CWL - Common Workflow Language - Markup Support

Operation	<u>Keystroke</u>	Function	<u>Note</u>		
Editing CWL files See also: YAML	help manage YAML files als • Aside from the first 2 k	o help manage CWL files. ey bindings listed to access help	cwl-mode provides a major mode for CWL files. The minor modes and commands that and customization buffers for YAML, the key bindings listed in this page and their related		
Gee also. <u>G TAINL</u>	commands are also described in other PEL PDF pages. The links to these pages are on the first column. The <u>cwl-mode</u> external package provides a major mode support for CWL. PEL provides access to it when the <u>pel-use-cwl</u> user-option is turned on (set to t). PEL associates the following file extensions with <u>cwl-mode</u> : .cwl.				
Open this PDF file. See also: <u>Nelp/Info</u>		(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the MCWL 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 useroption is set it's the other way around.		
<u>∑ Customize</u> PEL CWL control	<f11> SPC M-c <f2> <f12> <f2></f2></f12></f2></f11>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL CWL support. • If OTHER-WINDOW is non-nil (use C-u), display in other window.		
<u>∑ Customize</u> Emacs CWL control	<f11> SPC M-c <f3> <f12> <f3></f3></f12></f3></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs CWL support groups: yaml, fly check, indent-tools and smartparens • If OTHER-WINDOW is non-nil (use C-u), display in other window.		
Flycheck See also: SyntaxCheck	Flycheck is a minor mode for on-the-fly syntax checking. The flycheck external package is activated by PEL when the pel-use-flycheck user-option is turned on or another activated PEL user-option requires it.				
		Aside from the following 2 key bindings that PEL provides to toggle the flycheck mode, flycheck key prefix is C-c : as set by its flycheck-keymap-prefix user-option. You can change it for a different key prefix.			
Toggle flycheck mode for current buffer	<f11> ! !</f11>	(flycheck-mode & optional ARG)	Toggle flycheck minor-mode for the current buffer.		
Toggle flycheck mode for all buffers	<f11> ! M-!</f11>	(global-flycheck-mode &optional ARG)	Toggle Flycheck mode in all buffers. • Flycheck mode is enabled in all buffers where 'flycheck-mode-on-safe' would do it.		
Flycheck buffer/file					
Syntax Check current buffer	C-c ! c	(flycheck-buffer)	Start checking syntax in the current buffer. • Get a syntax checker for the current buffer with 'flycheck-get-checker-for-buffer', and start it.		
Check syntax of current file	C-c ! C-c	(flycheck-compile CHECKER)	 Run CHECKER via 'compile'. CHECKER must be a valid syntax checker. Interactively, prompt for a syntax checker to run. Instead of highlighting errors in the buffer, this command pops up a separate buffer with the entire output of the syntax checker tool, just like 'compile'. 		
Highlight current column	The following command provide a vertical line across the entire window at the cursor location. Useful when creating tables or checking indentation manually. vline also provides the vline-global-mode to activate the vertical line in all buffers; PEL has no binding for it because it slows Emacs too much.				
Toggle Vline Mode See also: • ∑ Highlight • ∑ Hide/Show	• <f11> h • <f11> 9</f11></f11>	(vline-mode &optional ARG)	Toggle the display of a vertical line spanning the entire window at the cursor column. Requires: vline.el PEL activates it when pel-use-vline user option is t.		
Indented Text Folding	The following command folds (hide or show) all lines that are indented more than the current line. • You can also use the f key inside the indent-tools Hydra, shown below, to fold indented sections.				
Toggle hiding lines more indented than current line See also: Hide/Show	<f11> M-/ M-/</f11>	(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 shape.		
Indent-tools	display and you don't see what you change. The indent-tools external package provides several commands to indent, un-indent and navigate across indented text levels. It provides a minor mode and a key hydra that provides all of these commands. The indent-tools external package PEL activates it when the pel-use-indent-tools user-option is turned on (set to t). This also automatically activates the hydra external package. PEL provide a global key binding to its key hydra and provides the ability to activate the proposed key binding globally and for python mode: pel-indent-tools-key-bound: activates the C-c > key binding either globally or for python-mode only.				
Open the indent-tools hydra	<f11> <tab> ></tab></f11>	(indent-tools-hydra/body)	Activate the e body in the "indent-tools-hydra" hydra.		
See also: Indentation The indent-tools hydra provide	C-c >		 With PEL, this key binding is only available when: globally, when pel-indent-tools-key-bound is set to globally, in python-mode only when pel-indent-tools-key-bound is set to python. The actual key is selected by indent-tools indent-tools-keymap-prefix user-option, the default is C-c > 		
keys you can use to navigate across the indented CWL elements.	The heads for the associated hydra are: >: 'indent-tools-indent', <: 'indent-tools-demote', E: 'indent-tools-indent-end-of-defun', c: 'indent-tools-comment', U: 'indent-tools-indent-paragraph', 1: 'indent-tools-indent-paragraph', 1: 'indent-tools-kill-tree', C: 'indent-tools-copy-hydra/body', s: 'indent-tools-goto-end-of-tree', u: 'indent-tools-goto-parent', d: 'indent-tools-goto-child', S: 'indent-tools-goto-end-of-tree', n: 'indent-tools-goto-next-sibling', p: 'indent-tools-goto-previous-sibling', i: 'helm-imenu', j: 'forward-line', k: 'previous-line', SPC: 'indent-tools-indent-space', -: 'undo-tree-undo', L: 'recenter-top-bottom',		-UUU:F1 somedata.yml All (1,0) (YAML WK Fly Anzu) Indent Navigation Actions		
See also: <u>∑ Hide/Show</u>	f: 'yafolding-togg' q: exit		The f key toggles the element folding. Press once to hide the sub-tree, press-again to display it back.		

Operation	<u>Keystroke</u>	Function	<u>Note</u>		
Smartparens Mode • Smartparens manual See also: Inserting Text	Simplify insertion of matching pairs with the smartparens minor mode. PEL binds a set of keys, described below, to toggle activation of that mode. This uses the smartparens external package. PEL activates it when pel-use-smartparens is set to t. **Mode line lighter: **smartparens-mode: SP				
200 also: <u>//ss</u>	• smartparens-strict-mode: SP/s				
Help on smartparens	<f11> i (?</f11>	(sp-cheat-sheet &optional ARG)	Generate a cheat sheet of all the smartparens interactive functions. Shows inside Emacs buffer. • Without a prefix argument, print only the short documentation and examples. • With non-nil prefix argument ARG, show the full documentation for each function. • You can follow the links to the function or variable help page. • To get back to the full list, use M-x help-go-back. • You can use 'beginning-of-defun' and 'end-of-defun' to jump to the previous/next entry. • Examples are fontified using the 'font-lock-string-face' for better orientation.		
Toggle smartparens mode	<f11> i ((</f11>	(smartparens-mode &optional ARG)	Toggle smartparens mode.		
Toggle smartparens-strict mode	<f11> i ()</f11>	(smartparens-strict-mode &optional ARG)	Toggle the strict smartparens mode. • When strict mode is active, 'delete-char', 'kill-word' and their backward variants will skip over the pair delimiters in order to keep the structure always valid (the same way as 'paredit-mode' does). This is accomplished by remapping them to 'sp-delete-char' and 'sp-kill-word'. There is also function 'sp-kill-symbol' that deletes symbols instead of words, otherwise working exactly the same (it is not bound to any key by default). • When strict mode is active, this is indicated with "/s" after the smartparens indicator in the mode list		
Toggle smartparens mode	<f11> i (M-(</f11>	(smartparens-global-mode &optional ARG)	Toggle Smartparens mode in all buffers. With prefix ARG, enable Smartparens-Global mode if ARG is positive; otherwise, disable it. Smartparens mode is enabled in all buffers where 'turn-on-smartparens-mode' would do it.		
Toggle smartparens-strict mode	<f11> i (M-)</f11>	(smartparens-global-strict-mode &optional ARG)	Toggle Smartparens-Strict mode in all buffers. With prefix ARG, enable Smartparens-Global-Strict mode if ARG is positive; otherwise, disable it. Smartparens-Strict mode is enabled in all buffers where 'turn-on-smartparens-strict-mode' would do it.		
Smart-shift See also: ∑ Indentation	The <u>smart-shift</u> external package simplifies shifting a complete line or region of lines right or left but also up or down. It is implemented as a minor or global minor mode that must be enabled first. You can identify the smart-shift-mode inside one of the pel- <mode>-activates-minor-modes user-options to activate it automatically. You can also use the commands manually or through the key bindings provided by PEL to activate the smart-shift-mode in the current buffer or globally for all buffers. PEL controls it through customization user-options: The <u>smart-shift</u> external package PEL activates it when the pel-use-smart-shift user-option is turned on (set to t). PEL also provides the <u>pel-smart-shift-keybinding</u> user-option that allows you to select additional alternative key bindings for the smart-shift commands that shift line(s). By default the key bindings are using C-c as a key prefix. With PEL you can also use a control key for the cursor or change the prefix key to use the <f9> key. The 3 possible key bindings are shown below but only one of them will be available at any given time. The one available is the one selected by the user-option value.</f9></mode>				
Toggle smart-shift mode in current buffer	<f11> <tab> s</tab></f11>	(smart-shift-mode &optional ARG)	Activate/de-activate the smart-shift mode in the current buffer. Activate the line-shift key bindings listed below, in the current buffer. With PEL, the actual key binding selected for the line shift commands depend on the value of the pel-smart-shift-keybinding user-option.		
Toggle smart-shift mode globally	<f11> <tab> S</tab></f11>	(global-smart-shift-mode &optional ARG)	Toggle Smart-Shift mode in all buffers. With prefix ARG, enable Global Smart-Shift mode if ARG is positive; otherwise, disable it. Smart-Shift mode is enabled in all buffers where 'smart-shift-mode-on' would do it.		
Shift line or region right	• C-c <right> • C-c <c-right> • <f9> <right></right></f9></c-right></right>	(smart-shift-right &optional ARG)	Shift the line or region to the ARG times to the right. With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.		
Shift line or region left	• C-c <left> • C-c <c-left> • <f9> <left></left></f9></c-left></left>	(smart-shift-left &optional ARG)	Shift the line or region to the ARG times to the left. With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.		
Shift line or region up	• C-c <up> • C-c <c-up> • <f9> <up></up></f9></c-up></up>	(smart-shift-up &optional ARG)	Shift the line or region to the ARG times to the upwards. With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.		
Shift line or region down	• C-c <down> • C-c <c-down> • <f9> <down></down></f9></c-down></down>	(smart-shift-down &optional ARG)	Shift the line or region to the ARG times to the downwards With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.		

CWL & Emacs — References

Description & URL	Notes
Common Workflow Language	Common Workflow Language (CWL) uses a <u>subset of YAML</u> and provides YAML supporting tools.
	CWL home page CWL User Guide CWL YAML Guide