification style. The value can be one of left, right, center, full, or none. The default value is left.

*default-major-mode*
Selects the default major mode for new files or buffers. The default value is *fundamental-mode*.

*display-time-24hr-format*
If set to t, Emacs displays time in 24-hour military format, instead of the standard 12-hour format with AM or PM suffix. The default value is *nil*.

*display-time-day-and-date*
If not set to *nil*, Emacs displays time with the current day of the week, current month, and current day of the month, instead of just the hour and minute. The default value is *nil*.

*fill-column*
Contains the number for the column on each line where text begins to be filled to the next line. The default value is 70.

*initial-major-mode*
Specifies the major mode to use for the **scratch** buffer on startup. The default value is *lisp-interaction-mode*.

*inverse-video*
If not set to *nil*, Emacs inverts the display colors, if possible. The default value is *nil*.

*kill-ring*
Contains the contents of the Emacs kill ring.

*kill-ring-max*
Sets the number of allowable entries in the Emacs kill ring. The default value is 60.

*kill-whole-line*
If not set to *nil*, the kill-line function (bound to `C-k`) kills the current line and its trailing newline character, if the function is executed at the very beginning of the line. The default value is *nil*.

*make-backup-files*
If not set to *nil*, Emacs saves a backup of a buffer before any changes are made to a file of the same name but with a tilde character (`~`) appended to the end.

*mark-ring*
Contains the contents of the current mark ring of the buffer.

*mark-ring-max*
Contains the number of allowable entries in the mark ring. The default value is 16.

*mouse-avoidance-mode*
Contains a value describing the type of mouse-avoidance mode. The default value is *nil*.

*next-line-add-newline*
If not set to *nil*, Emacs adds a new line whenever the down arrow is pressed at the end of the buffer. The default value is *nil* (in more recent versions of Emacs).

*scroll-bar-mode*
Contains the value for the side of the Emacs frame on which to place the scroll bar: right or left. If set to *nil*, the scroll bar is turned off. The default value is left.

*scroll-step*
Contains the number of lines to move through the buffer lines with the *scroll-down* and *scroll-up* functions (which are bound to the `PgDn` and `PgUp` keys by default). If set to 0, Emacs centers point in the middle of the window when scrolling.

*show-trailing-whitespace*
If not set to *nil*, Emacs makes any whitespace at the end of lines in the current buffer visible. The default value is *nil*.

*visible-bell*
If not set to *nil*, Emacs makes the frame blink instead of ringing the audible system bell. The default value is *nil*.

*x-cut-buffer-max*
Sets the maximum number of characters from the kill ring that are also stored in the X Window System cut buffer. The default value is 20000.

**Interface functions**

*column-number-mode*
Toggle the display, in the mode line, of the current column the cursor is at, preceded by a `C`. The default value is *nil*.

*display-time*
Toggle the display of the current time in the mode line. The default value is *nil*.

*font-lock-mode*
If not set to *nil*, Emacs turns on the Font Lock mode automatically for the current buffer. The default value is *nil*.

*global-font-lock-mode*
If not set to *nil*, Emacs turns on the Font Lock mode automatically for all buffers. The default value is *nil*.

*line-number-mode*
Toggle the display, in the mode line, of the current line the cursor is at, preceded by an `L`. The default value is *t*.

*show-paren-mode*
Allows one to see matching pairs of parentheses and other characters. When point is on one of the paired characters, the other is highlighted.

*menu-bar-mode*
Toggle the display of the Emacs menu bar. The default value is *t*.

*sunrise-sunset*
Display the time of today's sunrise and

sunset for the current geographic location. If preceded with the **universal-argument**, this function prompts for a specific day.

*tool-bar-mode*
Toggle the display of the Emacs toolbar. The default value is *t*.

### Customize functions

*customize-changed-options* <*Enterversion*>
Open a new customization buffer for all faces, options, or groups that have been changed since the version of Emacs given by version.

*customize-customized*
Open a new customization buffer for all options and faces that have already been customized but haven't been saved to disk.

*customize-face* <*Enter*> <*regexp*>
Open a new customization buffer for all the face, option, or groups relevant to the regular expression given by **<regexp>**.

*customize-face* <*Enter*> <*face*>
Open a new customization buffer for the face name given by **<face>**.

*customize-group* <*Enter*> <*group*>
Open a new customization buffer for the group name given by **<group>**.

*customize-option* <*Enter*> <*option*>
Open a new customization buffer for the option name given by **<option>**.

*customize-saved*
Open a new customization buffer for all faces and options that you've changed with the Customize function.

### Help commands

**Tab**
This command performs command completion if given as part of a command, showing all possible input values for the given command.

**<command prefix> or <keystroke>**
**C-h**
This command describes all the possible commands and functions available for the given **<command prefix>** or **<keystroke>**.

**C-h c <keystroke>** *describe-key-briefly*
This command reports in the minibuffer the name of function that **<keystroke>**is bound to.

**C-h k <keystroke>** *describe-key*
This command opens a new help-buffer window that describes the function that **<keystroke>** is bound to.

**C-h l** *view-lossage*
This command opens a new buffer and displays the last 100 characters typed.

*open-dribble-file*
This command opens a specified file and dribbles a copy of all keyboard input to that file.

*apropos*
This command gives a list of apropos commands and variables to a given **regexp**.

**C-h a <regexp>** *command-apropos*
This command gives a list of apropos commands to regexp.

**C-h b** *describe-bindings*
This command describes all the valid key bindings for the current major mode in a new help buffer window.

**C-h f <function>** *describe-function*
This command describes the purpose of **<function>** in a new help buffer window.

**C-h v <variable>** *describe-variable*
This command describes the purpose of **<variable>** in a new help buffer window.

**C-h w <function>** *where-is*
This command describes which keyboard binding (if any) a particular **<function>** is bound to.

**C-h s | F1 s** *describe-syntax*
Display the *Emacs Syntax Table* for the current mode (useful for the matching parentheses commands).

### INFO commands

**H** *Info-help*
This command opens a hands-on **Info** tutorial in a new buffer.

**Q** *Info-exit*
This command moves to the last buffer you visited, putting the ***info*** buffer in the end of the buffer list.

**<Enter>**
This command follows the cross reference at or near point.

**N** *Info-next*
This command moves to the current node's **Next** node.

**P** *Info-prev*
This command moves to the current node's **Previous** node.

**U** *Info-up*
This command moves to the current node's **Up** node.

**D** *Info-directory*
This command moves to the **Directory** node.

**L** *Info-last*
This command moves to the last node you visited.

**T** *Info-top-node*
This command moves to the **Top** node of the current document.

**>** *Info-final-node*
This command moves to the final node pointed to in the current document.

**<Spacebar>** *Info-scroll-up*
This command moves forward in the current node by a single screen; if at the end of the node, then move to the Next node.

**<Backspace>** *Info-scroll-down*
This command moves backward in the current node by a single screen; if at the beginning of the node, then move to the Previous node.

**B** *beginning-of-buffer*
This command goes to the beginning of the current node.

**S** *Info-search*
This command searches forward in the current **Info** document for a given **regexp**.

**Tab** *Info-next-reference*
This command moves the cursor forward to the first cross reference.

**M-Tab** *Info-prev-reference*
This command moves the cursor

backward to the last cross reference.

### Documentation files

**C-h C-d** *describe-distribution*
File **DISTRIB**: Information on obtaining a copy of the latest distribution of the Emacs software

**C-h F** *view-emacs-faq*
**/usr/share/info/**; **emacs-mainversion/efaq.gz** Emacs FAQ

**C-h C-c** *describe-copying*
File **COPYING**: GNU General Public License (GNU GPL)

**C-h C-w** *describe-no-warranty*
File **COPYING**: Section "NO WARRANTY" of the GNU General Public License (GNU GPL)

**C-h n** *view-emacs-news*
File **NEWS**: News concerning the latest changes in the current version of Emacs

**C-h P** *view-emacs-problems*
**File PROBLEMS**: Emacs problems file

**C-h C-p** *describe-project*
File **THE-GNU-PROJECT**: Essay by Richard Stallman concerning the founding of the GNU Project

**C-h t** *help-with-tutorial*
File **TUTORIAL**: Hands-on tutorial for learning the basics of Emacs

### Things you should never know about

*hanoi*
The hanoi tower, with a default of 3 discs; **M-x hanoi-unix** and **M-x hanoi-unix-64** uses the unix timestamp, making a move each second in line with the clock, and with the latter pretending it uses a 64-bit clock.

*5x5*
You are given a 5x5 grid with a central cross already filled-in; your goal is to fill all the cells by toggling them on and off in the right order to win. With an optional digit argument you can change the size of the grid.

*animate-birthday-present*
A fancy birthday present animation.

*butterfly*
The animate package is also used by **M-x butterfly** command, a command added to Emacs as an homage to the XKCD strip at **www.xkcd.com/378/**

*blackbox*
The object of the game is to find four hidden balls by shooting rays into the black box. There are four possibilities: 1) the ray will pass thru the box undisturbed, 2) it will hit a ball and be absorbed, 3) it will be deflected and exit the box, or 4) be deflected immediately, not even being allowed entry into the box.

*bubbles*
You must clear out as many "bubbles" as you can in as few moves as possible. When you remove bubbles the other bubbles drop and stick together. You can configure the difficulty of the game by calling **M-x bubbles-set-game-<difficulty>** where <difficulty> is one of: **easy**, **medium**, **difficult**, **hard**, or **userdefined**. Furthermore, you can alter the graphics, grid size and colors using Customize: **M-x customize-group bubbles**.

*decipher*
It's a (very complex) package to help you break simple substitution ciphers (like cryptogram puzzles) using a helpful user interface.

*dissociated-press*
It's a semi-randomizing algorithm that takes your buffer, runs it through a blender, and displays the result.

*doctor*
Based on the original *ELIZA*, the "Doctor" tries to psychoanalyze what you say and attempts to repeat the question back to you. It simulates a Rogerian psychotherapist and uses rules, dictated into a script, to respond with non-directional questions to user inputs.

*dunnet*
Emacs's very own Zork-like text adventure game.

*gomoku*
You have to connect 5 squares, tic-tac-toe style. You can customize the group **gomoku** to adjust the size of the grid.

*life*
Conway's Game of Life is a famous example of cellular automata. The Emacs version comes with a handful of starting patterns that you can (programmatically with elisp) alter by adjusting the **life-patterns** variable.

*pong* | *snake* | *tetris*
These classic games are all implemented using the Emacs package gamegrid, a generic framework for building grid-based games like Tetris and Snake.

*solitaire*
It is a peg-based game where you have to end up with just one stone on the board, by taking a stone (the **o**) and "jumping" over an adjacent stone into the hole (the **.**), removing the stone you jumped over in the process. Rinse and repeat until the board is empty. There is a handy solver built in called **M-x solitaire-solve** if you get stuck.

*zone*
A series of screensavers. Type **M-x zone** and watch what happens to your screen! You can configure a screensaver idle time by running **M-x zone-when-idle** (or calling it from elisp) with an idle time in seconds. You can turn it off with **M-x zone-leave-me-alone**.

*mpuz*
A multiplication puzzle where you have to replace the letters with numbers and ensure the numbers add (multiply?) up. You can run **M-x mpuz-show-solution** to solve the puzzle if you get stuck.

*morse-region*
Translates a region into morse code. Undo (**C-S-_**) or **M-x unmorse-region** to unmorse.

---



For more Emacs documentation see the Emacs distribution or http://www.gnu.org/software/emacs
For the XML-XSLT source for this cheat sheet see https://github.com/PeteZaSayari/emacs-cs.