

GNU Emacs Reference Card

Buffers

select another buffer	C-x b
list all buffers	C-x C-b
kill a buffer	C-x k

Transposing

transpose characters	C-t
transpose words	M-t
transpose lines	C-x C-t
transpose sexps	C-M-t

Spelling Check

check spelling of current word	M-\$
check spelling of all words in region	M-x ispell-region
check spelling of entire buffer	M-x ispell-buffer
toggle on-the-fly spell checking	M-x flyspell-mode

Tags

find a tag (a definition)	M-.
find next occurrence of tag	C-u M-.
specify a new tags file	M-x visit-tags-table
regexp search on all files in tags table	M-x tags-search
run query-replace on all the files	M-x tags-query-replace
continue last tags search or query-replace	M-,

Shells

execute a shell command	M-!
execute a shell command asynchronously	M-&
run a shell command on the region	M-
filter region through a shell command	C-u M-
start a shell in window *shell*	M-x shell

Rectangles

copy rectangle to register	C-x r r
kill rectangle	C-x r k
yank rectangle	C-x r y
open rectangle, shifting text right	C-x r o
blank out rectangle	C-x r c
prefix each line with a string	C-x r t

Abbrevs

add global abbrev	C-x a g
add mode-local abbrev	C-x a l
add global expansion for this abbrev	C-x a i g
add mode-local expansion for this abbrev	C-x a i l
explicitly expand abbrev	C-x a e
expand previous word dynamically	M-/

Miscellaneous

numeric argument	C-u <i>num</i>
negative argument	M--
quoted insert	C-q <i>char</i>

Regular Expressions

any single character except a newline	.	(dot)
zero or more repeats	*	
one or more repeats	+	
zero or one repeat	?	
quote special characters	\	
quote regular expression special character <i>c</i>	\c	
alternative (“or”)		
grouping	\(... \)	
shy grouping	\(?: ... \)	
explicit numbered grouping	\(:NUM ... \)	
same text as <i>n</i> th group	\n	
at word break	\b	
not at word break	\B	

entity	match start	match end
line	^	\$
word	\<	\>
symbol	_<	_>
buffer	\‘	\’
class of characters	match these	match others
explicit set	[...]	[^ ...]
word-syntax character	\w	\W
character with syntax <i>c</i>	\sc	\Sc
character with category <i>c</i>	\cc	\Cc

International Character Sets

specify principal language	C-x RET l
show all input methods	M-x list-input-methods
enable or disable input method	C-\
set coding system for next command	C-x RET c
show all coding systems	M-x list-coding-systems
choose preferred coding system	M-x prefer-coding-system

Info

enter the Info documentation reader	C-h i
find specified function or variable in Info	C-h S

Moving within a node:

scroll forward	SPC
scroll reverse	DEL
beginning of node	b

Moving between nodes:

next node	n
previous node	p
move up	u
select menu item by name	m
select <i>n</i> th menu item by number (1–9)	n
follow cross reference (return with 1)	f
return to last node you saw	l
return to directory node	d
go to top node of Info file	t
go to any node by name	g

Other:

run Info tutorial	h
look up a subject in the indices	i
search nodes for regexp	s
quit Info	q

Registers

save region in register	C-x r s
insert register contents into buffer	C-x r i
save value of point in register	C-x r SPC
jump to point saved in register	C-x r j

Keyboard Macros

start defining a keyboard macro	C-x (
end keyboard macro definition	C-x)
execute last-defined keyboard macro	C-x e
append to last keyboard macro	C-u C-x (
name last keyboard macro	M-x name-last-kbd-macro
insert Lisp definition in buffer	M-x insert-kbd-macro

Commands Dealing with Emacs Lisp

eval sexp before point	C-x C-e
eval current defun	C-M-x
eval region	M-x eval-region
read and eval minibuffer	M-:
load a Lisp library from load-path	M-x load-library

Simple Customization

customize variables and faces	M-x customize
-------------------------------	---------------

Making global key bindings in Emacs Lisp (example):

```
(global-set-key (kbd "C-c g") 'search-forward)
(global-set-key (kbd "M-#") 'query-replace-regexp)
```

Writing Commands

```
(defun command-name (args)
  "documentation" (interactive "template")
  body)
```

An example:

```
(defun this-line-to-top-of-window (line)
  "Reposition current line to top of window.
With prefix argument LINE, put point on LINE."
  (interactive "P")
  (recenter (if (null line)
                0
                (prefix-numeric-value line))))
```

The **interactive** spec says how to read arguments interactively. Type **C-h f interactive RET** for more details.

Copyright © 2015 Free Software Foundation, Inc.
For GNU Emacs version 24
Designed by Stephen Gildea

Permission is granted to make and distribute modified or unmodified copies of this card provided the copyright notice and this permission notice are preserved on all copies.

For copies of the GNU Emacs manual, see:

<http://www.gnu.org/software/emacs/#Manuals>