↖ Topic Index

Auto-Completion Support

Description	Keystroke	Function	Note
Auto Completion 🚧 See also: ∑ **Abbreviations**	🚧 Work on auto-completion is in very early stage. This table currently only contains basic information. More information will become available with the integration of the support for multiple programming languages and other completion sources. The information will also include info on Tags. ☝ More on abbreviation completion is available in the ∑ **Abbreviations** table. Both abbreviation completion and one auto-completion mechanism can be used at the same time (using different keys if any), in some case the abbreviation choices can also be available via auto-completion. • Emacs comes with a basic completion system, accessible via the (**completion-at-point**) command bound to **C-M-i** . • At least 2 other external packages provide pop-up menu completion: auto-complete and company-mode. PEL supports both and prevents activation of both of them in one buffer. See below.		
Open this PDF file. See also: ∑ **Help/Info**	**<f11> , <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the ∑ **Auto-Completion** 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 auto-completion support. See also:∑ **Customize**	**<f11> , <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Open the PEL customize group(s) for the current context: auto-completion support. Use this to open to change PEL user option variables the activate and control the various Apple script features such as the name of the narrator voice. • When a prefix argument (like **C-u**) opens the buffer inside another window.
Customize Emacs built-in auto-completion support See also:∑ **Customize**	**<f11> , <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs auto-completion group which includes: auto-complete, company, hippie-expand. • When a prefix argument (like **C-u**) opens the buffer inside another window. • Group belonging to files that have not yet been loaded are normally not accessible in Emacs and via the customize-group command. PEL, however, attempts to locate the file that defines a non-loaded customization group and will prompt you for loading the file if it finds it.
Display Auto-completion status	**<f11> , ?**	(**pel-completion-info** &optional APPEND)	Print information about available auto-completion. info in a *pel-autocomplete-info* **help-mode** buffer. Shows which one is enabled via customization and their current activation state. • Prints current state and values of relevant user-options as buttons you can use to get more info and change their customized values. • Clear previous buffer content by default. Use prefix arg (like **C-u**) to append instead.
Emacs Built-in Completion	Emacs built-in completion is provided by the completion-at-point command, described below.		
Symbol Completion at point See also: ∑ **Xref**	• **C-M-i** • **<Esc> <tab>** • **M-<tab>**	(**completion-at-point**)	Perform completion on the text around point. • The completion method is determined by '**completion-at-point-functions**'. • The tags-completion-at-point-function is used for Emacs Lisp code by default. It provides a list of possible values in the *Completions* buffer. ☝ **C-M-i** is also used for Flyspell, which can be used to spell check only moments and strings. See the specific programming language tables for more information.
Symbol Completion at point • **and language specific completion**	**<f6> 6**	(**complete-symbol** ARG)	Perform completion of the text around point, using the method identified by the variable **completion-at-point-functions**, acting as the **completion-at-point** command above. • With a prefix argument, such as **C-u <f6> 6**, this command does completion within the collection of symbols listed in the index of the manual for the language you are using.
Complete language specific symbol at point	**<f6> 7**	(**info-complete-symbol** &optional MODE)	Perform completion of symbol at point using mode-specific language items. • For example, inside a Emacs-Lisp buffer it finds the Emacs-Lisp functions, variables names.
Completion function - using tags Candidate for: completion-at-point-functions		(**tags-completion-at-point-function**)	Using tags, return a completion table for the text around point. If no tags table is loaded, do nothing and return nil. This uses the tag facility.

PEL controlled completion activation	PEL provides logic to dynamically activate either auto-complete-mode or company-mode for one buffer or all of them, globally through the commands accessible via the **<f11> ,** prefix.
	• ☑ You must first select witch one should be available via PEL customization:
	  • Set **pel-use-auto-complete** to t to enable the ability to use auto-complete-mode.
	  • Set **pel-use-company** user option to t to enable the ability to use company-mode.
	• When **pel-init** is executed, the commands made available depend on the section made by customization of PEL but also of the two modes. Then the corresponding commands listed in the sections below are available.
	
	🚧 PEL's auto-completion support implementation is not yet completed in this early version of PEL. For now just standard hooks are setup, but not for all programming languages that is planned for PEL support. Upcoming versions of PEL will integrate the configuration of autocompletion for more programming languages.		

Explicitly List Completion Candidates with the currently active auto completion system	• **<f11> , ,** • **M-1**	(**pel-complete**)	List completion candidates. • Force auto-completion of text at point, don't wait for timeout, using the currently active auto-completion system (either auto-complete-mode or company-mode). • There must be at least 1 character preceding point. • If no auto completion system is active in the current buffer, the command issues an error. ☛✂ **M-1** default binding is to downcase-word. PEL rebinds this key company-mode is activated via customization and company-mode is active. ☛☝ With PEL, the **M-1** key is close to the **M-/** key, bound to the command used for abbreviation expansions. It becomes easy to use either.
Completion Menu keys • **Auto-completion Menu Operations** • **Company-Mode Menu Operations** See also: ∑ **Scrolling**	When an completion pop-up menu generated **either** by auto-complete or company-mode is shown, you can use the following keys for operating on that menu: • **M-n** : next candidate (or <down> cursor) • **M-p** : previous candidate (or <up> cursor) • **M-1, M-2, M-3,** etc…: select candidate by line number • **<tab>** : complete using 1 candidate (if 1 choice), using the prefix part among many candidates, or cycle through all candidates. • **<DEL>** : Delete 1 char of the current candidate prefix • **<RET>** : Select current candidate, execute action for candidate if any (eg. when template selection used) • **C-?** : Show candidate help in separate buffer • **<f1>** : Show candidate help in separate buffer. ☝This is **very** handy to quickly review documentation of several symbols! • **C-M-v** : Scroll help buffer forward (note: see the ∑ **Scrolling** table for more info on scrolling) • **Esc <PgDown>** : Scroll help buffer forward • **C-M-S-v** : Scroll help buffer backward • **Esc <Pg-up>** : Scroll help buffer backward • **C-g** : Stop completion		

Description	Keystroke	Function	Note

auto-complete	Auto-Complete is one of the auto completion package for Emacs supported by PEL.
	This requires the auto-complete package.
	• PEL provides basic configuration and installation logic: if the **pel-use-auto-complete** customization variable is set to **t**, the PEL is able to install auto-complete from the MELPA archive, and provides the basic configuration: you do not have to modify your Emacs init file. Once activated by customization with PEL you can then activate it globally and/or control whether it is available for the current buffer, using the following commands.
	• The **pel-init** function will install Auto-Complete from the MELPA archive if it is not already present. You may want to use another version (such as the one from MELPA stable). Just install it before customizing to use it and executing **pel-init**.
	• This is an early version of PEL. Future versions of PEL will integrate logic to support use of Auto-Complete for more programming languages and systems (like templating package). For now PEL only incorporates the basic configuration of Auto-Complete provided by its **ac-config-default** function.
	• Auto-complete provides the following customizable variables (and several others):
	  • **ac-use-quick-help** : set to **t** to activate a quick pop-up help display that shows right beside the menu choice.
	  • **ac-quick-help-delay** : delay before the quick help pops up. Default is 1.5 seconds.
	
	PEL provides access to the auto-complete commands via the commands below.		

Toggle Auto-Complete mode for current buffer	**<f11> , a**	(**pel-auto-complete-mode** &optional ARG)	Toggle Auto-Complete mode in current buffer. • With prefix ARG, enable buffers' Auto-Complete mode if ARG is positive, otherwise de-activate it. • Does not allow activation if company-mode is active. • If Global Auto-Complete is on, you can turn it off for one buffer using this command. This command calls **auto-complete-mode** when appropriate.
Toggle Global Auto-Complete mode	**<f11> , A**	(**pel-global-auto-complete-mode** &optional ARG)	Toggle Global Auto-Complete mode. • With prefix ARG, enable Global Auto-Complete mode if ARG is positive, otherwise de-activate it. • Does not allow activation if company-mode is active. • The **global-auto-complete-mode** variable is customizable. If you set its customized value to **t**, then **pel-init** will automatically activate it. You will still be able to turn it off later in an Emacs session using this command and without having to change the customization value. This command calls **global-auto-complete-mode** when appropriate.
company-mode	Company-Mode is the other auto completion package supported by PEL. This requires the **company-mode** external package. PEL provides basic configuration and installation logic: if the **pel-use-company** customization variable is set to **t**, the PEL is able to install Company Mode from the MELPA archive, and provides the basic configuration: you do not have to modify your Emacs init file. Once activated by customization with PEL you can then activate it globally and/or control whether it is available for the current buffer, using the following commands. • The **pel-init** function will install Complete Mode from the MELPA archive if it is not already present. You may want to use another version (such as the one from MELPA stable). Just install it before customizing to use it and executing **pel-init**. • This is an early version of PEL. Future versions of PEL will integrate logic to support use of Company Mode for more programming languages and systems (like templating package). For now PEL only incorporates the basic configuration of Company Mode. PEL provides access to the company-mode commands via the commands below.		
Toggle Company mode for current buffer	**<f11> , c**	(**pel-company-mode** &optional ARG)	Toggle Company Mode mode in current buffer. • With prefix ARG, enable buffers' Company Mode if ARG is positive, otherwise de-activate it. • Does not allow activation if auto-complete-mode is active. • If Global Company Mode is on, you can turn it off for one buffer using this command. • "complete anything"; is an in-buffer completion framework. Completion starts automatically, depending on the values 'company-idle-delay' and 'company-minimum-prefix-length'. • Completion can be controlled with the commands: 'company-complete-common', 'company-complete-selection', 'company-complete', 'company-select-next', 'company-select-previous'. If these commands are called before 'company-idle-delay', completion will also start. • Completions can be searched with 'company-search-candidates' or 'company-filter-candidates'. These can be used while completion is inactive, as well. The completion data is retrieved using 'company-backends' and displayed using 'company-frontends'. If you want to start a specific backend, call it interactively or use 'company-begin-backend'. This command calls **company-mode** when appropriate.
Toggle Global Company mode	**<f11> , C**	(**pel-global-company-mode** &optional ARG)	Toggle Global Company mode. • With prefix ARG, enable Global Company mode if ARG is positive, otherwise de-activate it. • Does not allow activation if auto-complete-mode is active. • The **global-company-mode** variable is customizable. If you set its customized value to **t**, then **pel-init** will automatically activate it. You will still be able to turn it off later in an Emacs session using this command and without having to change the customization value. This command calls **global-company-mode** when appropriate.

Auto-completion — References

Document	Note
Basic Auto Completion	
GNU Emacs Manual - Completion for Symbol Names	
Auto Completion with Auto-Complete	

Document	Note
Auto Complete @ MELPA	You can get auto-complete from MELPA. An interesting point of this page lists the other packages that need auto-complete. There's over 45 packages that use it for various programming languages and environments.
Auto Complete @ GitHub	Auto complete source code
Auto Complete Manual @ Github	Covers installation, check, features, concepts, configuration, advanced usage. Reading required for users.
Using Emacs: 8 - Auto-complete @ Youtube	Mike Zamansky video that covers abbreviation and auto-complete. Duration: 5 minutes.
Using Emacs: 45- Company or Autocomplete @ Youtube	Another video from Mike Zamansky that covers both auto-complete and company-mode. Duration: 13 minutes.
Auto Completion with Company-mode	
company-mode ; Modular in-buffer completion framework for Emacs	Text completion framework for emacs
Using digits to select company-mode candidates @ (or emacs irrelevant)	