

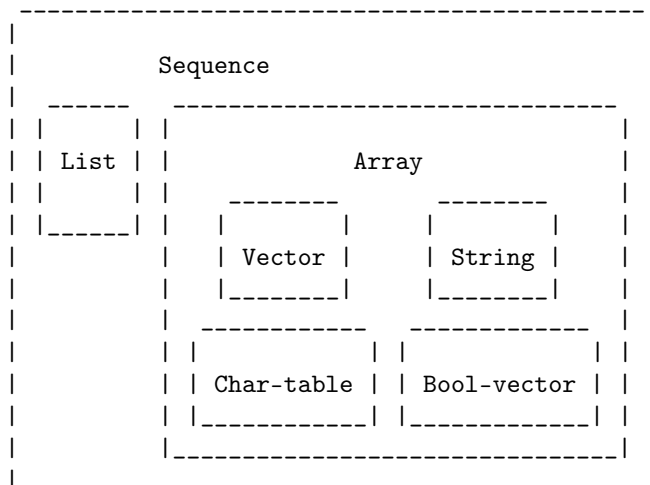
# 1 Emacs CheatSheet

## LANGUAGES

- PDF Link: [cheatsheet-emacs-A4.pdf](#), Category: languages
- Blog URL: <https://cheatsheet.dennyzhang.com/cheatsheet-emacs-A4>
- Related posts: Vim CheatSheet, [#denny-cheatsheets](#)

File me Issues or star this repo.

See more CheatSheets from Denny: [#denny-cheatsheets](#)



## 1.1 Features

### 1.1.1 View In Emacs

Name	Comment
Move forward across one balanced expression	<code>forward-sexp</code> C-M-f
Move backward across one balanced expression	<code>backward-sexp</code> C-M-b

### 1.1.2 Org-mode export Latex

Name	Comment
Make page size bigger	<code>#+LATEX_HEADER: \usepackage[margin=0.5in]{geometry}</code> Link: <a href="#">stackexchange</a>
Change document class	<code>#+LaTeX_CLASS_OPTIONS: [a4paper]</code> Link: <a href="#">stackexchange</a>
Add the date of today	<code>#+LATEX_HEADER: \rhead{Updated: \today}</code> Link: <a href="#">stackexchange</a>
Add page number with total pages	<code>#+LATEX_HEADER: \rfoot{\thepage\ of \pageref{LastPage}}</code>

### 1.1.3 Buffer Operations

Name	Comment
Move to top	<code>(goto-char (point-min))</code>
Replace string by regexp	<code>buffer-replace.el</code>
Delete region	<code>(delete-region start-pos end-pos)</code>
Buffer string with plain text	<code>(buffer-substring-no-properties start-pos end-pos)</code>
	<code>(get-buffer-create BUFFER-OR-NAME)</code>
	<code>(current-buffer)</code>
	<code>(set-buffer BUFFER-OR-NAME)</code>
	<code>(kill-buffer)</code>
	<code>(set-buffer-modified-p nil)</code>

### 1.1.4 GNUS - Mail In Emacs

Name	Comment
Create delayed email	<code>gnus-delay-article C-c C-j</code>
Save mail's attachment	<code>gnus-summary-save-parts</code>
Forward mail	<code>gnus-summary-mail-forward</code>
Send gnus drafts	<code>gnus-draft-send-message</code>
Send all the sendable drafts	<code>gnus-draft-send-all-messages</code>
Add attachment	<code>mml-attach-file(C-c C-m f)</code>
Create group	<code>gnus-group-make-group (G m)</code>

### 1.1.5 Table in Org-mode

```
#+NAME: supplies
| Date          | Category          | Amount |
|-----+-----+-----|
| 2014/01/14 | Supplies          | 43.97 |
| 2014/02/15 | Supplies          | 56.48 |
| 2014/02/11 | Book              | 17.99 |
| 2014/06/10 | Kinesis Keyboard | 289.16 |
| 2014/08/23 | Printer           | 99.96 |
| 2014/08/30 | Supplies          | 58.26 |
| 2014/08/22 | Books             | 18.99 |
| 2014/08/25 | Books             | 7.50 |
| 2014/09/15 | Books             | 21.49 |
| 2014/12/31 | Toner Service     | :=24.95*4 |
|-----+-----+-----|
|              | Total:           |        |
#+TBLFM: @>$3=vsum(@2..@-1);%.2f
```

## 1.2 Data Structures

### 1.2.1 Debug

Name	Comment
Debug a function	<code>edebug-defun</code>
Change function via advice	<code>defadvice ;; Super inspiring feature!</code>
Set default value	<code>(setq-default indent-tabs-mode nil)</code>

### 1.2.2 String

Name	Comment
string1 contains string2	<code>(string-match ".*README.org" buffer-file-truename)</code>
Replace by regexp	<code>(setq ret (replace-regexp-in-string "&lt;hr/&gt;" "" ret))</code>
Format string	<code>(format "%s/%s" mywordpress-server-url blog-uri)</code>
String replace	<code>(replace-string from-string to-string &amp;optional start end)</code>
Replace by regexp	<code>(replace-regexp REGEXP TO-STRING &amp;optional DELIMITED START END)</code>
replace-match	<code>(while (search-forward-regexp "myRegexPattern" nil t) (replace-match " " t))</code>
The second captured string	<code>(match-string 2)</code>
Get the position of the 2nd captured string	<code>(match-beginning 2) (match-end 2)</code>
List matched count	<code>(setq myStr (replace-regexp-in-string "myRegex1" "myRep1" myStr)) (count myStr "myRep1")</code>
Grab the start and end positions of a word	<code>(setq myBoundaries (bounds-of-thing-at-point 'word))</code> <code>(setq myStr (buffer-substring myStartPos myEndPos))</code> <code>(setq myStr (buffer-substring-no-properties myStartPos myEndPos))</code>

### 1.2.3 Regex

Name	Comment
Regex In Emacs	<code>regexp-string-match.el</code>
Change a given string using regex	<code>(replace-regexp-in-string "^ +" "" url)</code>
Seach regexp in some string	<code>(string-match myRegex myStr)</code>
Get captured match	<code>(match-string 1 myStr)</code>
Escape special characters	<code>(regexp-quote "^")</code> <code>(regexp-opt '("hello" "world"))=</code>

### 1.2.4 Integer

Name	Comment
String to int	<code>(string-to-number STRING &amp;optional BASE)</code>
Check whether it's int	<code>(integerp 23)</code>
decimal to hex	<code>(format "%x" 10)</code>
hex to decimal	<code>(format "%d" #xa)</code>

### 1.2.5 Array & List

Name	Comment
Get the first element	<code>(car mylist)</code>
Get the nth element	<code>(nth n mylist)</code>
Get the last element	<code>(car (last mylist))</code>
Get the 2nd to the last elements	<code>(cdr mylist)</code>
Get the nth to the last elements	<code>(nthcdr n mylist)</code>
Similar to (car (car value))	<code>(caar value)</code>
Similar to (cdr (car value))	<code>(cdar value)</code>
Return the cdr of the cdr of X.	<code>(cddr X)</code>

### 1.2.6 Array & List - More

Name	Comment
Create a list	<code>(defvar my-list (list "item1, item2"))</code>
Add item to list	<code>(add-to-list 'my-list "item3")</code>
Head of a list	<code>(car '(a b c))</code>
Tail of a list	<code>(cdr '(a b c))</code>
Loop a list	<code>(dolist (item my-list) (message item))</code>
Concat two lists	<code>(nconc '("a" "b" "c") '("d" "e" "f")) link</code>
Return a newly created list	<code>(list x)</code>
Append x to the head of a list	<code>(cons x mylist)</code>
Append without duplication	<code>(add-to-list 'auto-mode-alist '("\\.gp\$" . gnuplot-mode))</code>
Add ELEMENT if missing	<code>(add-to-list LIST-VAR ELEMENT &amp;optional APPEND COMPARE-FN)</code>

### 1.2.7 Position

Name	Comment
Return character at position	<code>(char-after (point))</code>
Return character preceding position	<code>(char-before (point))</code> <code>(setq myStr (thing-at-point 'word))</code> <code>(setq myStr (thing-at-point 'symbol))</code> <code>(setq myStr (thing-at-point 'line))</code>

### 1.2.8 Insert text

Name	Comment
Insert string	<code>(insert "hello world")</code> <code>(insert-buffer-substring buffer &amp;optional start end)</code> <code>(insert-buffer-substring-no-properties buffer &amp;optional start end)</code> <code>(insert-file-contents myPath)</code> <code>(insert-file-contents-literally filename &amp;optional visit beg end replace)</code>

### 1.2.9 Delete text

Name	Comment
	<code>(delete-char 9)</code>
	<code>(delete-region myStartPos myEndPos)</code>
	<code>(erase-buffer)</code>
	<code>(upcase obj)</code>
	<code>(upcase-word n)</code>
	<code>(upcase-region beg end)</code>
	<code>(upcase-initials obj)</code>
	<code>(upcase-initials-region beg end)</code>
	<code>(capitalize obj)</code>
	<code>(capitalize-word n)</code>
	<code>(capitalize-region beg end)</code>
	<code>(downcase)</code>
	<code>(downcase-word n)</code>
	<code>(downcase-region beg end)</code>

### 1.2.10 DateTime

Name	Comment
Convert time to string	<code>(format-time-string "&lt;%Y-%m-%d %H:%M UTC +8&gt;" (current-time))</code>
Get current time	<code>(current-time)</code>
Add some offset for a time	<code>(time-add time (seconds-to-time seconds))</code>
Subtract two time values	<code>(time-subtract after-init-time before-init-time)</code>
Get second count	<code>(float-time (time-subtract after-init-time before-init-time))</code>
Return date as a list (mm/dd/yyyy)	<code>calendar-current-date</code>
	<code>(calendar-extract-month date)</code>
m1 will be changed	<code>(calendar-increment-month m1 y1 -1)</code>
	<code>(calendar-date-compare '((12 27 2012)) '((12 26 2012)))</code>
	<code>(calendar-holiday-list)</code>

### 1.2.11 Hook

Name	Comment
Add hook	<code>(add-hook 'myhook '(lambda () (insert "fun1 was called ")))</code>
Run each hook in myhook	<code>(run-hooks 'myhook)</code>

### 1.2.12 Files

Name	Comment
Open file	<code>(find-file html-file)</code>
Save file	<code>(write-file html-file nil)</code>
Get short filename	<code>(file-name-nondirectory somefilename)</code>
Get the directory name from filename	<code>(file-name-directory FILENAME)</code>
Check file/directories existence	<code>(file-exists-p bfilename)</code>
Insert contents of file FILENAME after point	<code>(insert-file-contents somefilename)</code>
Return FILENAME's final "extension"	<code>(file-name-extension "test.erl")</code>
Return FILENAME sans final "extension"	<code>(file-name-sans-extension "test.erl")</code>
Return a list of names of files in DIRECTORY	<code>(directory-files DIRECTORY &amp;optional FULL MATCH NOSORT)</code>
Insert contents of file FILENAME after point	<code>(insert-file-contents FILENAME &amp;optional VISIT BEG END REPLACE)</code>
Confirm directory exists	<code>(file-directory-p FILENAME)</code>
Create directory	<code>(make-directory "~/emacs.d/autosaves/" t)</code>
Find files by name	<code>(find-dired "../" "-name defined.hrl")</code>
read file content into a string	<code>(setq dddstring (with-temp-buffer (insert-file-contents "dd.txt"))=</code>

## 1.3 More Resources

License: Code is licensed under MIT License.