

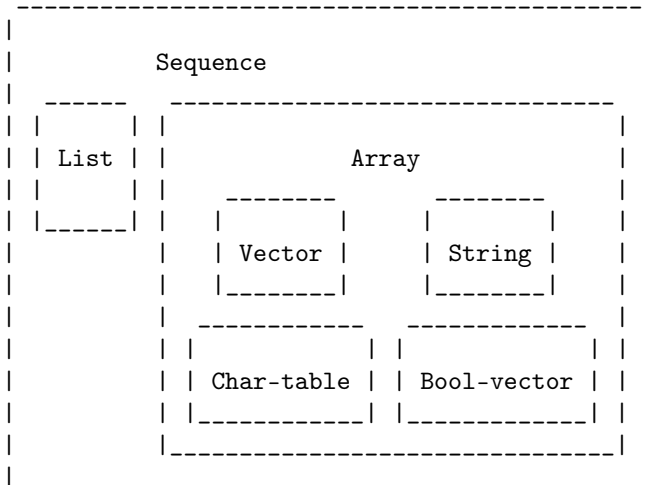
1 Emacs CheatSheet

LANGUAGES

- PDF Link: [cheatsheet-emacs-A4.pdf](#)
- Blog URL: <https://cheatsheet.dennyzhang.com/cheatsheet-emacs-A4>
- Category: languages

File me Issues or star this repo.

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



1.1 Data Structures

1.1.1 Debug

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

1.1.2 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 "<hr/>" "" ret))</code>
Format string	<code>(format "%s/%s" mywordpress-server-url blog-uri)</code>
String replace	<code>(replace-string from-string to-string &optional start end)</code>
Replace by regexp	<code>(replace-regexp REGEXP TO-STRING &optional DELIMITED START END)</code>
replace-match	<code>(while (search-forward-regexp "myRegexPattern" nil t) (replace-mat</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))</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>

1.1.3 Regexp

Name	Comment
Regexp In Emacs	<code>regexp-string-match.el</code>
Change a given string using regex	<code>(replace-regexp-in-string "^ +" "" url)</code>
Search 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>

1.1.4 Integer

Name	Comment
String to int	<code>(string-to-number STRING &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>

1.1.5 Array & List

- Get items from 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 <code>(car (car value))</code>	<code>(caar value)</code>
Similar to <code>(cdr (car value))</code>	<code>(cdar value)</code>
Return the cdr of the cdr of X.	<code>(cddr X)</code>

- More about list

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 &optional APPEND COMPARE-FN)</code>

1.1.6 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>

- Insert text

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

- Delete text

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

1.1.7 DateTime

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

1.1.8 Hook

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

1.1.9 Files

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

1.2 Features

1.2.1 View In Emacs

Name	Comment
Move forward across one balanced expression	forward-sexp C-M-f
Move backward across one balanced expression	backward-sexp C-M-b

1.2.2 Buffer Operations

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

1.2.3 GNUS - Mail In Emacs

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

1.3 Common Scripts

- emacs multiline regexp

```
;; http://stackoverflow.com/questions/1309050/emacs-query-replace-regexp-multiline
(setq content-str "hello
this
----
Denny
Sophia")

(message (replace-regexp-in-string
  "\n----\\(\\.\\|\\|\\n\\)*" "" content-str))
```

- cond: like switch or case

link: [WhenToUseIf](#)

```
(let ((x 1))
  (cond ((eq x 0) "It's zero")
        ((eq x 1) "It's one")
        :else "It's something else")
)
```

1.4 More Resources

License: Code is licensed under MIT License.