

## Easily Making CheatSheets with Org-mode

### —Reference Sheet for “What I’m Currently Learning”—

#### Hello, World!

**Pretty PDF** Enter `M-x compile` to produce a nice looking PDF of your reference sheet.  
 ◇ I’ve bound this command to `C-c C-m` in [my Emacs setup](#) ;-)  
 My Emacs configuration also documents how I utilise ‘minted’ to obtain colourful source code blocks.

**Section Headers** A usual Org header, say `* my section`, results in the boxed headers used in this cheat sheet.

**Parallel Environments** The  $\text{\LaTeX}$  matter here supports an org-environment named `parallel` for producing text side-by-side.  
 The column break is automatic, but as this is sugar for a `minipage` containing a `multicolumn` we can force a column separation with `\columnbreak`: This command, in Org, necessities newlines between the items being separated.

To learn more, manipulating this source is the way to go!

Also, opening this file produces a `README.md` ;-)  
 Which can then be regenerated on-demand with `f11`.

#### CheatSheet Examples

Reference sheets created from this project include:

**ElispCheatSheet** Quick reference to the core language of Emacs —Editor MACroS.

**Islam** Important figures in the faith.

**PrologCheatSheet** Program where everything is a relation —i.e., a database table.

**CatsCheatSheet** Listing of common theorems in elementary category theory.

**LatticesCheatSheet** Reference sheet for definitions and results in Lattice Theory.

**OCamlCheatSheet** Basics of OCaml, “the best imperative language”.

**CoqCheatSheet** Reference sheet for the Coq language.

**GojuRyuCheatSheet** A quick cheat sheet for common terms in Goju Ryu Karate —the hard-soft style of karate.

*If you use this org-setup to produce a neat cheat sheet, please let me know!*

#### Why Learn & Relearn?

*The world of ideas is not revealed to us in one stroke; we must both permanently and unceasingly recreate it in our consciousness. —Rene Thom*

#### Getting Started

The steps to utilising this git project for your own cheat sheet may be:

1. Go to the repo where you want to make a cheat sheet.
2. Add this project as a submodule then copy its core to where you’re working:

```
git submodule add https://github.com/alhassy/CheatSheet.git
; cp CheatSheet/CheatSheet.org .
; cp CheatSheet/README.org .
```

3. Open `CheatSheet.org` and locate `#+INCLUDE: CheatSheetSetup.org` then rewrite `CheatSheetSetup.org` → `CheatSheet/CheatSheetSetup.org`.

I don’t think this is difficult to automate, so I will likely get to doing it. Indeed, just copy-paste the following into, say the `*scratch*` buffer then `C-x C-e` after the final closing parenthesis.

```
(let ((your-repo "~/example") ;; Alter this location!
      (enable-local-variables :all))
  ;; Look at my “local variables” below; ensure nothing malicious.
  ;; So no need to be queried about loading them.
```

```
;; Obtain the submodule then make a /copy/ of this cheatsheet.
(eshell-command (concat
  " cd " your-repo
  "; git submodule add https://github.com/alhassy/CheatSheet.git"
  "; cp CheatSheet/CheatSheet.org ."
))
```

```
;; Make your cheat sheet refer to the submodule’s setup file.
(find-file-other-window (concat your-repo "/CheatSheet.org"))
(beginning-of-buffer)
(re-search-forward "INCLUDE: CheatSheetSetup.org" nil t)
(replace-match "INCLUDE: CheatSheet/CheatSheetSetup.org")
(beginning-of-buffer)
)
```

```
;; To remove a submodule:
;; git submodule deinit path_to_submodule ; git rm path_to_submodule
```

For the `README.md` to be generated as desired, fill in the macros `URL` and `blurb` at the top of this org file to point to your repository and provide a description of what the cheatsheet serves to accomplish.

#### What if it’s not good enough?

*“The person who thinks of doing something, is usually passed by the person doing it.”*

*The more that you read, the more things you will know. The more that you learn, the more places you’ll go. —Dr. Seuss*

## Org-mode Basics

- Read [Org-mode for beginners](#) for a refresher!  
◊ For more see [The Compact Org-mode Guide](#).

**Reloading** To reload a file with updated org settings, press `C-c C-c` on a settings line –i.e., one beginning with a `#+`, to reset the temporary file cache.

**Inclusion** During export, you can include the content of another file.

- ◊ Syntax: `#+INCLUDE: "<fileName>" [(markup) [(language)]]`
  - `markup ::= src | example`
  - `language ::= C | haskell | emacs-lisp | ...`
  - If the markup is not given, the text will be assumed to be in Org mode format and will be processed normally; c.f., [Setup files](#).
- ◊ To visit the file, `C-c '` while the cursor is on the line with the file name.
- ◊ Include only portions of a file by appending with `:lines "x-y"` where `x` is the first line and `y` is the second-to-last line. Also `"-y"` for upto but not including line `y`, and `"x-"` for taking line `x` until the end of the file.

## Basic Equational Support

Basic name-formula equational support. `\eqn{name}{formula}` yields a displayed equation with `formula` left aligned and `name` right aligned:

$$\begin{array}{l} \text{formula} \\ F(f_0 \circ \dots \circ f_{n-1}) = F f_0 \circ \dots \circ F f_{n-1} \end{array} \qquad \begin{array}{l} \text{NAME} \\ \text{FUNCTORIALITY} \end{array}$$

Moreover, we can refer to such a formula by invoking `\ref{name}` –e.g., `FUNCTORIALITY` and `NAME`. However, if `name` involves unicode symbols, then this may cause problems.

See the [CatsCheatSheet](#) for examples of this kind.

## What if I want 3 columns?

At the top, say after the `#+INCLUDE: CheatSheet/CheatSheetSetup.org` line, add a new section:

```
* begin multicols :ignore:
#+latex: \begin{multicols}{3}
```

Then at the very bottom, add a section to close this multicol:

```
* end multicols :ignore:
#+latex: \end{multicols}
```

Having three narrow columns is useful for term-heavy or formula heavy sheets.

## Parallel Environment

Cheat sheets should not waste space, so the setup provides a `parallel` L<sup>A</sup>T<sub>E</sub>X enviornment that takes an optional parameter indicating how many columns are desired —two by default. Importantly, we use this environment as if it were any normal org-block:

```
.,#
#+begin_parallel org
???content here???
#+end_parallel
```

The initial new line is important, otherwise the parallel environment occurs inline, which may not be the intended behaviour.

Below we demonstrate that [loops implement finite quantifications](#) by showing how the specification of a loop is implemented, unsurprisingly, using a loop. I tend to use a lot of unicode.

A finite quantification can be defined axiomatically by the empty-range rule and split-off term rules. Together these form a recursive definition which can be phrased as a loop.

```
// For ⊕_ : T → T → T,
// fold(A,a,b) ≈ (⊕ x : a..b-1 • A[x])
/*@ axiomatic Fold {
  @
  @ logic T
  @ fold{L}(T *A, ℤ a, ℤ b)
  @ reads a,b,A, A[.] ;
  @
  @ axiom foldEmptyRange{L} :
  @   ∀ T *A, ℤ a, b; a ≥ b
  @   ⇒ fold(A,a,b) == identity(⊕);
  @
  @ axiom foldSplitOffTerm{L} :
  @   ∀ T *A, ℤ a, b; a ≤ b
  @   ⇒ fold(A, a, b+1)
  @       == fold(A, a, b ) ⊕ A[b];
  @ }
  @*/

/* requires \valid(A+(0..N-1));
   @ assigns \nothing;
   @ ensures \result == fold(A,0,N);
   @*/
T fold(int N, T* A) {
  T total = identity(⊕);

  /* loop invariant 0 ≤ n ≤ N;
   @ loop invariant total == fold(A,0,n);
   @ loop assigns n, total;
   @ loop variant N-n;
   */
  for(int n = 0; n != N; n++)
    total = total ⊕ A[n];
  return total;
}
```

This pseudo-code is reified by giving concrete values for  $(T, \oplus, \text{identity})$  such as  $(\text{int}, +, 0)$  or  $(\text{bool}, ||, \text{false})$ . Any monoid will do.

`parallelNB` produces a side-by-side rendition with ‘N’o ‘B’ar:

left		right
left		right
left		right

Here is an example with four columns:

left	middle	middle	right
left	middle	middle	right
left	middle	middle	right

Here is an example with three columns and ‘n’o ‘b’ar:

left	middle	right
left	middle	right
left	middle	right