Emacs support for the Go Programming Language

Description	Keystroke	Function	Note
Go programming Language Support	Support for the **Go programming language** is described in this page. Go support requires the **go-mode** external package. PEL supports it when the **pel-use-go-mode** user option is turned on (set to **t**). This activates the following: • Files with the .go extensions are recognized as Go source files and use the go-mode major mode, • Speedbar support for .go files listing functions and types, • Automatic execution of gofmt when saving a buffer into a file, • Generic programming language features like template text insertion handle Go comment style. See ∑ **Inserting Text** ., • Control of the tab width for all go files, via the **pel-go-tab-width** user-option (access the PEL customer buffer with the **<f11> SCP g <f2>** key sequence or **<f12> <f2>** from inside a buffer visiting a Go source code file.		
Open this PDF file. See also: ∑ **Help/ Info**	**<f11> SPC g <f1>** **<f12> <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the local copy of the **Go** PDF file unless a command prefix (like **C-u**) was used. In that case it opens the Github-hosted file instead.
∑ **Customize** PEL Forth support	**<f11> SPC g <f2>** **<f12> <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Go support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
∑ **Customize** Emacs Erlang support	**<f11> SPC g <f3>** **<f12> <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs Go support: go, go-cover, godoc, go-dot-mod. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Set tab width for current buffer	**<f11> SPC g M-t** **<f12> M-t**	(**pel-go-set-tab-width** N)	Change the tab width used in current buffer. • The change is temporary and affects the current buffer only. • To change the tab width used for all Go source code files, change the 'pel-go-tag-width' user-option variable instead.
Toggle gofmt run on file save	**<f11> SPC g M-s** **<f12> M-s**	(**pel-go-setup-info**)	Display Go setup information: • tab width • whether gofmt is executed before saving buffer.
Add new import package to list of module package import statement	**C-c C-a**	(**go-import-add** ARG IMPORT)	Add a new IMPORT to the list of imports. Don't move point. • When called with a prefix ARG asks for an alternative name to import the package as. • If no list exists yet, one will be created if possible. • If an identical import has been commented, it will be uncommented, otherwise a new import will be added.
Describe expression at point.	**C-c C-d**	(**godef-describe** POINT)	Describe the expression at POINT. This uses the **gofef executable**, a Go program. • To install it, run the following command from a shell: `go get github.com/rogpeppe/godef`. • The GOPATH environment variable must be setup and GOPATH/bin must be in the PATH to be able to run godef.
Move to expression definition	**C-c C-j**	(**godef-jump** POINT &optional OTHER-WINDOW)	Jump to the definition of the expression at POINT. • after that command, use **M-,** to go back to original point.
Move to expression definition in other window	**C-x 4 C-c C-j**	(**godef-jump-other-window** POINT)	Jump to the definition of the expression at POINT but into the other window. • after that command, use **M-,** to go back to original point.
Move to current function arguments	**C-c C-f a**	(**go-goto-arguments** &optional ARG)	Go to the arguments of the current function. • If ARG is non-nil, anonymous functions are skipped.
Move to current function docstring	**C-c C-f d**	(**go-goto-docstring** &optional ARG)	Go to the top of the docstring of the current function. • If there is none, add one beginning with the name of the current function. • Anonymous functions do not have docstrings, so when this is called interactively anonymous functions will be skipped. If called programmatically, an error is raised unless ARG is non-nil.
Move to function definition	**C-c C-f f**	(**go-goto-function** &optional ARG)	Go to the function definition (named or anonymous) surrounding point. • If we are on a docstring, follow the docstring down. • If no function is found, assume that we are at the top of a file and search forward instead. • If point is looking at the func keyword of an anonymous function, go to the surrounding function. • If ARG is non-nil, anonymous functions are ignored.

Move to imports statement	**C-c C-f i**	(**go-goto-imports**)	Move point to the block of imports.
			• If using
			`import (`
			`  "foo"`
			`  "bar"`
			`)`
			it will move point directly behind the last import.
			• If using
			`import "foo"`
			`import "bar"`
			it will move point to the next line after the last import.
			• If no imports can be found, point will be moved after the package declaration.

Move to current method receiver	**C-c C-f m**	(**go-goto-method-receiver** &optional ARG)	Go to the receiver of the current method. • If there is none, add parenthesis to add one. • Anonymous functions cannot have method receivers, so when this is called interactively anonymous functions will be skipped. If called programmatically, an error is raised unless ARG is non-nil.
Move to current function name	**C-c C-f n**	(**go-goto-function-name** &optional ARG)	Go to the name of the current function. • If the function is a test, place point after 'Test'. • If the function is anonymous, place point on the 'func' keyword. • If ARG is non-nil, anonymous functions are skipped.
Move to current function return value declaration	**C-c C-f r**	(**go-goto-return-values** &optional ARG)	Go to the return value declaration of the current function. • If there are multiple ones contained in a parenthesis, enter the parenthesis. • If there is none, make space for one to be added. • If ARG is non-nil, anonymous functions are skipped.
	C-M-q	(prog-indent-sexp &optional DEFUN)	Indent the expression after point. When interactively called with prefix, indent the enclosing defun instead.

Go— References

Document	Notes

Document	Notes
The Go Programming Language - Wikipedia	