↖ Topic Index

Frames

Operation	Keystroke	Function	Note
Emacs Frames	• Emacs calls frames what model graphical Operating Systems call “windows”, or more specifically “OS windows”. • Emacs supports multiple frames, both when Emacs works in graphical mode but also when it is used in text terminal mode.		
Open this PDF file. See also: ∑ Help/Info	`<f11> F <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the ∑ **Frames** 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 **PEL frame control**	`<f11> F <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL frame management support. • If OTHER-WINDOW is non-nil (use **`C-u`**), display in other window.
∑ Customize **Emacs frame control**	`<f11> F <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs frame management support.
Enter/Exit full screen	`<f11> <f11>`	(**pel-toggle-frame-fullscreen**)	Toggle frame fullscreen mode on/off in graphics mode.
	• In Terminal mode, issue error to show how to use the OS keystrokes to toggle the fullscreen mode. For example it will tell you to use **`⌘-C-f`** when Emacs is running inside a Terminal.app frame. • 🚧 Unfortunately the information is only provided for macOS Terminal.app and iTerm.app. • ✂ Standard GNU Emacs normally binds **`<f11>`** to (toggle-frame-fullscreen), which the PEL command uses. PEL changes that binding; instead it binds **`<f11><f11>`** to the command in order to use **`<f11>`** as a prefix key.		
Set Frame Font	With Emacs running in graphics mode, you can change the font of all windows with the menu-set-font command.		
Change font of current Frame See also: ∑ Faces/Fonts	`<f11> F F`	(**menu-set-font**)	Interactively select a font and make it the default on all frames. • The selected font will be the default on both the existing and future frames in the current session. It is not persistent, so next time you start Emacs the default font is used.
Move directly to another graphical frame using cursors	The following commands move point to another frame existing in the direction selected by the cursor key. ⚠️These commands only work in **graphics mode**, in terminal mode, Emacs frames are sharing the same terminal application frame window. 📦 These require the framemove package, which unfortunately is currently not supported by an Elpa or MELPA package archive. For the moment PEL does not support installing such packages, so you must install it yourself inside a directory dedicated for such extra files and that directory must be located inside Emacs load-path. You can get this package from the Emacs Wiki framemove or from Github framemove page. 🔛 PEL activates this when the **pel-use-framemove** user option is set to **t**.		
Move to frame above	• `<f11> <S-up>` • `<Esc> <S-up>`	(**fm-up-frame**)	Move point to frame above current frame (if one exists and is located there).
Move to frame below	• `<f11> <S-down>` • `<Esc> <S-down>`	(**fm-down-frame**)	Move point to frame below current frame (if one exists and is located there).
Move to frame at right	• `<f11> <S-right>` • `<Esc> <S-right>`	(**fm-right-frame**)	Move point to frame at right of the current frame (if one exists and is located there).
Move to frame at left	• `<f11> <S-left>` • `<Esc> <S-left>`	(**fm-left-frame**)	Move point to frame at left of the current frame (if one exists and is located there).
Manage Frames	Emacs frame operations work in **both** modes: graphics and text terminal. In graphics mode a new frame is creating a separate OS frame. Although surprising, in text terminal mode a new frame is inside the **same** OS frame: it simply hides the other(s) existing frame(s) as the text terminal can only display one Emacs frame at any given time. When Emacs runs in text terminal mode, each mode line identifies the currently active frame number with “F1”, “F2”, etc… The windows numbers are all part of the space number-space.		
Show frame count	`<f11> F ?`	(**pel-show-frame-count**)	Display the number of Emacs active frames in the mini-buffer.
Delete this frame	• **`C-x 5 0`** • `<f11> F 0` • `⌘-w`	(**delete-frame** &optional FRAME FORCE)	Delete FRAME, permanently eliminating it from use. FRAME must be a live frame and defaults to the selected one.
Delete all other frames	• **`C-x 5 1`** • `<f11> F 1`	(**delete-other-frames** &optional FRAME)	Delete all frames on FRAME’s terminal, except FRAME.
New Frame below	• **`C-x 5 2`** • `<f11> F 2` • `⌘-n`	• (**make-frame-command**) • (**make-frame** &optional PARAMETERS)	Make a new frame, on the same terminal as the selected frame. On macOS in graphics mode only: make-frame is called for `⌘-n`, it has the same effect as make-frame-command.
Display buffer in other (next) frame	• **`C-x 5 C-o`** • `<f11> F b`	(**display-buffer-other-frame** BUFFER)	Display a buffer preferably in another frame.
Run Dired in other (next) frame	• **`C-x 5 d`** • `<f11> F d`	(**dired-other-frame** DIRNAME &optional SWITCHES)	"Edit" a directory. Like ‘dired’ but makes a new frame.
Find file in other (next) frame	• **`C-x 5 f`** • **`C-x 5 C-f`** • `<f11> F f`	(**find-file-other-frame** FILENAME &optional WILDCARDS)	Switch to/open another file and show it in another frame.

Select next frame	• `<f11> F n`
	• `⌘-`` ` ``
		(**pel-next-frame** ARG)	Make next frame visible.
			• In text terminal mode: iterate through all frames.
			• In graphics mode:
			  • with no/nil argument (the default): iterate through visible frames,
			  • with ARG non nil: iterate through all frames (included iconified frames).
			The `⌘-` ` ` key is a macOs command that moves to the next frame of the same application: no Emacs code is invoked but the effect is the same.

Move cursor to other (next) frame	• **`C-x 5 o`** • `<f11> F o`	(**other-frame** ARG)	Activate the other frame. Numeric argument identifies frame in Z order.
Select buffer in other frame	• **`C-x 5 O`** • `<f11> F O`	(**switch-to-buffer-other-frame** BUFFER-OR-NAME &optional NORECORD)	Prompt for buffer and open in other frame.

Select previous frame	• `<f11> F p`
	• `⌘-_`
		• (**pel-previous-frame** ARG)
		• (**ns-prev-frame**)
			Make previous frame visible.
			• In text terminal mode: iterate through all frames.
			• In graphics mode:
			  • with no/nil argument (the default): iterate through visible frames,
			  • with ARG non nil: iterate through all frames (included iconified frames).

Find file in other (next) frame in read-only	• **`C-x 5 r`** • `<f11> F r`	(**find-file-read-only-other-frame** FILENAME &optional WILDCARDS)	Edit file read-only in other frame with name obtained via minibuffer.
Hide Emacs frame	`⌘-h`	(**ns-do-hide-emacs**)	On macOS in graphics mode only: hide Emacs frame. • Retrieve it via the **`⌘-<tab>`** key.
Hide all other applications	`⌘-H`	(**ns-do-hide-others**)	On macOS in graphics mode only: hide all other applications, except Emacs. • Retrieve them via the **`⌘-<tab>`** key.
Iconify frame	`⌘-m`	(**iconify-frame** &optional FRAME)	On macOS in graphics mode only: iconify the frame.