↖ Topic Index

Hide/Show, Fold Code Blocks & Selective Display

Operation	Keystroke	Function	Note
Hide/Show: • **Text** • **Lines** • **Comments** **Code folding** See also: **Outline**	With Emacs and external packages there are several ways you can hide text inside a buffer. • Emacs provides the HideShow Minor Mode • PEL provides access to the following external packages and libraries that extend the basic capabilities of Emacs: 📦 The **hide-cmnt** external library. PEL activates it when the **pel-use-hide-comnt** user option is is turned on (set to **t**). 📦 The **hide-lines** external package PEL installs and activates it when the **pel-use-hide-lines** user-option is is turned on (set to **t**). 📦 The **Hydra** external package PEL provides a Hide/Show Hydra when **pel-use-hydra** user option is is turned on (set to **t**). 📦 The **origami** external package PEL installs and activates it when the **pel-use-origami** user-option is is turned on (set to **t**). PEL provides extra key bindings for the commands for these packages. It also provides bindings for controlling visibility of comment and docstrings in the **origami-mode** key-map to simplify the hiding/showing of code, comments and docstrings.		
HideShow Minor Mode	When working with source code files, you can use the Hide/Show minor mode to collapse and expand blocks of code, where the concept of "*block of code*" depends on the specific programming language. For example C-like programming language use braces to delimit blocks, while in Lisp languages use parentheses for all blocking. • When a block is hidden (collapsed) it is replaced by "…" surrounded by the block delimiter of the specific programming language. • ☝📦 PEL provides an easy to use Hydra invoked via the **<f7> /** key prefix to control the Hide/Show mode and quickly issue commands to hide or show portions of the text when the **pel-use-hydra** is set to **t** to activate the **external hydra package**. The next row provides more information. • Without PEL you must activate the Hide/Show mode with M-x hs-minor-mode and then use the Emacs commands bound to **C-c @** key prefix. • Once the Hide/Show minor mode is active the mode line will show "hs" . • When you disable the Hide/Show mode, hidden text is restored. ☝ You may want to hide comments as well when collapsing code: you can use **<f11> ; ;** to toggle visibility of comments,		
Open this PDF file. See also: **Help/Info**	**<f11> M-/ <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **Hide/Show** 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 highlighting control	**<f11> M-/ <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL support for buffer hide/show management: hide-cmnt, hide-lines. • If OTHER-WINDOW is non-nil (use **C-u**), display in other window.
Customize Emacs hide control	**<f11> M-/ <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs support for: 1. hideshow 2. hide-lines 3. origami
HideShow Hydra Type **<f7> /** followed by one of the keys in the hydra to activate this hydra:	**Using the PEL HideShow Hydra** : control hiding of all or current **code block(s)**. 📦 PEL provides a Hide/Show **Hydra** when **pel-use-hydra** user option is set to **t**. • Activate this hydra with the **<f7> /** key prefix. • Once this hydra is active, you can then type any of the keys in the hydra (see the menu below) without having to type the **<f7> /** prefix again and so until you terminate the hydra by typing the **<f7>** key again. • While active the hydra displays the menu shown below and operation results below the menu. • You can issue any other command (not bound to the keys listed in the menu) while the hydra is active. • You do not have to activate the **hs-minor-mode** for any of the commands in the menu: they automatically activate it. • Use **<f7> / / <f7>** to de-activate **hs-minor-mode** (or **/ <f7>** if the hydra is already active). Hide/Show: State / Hide/Show / Hide / Show / Hide levels / Hide levels: / End /: Toggle hs mode / a: all / H: all / S: all / 1: >= 1 / >: +1 / <f7>: cancel ?: info / b: block / h: block / s: block / 2: >= 2 / <: -1 3: >= 3 4: >= 4		

Toggle Hide/Show Minor Mode	✱ **<f7> / /**	(**hs-minor-mode** &optional ARG)	Toggle Hide/Show minor mode to selectively hide/show code and comment blocks.
			• With a prefix argument ARG, enable the mode if ARG is positive, and disable it otherwise.
			• When hideshow minor mode is on:
			  • The menu bar is augmented with hideshow commands and the hideshow commands are enabled.
			  • The line-mode shows 'hs'.

Describe current state of PEL show/hide	✱**<f7> / ?**	(**pel-show-hide-state**)	Display state of pel-hideshow in current buffer.
Show (expand) all blocks in buffer	✱**<f7> / S**	(**pel-show-all**)	Show all blocks.
	• **C-c @ C-M-s** • **C-c @ C-a**	(**hs-show-all**)	
Hide (collapse) all blocks in buffer	✱**<f7> / H**	(**pel-hide-all**)	Hide all top level blocks, displaying only first and last lines.
	• **C-c @ C-M-h** • **C-c @ C-t**	(**hs-hide-all**)	
Hide (collapse) current block	✱**<f7> / h**	(**pel-hide-block** &optional END)	Select a block and hide it. • With prefix arg, reposition at END.
	• **C-c @ C-h** • **C-c @ C-d**	(**hs-hide-block** &optional END)	
Show (expand) current block	✱**<f7> / s**	(**pel-show-block** &optional END)	Select a block and show it. • With prefix arg, reposition at END.
	C-c @ C-s	(**hs-show-block** &optional END)	
Toggle visibility of all blocks in buffer	✱**<f7> / a**	(**pel-toggle-hide-all**)	Toggle hide/show of all blocks. • Activates the Hide/Show mode if not already active (and hide all blocks)
Toggle visibility of current block	✱**<f7> / b**	(**pel-toggle-hide-block**)	Toggle hide/show of current block.
	• **C-c @ C-c** • **C-c @ C-e**	(**hs-toggle-hiding**)	
Hide all blocks 1 level below current block	✱**<f7> / 1**	(**pel-hide-level-1**)	Hide all blocks 1 level below the current block. ☝ Useful in language like Python to show the methods of a class.
Hide all blocks 2 level below current block	✱**<f7> / 2**	(**pel-hide-level-2**)	Hide all blocks 2 level below the current block.
Hide all blocks 3 level below current block	✱**<f7> / 3**	(**pel-hide-level-3**)	Hide all blocks 3 level below the current block.
Hide all blocks 4 level below current block	✱**<f7> / 4**	(**pel-hide-level-4**)	Hide all blocks 4 level below the current block.
Hide all blocks N levels below this block.	**C-u n C-c @ C-l**	(**hs-hide-level** ARG)	Hide all blocks ARG levels below this block. • Like all other commands that take a numeric argument, the numeric argument (shown in the keystroke column as **C-u n**, can also be typed with the **M-***number*)
Hide one more extra level below current level	✱**<f7> / >**	(**pel-hs-hide-block-below-inc**)	Hide all blocks of 1 more level deep below this block level of point. • Warns/stops upon reaching the limit of +10 levels (a hard-coded limit) of blocks.
Hide one less level below current level	✱**<f7> / <**	(**pel-hs-hide-block-below-dec**)	Hide all blocks of 1 less level deep below the block level of point. • Warns/stops upon going down to +0 levels.

Operation	Keystroke	Function	Note
Selective Display	As stated in the Emacs manual: "Emacs has the ability to hide lines indented more than a given number of columns. You can use this to get an overview of a part of a program. • To hide lines in the current buffer, type **C-x $** (set-selective-display) with a numeric argument n. Then lines with at least n columns of indentation disappear from the screen. The only indication of their presence is that three dots ('...') appear at the end of each visible line that is followed by one or more hidden ones. • The commands **C-n** and **C-p** move across the hidden lines as if they were not there. • The hidden lines are still present in the buffer, and most editing commands see them as usual, so you may find point in the middle of the hidden text. When this happens, the cursor appears at the end of the previous line, after the three dots. If point is at the end of the visible line, before the newline that ends it, the cursor appears before the three dots. • To make all lines visible again, type **C-x $** with no argument."		
Set/clear selective display of lines with indentation >= n	**C-x $**	(**set-selective-display** ARG)	Set 'selective-display' to ARG; clear it if no arg. • When the value of 'selective-display' is a number > 0, lines whose indentation is >= that value are not displayed. • The variable 'selective-display' has a separate value for each buffer.
Start selective display Hydra Type **<f7> C-x $** followed by one of the hydra keys to activate this hydra	✱**<f7> C-x $**	**pel-𝕀hide-indent**	📦🔀 With PEL, when **pel-use-hydra** is set to **t**, this key starts a **hydra** to manage selective display and easily move the selective display column left or right with the cursors by column or indentation level, to stop it (**0**), to hide most lines (**1**) and to highlight the right-most visible column (|). The hydra menu that appears in the minibuffer when the hydra is active is shown below. • You can execute other commands while this hydra is active. • Terminate the Hydra by typing the **<f7>** key again.
	Selective Display: By Column | By Indent | Show | End <right>: +1 | S-<right>: +indent | |: rightmost visible limit | <f7>: cancel <left>: -1 | S-<left>: -indent 0: unhide 1: hide at 1		
Show all text	✱**<f7> C-x $ 0**	(**pel--sd-uc**)	Set **selective-display** to nil: un-hide all text.
Hide all text that is indented	✱**<f7> C-x $ 1**	(**pel--sd-h1**)	Set **selective-display** to 1: hide all text that does not start on the beginning of the line.
Increment selective-display by 1 column	✱**<f7> C-x $ <right>**	(**pel-selective-display-column-inc** N)	Increments **selective-display** by N. N is 1 unless a key prefix argument is used. • This hides text indented by that many columns.
Decrement selective-display by 1 column	✱**<f7> C-x $ <left>**	(**pel-selective-display-column-dec** N)	Decrement **selective-display** by N. N is 1 unless a key prefix argument is used. • This hides text indented by as many columns as specified by **selective-display** value.
Increment selective-display by 1 indent level	✱**<f7> C-x $ S-<right>**	(**pel-selective-display-indent-inc** N)	Increment variable 'selective-display' by N indentation levels. • N defaults to 1. If indentation level is not known, use N columns. • This hides text indented a the new value of variable '**selective-display**'.
Decrement selective-display by 1 indent level	✱**<f7> C-x $ S-<left>**	(**pel-selective-display-indent-dec** N)	Decrement variable 'selective-display' by N indentation levels. • N defaults to 1. If indentation level is not known, use N columns. • This hides text indented a the new value of variable 'selective-display'.
Show selective-display column limit visually with vline-mode	✱**<f7> C-x $ |**	(**pel--sd-maybe-vline-mode**)	Show selective-display column limit using the **vline-mode** if available. 📦 Requires: **vline.el** 🔀 activated when **pel-use-vline** user option is **t**.
Hide/Show Comments See also: 𝕀 **Comments**	The **hide-cmnt** file, written by Drew Adams, provides the following commands to quickly hide/show the comments in a buffer. 📦 PEL provides binding for these 2 commands and 🔀 installs the **hide-cmnt EmacsMirror** , a mirror of the original **hide-comnt.el EmacsWiki** repo in the PEL utils directory automatically when the **pel-use-hide-comnt** user—option is set to **t**. ☝These are very useful to see a list of methods without all comments when you also use the Hide/Show Mode commands.		
Toggle display of comments in buffer or active region	**<f11> ; ;** **M-/ M-;**	(**hide/show-comments-toggle** &optional START END)	Toggle hiding/showing of comments in the active region or whole buffer. • If the region is active toggle in the region, otherwise, in the whole buffer. • The **M-/ M-;** key binding is available when **origami-mode** is active.

Show (or hide) comments in buffer	**<f11> ; :**	(**hide/show-comments** &optional HIDE/SHOW START END)	Hide or show comments in buffer or active region.
			• Hide if no argument. To show, use any prefix argument (any of the **C-u**, **M- -**, **M-0** to **M-9** will do).
			• If a region is active the command applies to the active region, otherwise it applies to the entire or narrowed buffer.
			• Uses 'save-excursion', restoring point.
			• Option 'show-invisible-comments-shows-all':
			  • If non-nil then using this command to show invisible text shows *ALL* such text, regardless of how it was hidden. IOW, it does not just show invisible text that you previously hid using this command.
			  • If nil (the default value) then using this command to show invisible text makes visible only such text that was previously hidden by this command. (More precisely, it makes visible only text whose 'invisible' property has value 'hide-comment'.)

Hide/Show docstrings	Some programming languages support the concept of **docstrings**. These are highlighted differently than comments by Emacs. In some programming languages (Lisp, Python, Clojure) the docstring for functions appears between the argument list and the function body. In some cases the docstring is long and being able to quickly hide it helps when editing or reviewing source code. Docstrings can also be present in other places, at the beginning of a Python module or class definition for example. PEL provides its own facility to selectively hide and show the docstring of definitions for languages that support the concept of docstrings. With the commands below you can hide and show back the docstring of the current, previous or next Emacs Lisp definition (function, macro, defsubst, etc…) **Limitations:** • It currently only supports Lisp type languages and Python, yet the command is available everywhere. • Does not properly handle the ability to hide several docstrings, then show back some of them. That will work as long as you do not use another command that uses the visible property, such as comment hiding. Docstring and comment hiding co-exists without problem when only hiding one docstring at a time. Elixir, Haskell and Julia also support docstrings, but they are located before the function definition, just like documentation comments used by conventions in languages that do not support docstring. The following commands do not support those languages for the moment. There's also Erlang's Typer specifications. I might want to add some support there. My plan is to first complete robust support for Emacs Lisp, then for Python and then check if the Emacs Lisp works well with Clojure and Common Lisp. For now just be careful when using these commands.		
Toggle visibility of docstring	**<f11> ; d** **M-/ M-d**	(**pel-toggle-docstring** &optional NEXT SILENT)	Toggle the visibility of the docstring. • By default it affects the current or previous definition, but with any prefix argument (like **C-u**, **C--** or **M--**) toggles the docstring visibility of the next definition. • Return t on success. If no docstring detected issue a user-error by default. But if SILENT is non-nil, instead of issuing an error return nil instead. • The **M-/ M-d** key binding is available when **origami-mode** is active.
Toggle visibility of all docstrings in buffer	**<f11> ; D** **M-/ M-D**	(**pel-toggle-all-docstrings**)	Toggle visibility of all docstrings in buffer. • Display the number of docstrings affected. • The visibility of docstring is affected, but the buffer content is unchanged. • The **M-/ M-D** key binding is available when **origami-mode** is active.
Hide/show docstring	**<f11> ; '**	(**pel-hide/show-docstring** &optional SHOW SILENT)	Hide or show the docstring of current or previous definition. • Hide the docstring. • With any prefix argument (like **C-u**, **C--** or **M--**) show the docstring. Return t on success. If no docstring detected issue a user-error by default, but if SILENT is non-nil, instead of issuing an error return nil instead.

Operation	Keystroke	Function	Note
Hide all docstrings in buffer	`<f11> ; "`	(**pel-hide/show-all-docstrings** &optional SHOW)	Hide all docstrings in buffer. • With optional SHOW argument (any prefix argument like **C-u**, **C--** or **M- -**), show them all instead. • Display the number of docstrings affected. • The visibility of docstring is affected, but the buffer content is unchanged.
Hide/Show Lines matching regex	The following commands control the hiding of buffer lines using regular expressions. 📦 This requires the **hide-lines** external package PEL installs and activates it when the pel-use-hide-lines user-option is turned on (set to **t**). ⚠ there is no indication in the mode line when lines are hidden, so you must be careful. One way to show is to activate the line numbering by using the `<f11> l l` key sequence to toggle the display of the line number.s See Σ **Display - Lines** for more info.		
Hide lines matching (or not matching) specified regexp	• `<f11> M-/ h` • `C-c /`	(**hide-lines** &optional ARG)	Hide lines matching the specified regexp. • With prefix arg of 4 (**C-u**) hide lines that do not match the specified regexp. • With any other prefix arg, reveal all hidden lines. ☝ By default this calls **hides-lines-matching** without prefix argument and **hide-lines-not-matching** when the command is issued with a prefix argument. If you set the **hide-lines-reverse-prefix** user-option this behaves the other way around. Use `<f11> M-/ <f3> 2` to gain access to the customize group.
Hide lines matching specified regexp	`<f11> M-/ M-h`	(**hide-lines-matching** SEARCH-TEXT)	Hide lines matching the specified regexp.
Hide lines NOT matching specified regexp	`<f11> M-/ M-o`	(**hide-lines-not-matching** SEARCH-TEXT)	Hide lines that don't match the specified regexp.
Show all hidden lines	`<f11> M-/ M-s`	(**hide-lines-show-all**)	Show all areas hidden by the filter-buffer command.
Hide block of lines: between specified start and end lines	`<f11> M-/ b`	(**hide-blocks** &optional ARG)	Hide blocks of lines between matching regexps. • With prefix ARG of 4 (**C-u**) hide blocks that do not match the specified regexps. • With any other prefix arg, reveal all hidden blocks.
Hide block of lines: between specified start and end lines	`<f11> M-/ M-b`	(**hide-blocks-matching** START-TEXT END-TEXT)	Hide text that is between lines matching START-TEXT and END-TEXT.
Hide text not in block of lines: between specified start and end lines	`<f11> M-/ M-p`	(**hide-blocks-not-matching** START-TEXT END-TEXT	Hide text that is not between lines matching START-TEXT and END-TEXT.
Kill (or delete) hidden lines ⚠ Use with care!	`M-x hide-lines-kill-hidden`	(**hide-lines-kill-hidden** &optional DELETEP)	Kill all hidden areas. • If called with prefix arg (or DELETEP is non-nil) don't save the text to the kill ring (this is faster, but you can't retrieve the hidden text. ⚠ Use with care!
Indented Text Folding	The following command folds (hide or show) all lines that are indented more than the current line.		
Toggle hiding lines more indented than current line	`<f11> M-/ M-/`	(**pel-toggle-hide-indent**)	Toggle hiding lines more indented than current line. • Affects the entire buffer. Not syntax sensitive. Can be used anywhere. ⚠ Do not modify the buffer while lines are hidden, it's allowed but its using selective display and you don't see what you change.
Fold Code with origami-mode See also: Σ **Abbreviations**	📦 The **origami** package provides flexible code folding. PEL activates it when the **pel-use-origami** or the **pel-use-erlang-ls** user-option is turned on. • It provides support for several programming languages, with some explicitly supported by PEL: **C**, **C++**, **Clojure**, **Emacs Lisp**, **Go**, Java, Javascript, PHP, Perl, **Python**. It's possible to submit support for others. Some package do that with the help of LSP. It's the case for **Erlang**. • See **origami-arser-alist** user-option. • The origami code folding can close (fold, contract) or open (expand) a code block. For the moment PEL installs **my fork of the project**, until some fixes and enhancements are merged inside the main project. ⌨ PEL key bindings, with integration of other mode: • PEL includes keys to hide/show comments and docstrings inside the origami key-map when the corresponding packages are activated by PEL user-options: `M-/ M-;` , `M-/ M-d`, `M-/ M-D` to toggle hiding of all comments, docstring and all docstrings respectively. • PEL provides the key bindings described below, re-using the `M-/` key as a prefix, overriding the global binding to hippie-expand while origami-mode is active. Most key bindings should be easy to type quickly. ⚠ While **origami-mode** is active the **M-/** key binding to hippie-expand is not available: it is bound to **M-/ M-/** instead. See Σ **Abbreviations**.		
Toggle origami-mode	`<f11> M-/ o`	(**origami-mode** &optional ARG)	Toggle Origami mode: minor mode to selectively hide/show text in the current buffer. • With a prefix argument ARG, enable the mode if ARG is positive, and disable it otherwise.
Toggle global origami-mode	`<f11> M-/ O`	(**global-origami-mode** &optional ARG)	Toggle Origami mode in all buffers. • With prefix ARG, enable Global Origami mode if ARG is positive; otherwise, disable it
Open a fold node	`M-/ M-o`	(**origami-open-node** BUFFER POINT)	Open the fold node at POINT in BUFFER. • The fold node opened will be the deepest nested at POINT.
Open a fold node and all of its children	`M-/ O`	(**origami-open-node-recursively** BUFFER POINT)	Open the fold node and all of its children at POINT in BUFFER. • The fold node opened will be the deepest nested at POINT.
Open a fold node recursively to make code at point visible	`M-/ M-s`	(**origami-show-node** BUFFER POINT)	Like origami-open-node but also opens parent fold nodes recursively so as to ensure the position where point is is visible.
Close a fold node	`M-/ M-c`	(**origami-close-node** BUFFER POINT)	Close the fold node at POINT in BUFFER. The fold node closed will be the deepest nested at POINT.
Close a fold node and all of its children	`M-/ C`	(**origami-close-node-recursively** BUFFER POINT)	Close the fold node and all of its children at POINT in BUFFER. • The fold node closed will be the deepest nested at POINT.
Toggle open or closed a fold node	`M-/ M-t`	(**origami-toggle-node** BUFFER POINT)	Toggle the fold node at POINT in BUFFER open or closed. • The fold node opened or closed will be the deepest nested at POINT.
Search forward on this line for a node and toggle it open or closed	`M-/ M->`	(**origami-forward-toggle-node** BUFFER POINT)	Like 'origami-toggle-node' but search forward in BUFFER for a fold node. If a fold node is found after POINT and before the next line break, this will be toggled. Otherwise, behave exactly as 'origami-toggle-node'. ☝ This makes toggling nodes much more convenient.
Cycle a fold between open, recursively open, closed	`M-/ TAB`	(**origami-recursively-toggle-node** BUFFER POINT)	Cycle a fold node between recursively closed, open and recursively open depending on its current state. • The fold node acted upon is searched for forward in BUFFER from POINT. • If a fold node is found after POINT and before the next line break, this will be toggled otherwise the fold node nested deepest at POINT will be acted upon. This command will only work if bound to a key. For those familiar with org-mode heading opening and collapsing, this will feel familiar. It's easiest to grasp this just by giving it a go.
Open every fold in the buffer	`M-/ M-O`	(**origami-open-all-nodes** BUFFER)	Recursively open every fold node in BUFFER.
Close every fold in the buffer	`M-/ M-C`	(**origami-close-all-nodes** BUFFER)	Recursively close every fold node in BUFFER.
Toggle open/closed every fold node in the buffer	`M-/ M-T`	(**origami-toggle-all-nodes** BUFFER)	Toggle all fold nodes in the buffer recursively open or recursively closed.

Operation	Keystroke	Function	Note
Close everything but the folds necessary to see the point	`M-/ M-.`	(**origami-show-only-node** BUFFER POINT)	Close all fold nodes in BUFFER except for those necessary to make POINT visible. Very useful for quickly collapsing everything in the buffer other than what you are looking at. ☝Very useful for concentrating on an area of code.
Move to the previous fold	`M-/ M-p`	(**origami-previous-fold** BUFFER POINT)	Move point to the beginning of the fold before POINT. If POINT is in a fold, move to the beginning of the fold that POINT is in.
Move to the end of the next fold	`M-/ M-n`	(**origami-next-fold** BUFFER POINT)	Move point to the end of the fold after POINT. If POINT is in a fold, move to the end of the fold that POINT is in.
Move to the start of the next fold	`M-/ f`	(**origami-forward-fold** BUFFER POINT)	Move point to the beginning of the first fold in the BUFFER after POINT.
Move to the start of the next fold that is a sibling of the current fold	`M-/ M-f`	(**origami-forward-fold-same-level** BUFFER POINT)	Move point to the beginning of the next fold in the buffer that is a sibling of the fold the point is currently in.
Move to the start of the previous fold that is a sibling of the current fold	`M-/ M-b`	(**origami-backward-fold-same-level** BUFFER POINT)	Move point to the beginning of the previous fold in the buffer that is a sibling of the fold the point is currently in.
Undo the last folding operation	`M-/ M-u`	(**origami-undo** BUFFER)	Undo the last folding operation applied to BUFFER. Undo history is linear. If you undo some fold operations and then perform a new fold operation you will lose the history of operations undone.
Redo the last undone folding operation	`M-/ M-U`	(**origami-redo** BUFFER)	Redo the last folding operation applied to BUFFER. You can only redo undone operations while a new folding operation hasn't been performed to BUFFER.
Remove all folds from the buffer and reset all origami state	`M-/ R`	(**origami-reset** BUFFER)	Remove all folds from BUFFER and reset all origami state associated with this buffer. Useful during development or if you uncover any bugs. ☝Useful if origami messes up!

Hide/Show Code Blocks — References

Topic & Link	Description
GNU Emacs Manual - Hideshow minor mode	Emacs section that describes the Hide/Show minor mode.
GNU Emacs Manual - Selective Display	Description of the selective display feature.