# Easily Making CheatSheets with Org-mode

## —Reference Sheet for "What I'm Currently Learning"—

### Project Goal

*Use the elegant & intuitive Org-mode syntax to produce exquisite reference sheets.*

- For example, the boxed section headers here are produced from usual Org headers, as in `* my section`; and one may use org-ref for citations, as in Basic Equational Support

Read Org-mode for beginners for a refresher!

- For more see The Compact Org-mode Guide.

Execute `C-c C-e l o` or `M-x compile` to produce a nice looking PDF of your reference sheet.

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

### 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

I think org-mode is ideal for computing cheat sheets especially since it allows us to use org-babel tangle to have small minimal working examples that go along with the ideas.

> *'Why', said the Dodo, 'the best way to explain it is to do it.'*
> —Alice's Adventures in Wonderland

*A good stock of examples, as large as possible, is indispensable for a thorough understanding of any concept, and when I want to learn something new, I make it my first job to build one.* —Paul Halmos

### Getting Started

To use this project for your own cheatsheets, 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.

Keep your submodule up to date by running the following command from the parent project —i.e., your project.

```
git submodule update
```

**Alternatively**:

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` to `CheatSheet/CheatSheetSetup.org`.

### 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

## What if I want N columns? Or non-landscape? Or multiple formats?

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

```
#+LATEX_HEADER: \def\cheatsheetcols{N}
#+LATEX_HEADER: \landscapefalse
```

For example, having three narrow columns is useful for term-heavy or formula heavy sheets. In contrast, dense sheets may appear less daunting when rendered as single-column in portrait. Sometimes a double-column portrait is more appropriate.

Press `C-c C-c` on the following incantation to produce a single column portrait of the cheat sheet.

```
(with-temp-buffer
    (insert
    "#+EXPORT_FILE_NAME: CheatSheet_Portrait.pdf
     #+LATEX_HEADER_EXTRA: \\landscapefalse \\def\\cheatsheetcols{1}
     #+INCLUDE: CheatSheet.org
    ")

    (let ((org-export-use-babel nil))
      (org-mode)
      (org-latex-export-to-pdf)
      )
)
```

## Colourful Source Blocks

Invoke the following with `C-c C-c`, or better yet place it in your Emacs configuration, to ensure references are picked up and source code highlighting is turned on using the Minted package —which in turn requires the pygmentize system tool.

```
(setq org-latex-listings 'minted
      org-latex-packages-alist '(("" "minted"))
      org-latex-pdf-process
      '("pdflatex -shell-escape -output-directory %o %f"
        "biber %b"
        "pdflatex -shell-escape -output-directory %o %f"
        "pdflatex -shell-escape -output-directory %o %f"))
```

For faster pdf generation, consider invoking:

```
(setq org-latex-pdf-process
      '("pdflatex -interaction nonstopmode -output-directory %o %f"))
```

By default, Org exports LaTeX using the `nonstopmode` option, which tries its best to produce a PDF —which ignores typesetting errors altogether, and therefore is not necessarily ideal when using LaTeX.

## Basic Equational Support

`\eqn{name}{formula}` yields a displayed equation with `formula` left aligned and `name` right aligned:

$$formula \qquad \text{NAME}$$

$$F(f_0 \circ \cdots \circ f_{n-1}) \;=\; F\,f_0 \circ \cdots \circ F\,f_{n-1} \qquad \text{FUNCTORIALITY}$$

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.

We may also use org-ref style references, as in `eqref:name`. However, org-ref may warn that no context for the reference is found —that's okay.

**eqref** Parenthesised reference: (NAME)

**autoref** Prefix reference with type: Equation NAME

**nameref** The name of the section that contains this reference: Basic Equational Support

## Unicode

I tend to use a lot of unicode and so this project comes with a unicode style file. We may add additional support for unicode characters as follows.

```
#+LATEX_HEADER: \newunicodechar{⊕}{\ensuremath{\oplus}}
```

Below we demonstrate that loops implement finite quantifications by showing how the specification of a loop is implemented, unsurprisingly, using a loop.

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
            ∧  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, ⊕, identity)` such as `(int, +, 0)` or `(bool, ||, false)`. Any monoid will do.

## Parallel Environment

Cheat sheets should not waste space, so the setup provides a `parallel` LaTeX 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.

The column break is automatic, but as this is sugar for a `minipage` containing a `multicolum` we can force a column separation with `\columnbreak`.

`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 |

## Subsection Support

Ideally a cheat sheet is not too hierarchical and so a subsection, as in `** child`, is turned into a rule as follows.

---

Here is the first child's content.

---

Here is the sibling's content.

## Making `README`

Evaluate the following source block with `C-c C-c` to produce a `README` file.

```
(with-temp-buffer
    (insert
    "#+EXPORT_FILE_NAME: README.org
     # HTML: <h1> Easily Making CheatSheets with Org-mode </h1>
     #+OPTIONS: toc:nil d:nil
     # Toc is displayed below at a strategic position.

     {{{blurb}}}

     :Hide:
     This project is to contain a listing of common results in X Theory.

     *The repo contains other articles I've written on X Theory;*
     *which may be read in a blog-format at:*
     https://alhassy.github.io/blog/categories/#Xtheory
     :End:

    *The listing sheet, as PDF, can be found
     [here](CheatSheet.pdf)*,
     or as a [single column portrait](CheatSheet_Portrait.pdf),
     while below is an unruly html rendition.

     This reference sheet is built around the system
     https://github.com/alhassy/CheatSheet.

     #+TOC: headlines 2
     #+INCLUDE: CheatSheet.org
    ")

    ;; No code execution on export
    ;; ⟪ For a particular block, we use ":eval never-export" ⟫
    ;;
    (let ((org-export-use-babel nil))
      (org-mode)
      ; (org-md-export-to-markdown)
      (org-org-export-to-org)
      )
)
```

Note that the `blurb` macro is defined by the user, to provide a terse description of the project.

- ◇ Think the one-line statement at the top of a github repo page.