Org-mode Tutorial / Cheat Sheet

Common tasks in org-mode

Table of Contents

Headings

It's all about headings in org. Headings such as this are created with * (stars). The number of stars defines the depth of the heading. Pressing <TAB> on headings will minimize/maximize them. Pressing tab multiple times in a row on headings with subheadings will maximize each subheading by depth corresponding to the number of times you pressed <TAB>.

Subheadings

Subheadings can nest within other headings such as this one. This is really convenient for organizing stuff.

"TODO"s (C-c C-t)

TODO's are created by starting headers with the keyword TODO or DONE. You can also create one with C-c C-t on a header. They are meant to represent tasks you wish to create within your organizational document. These can be helpful when planning out projects or assignments.

The TODO cycle

You can turn an existing header into a TODO item by cycling its TODO status. The shortcut for this is C-c C-t (t for todo).

Non-TODO -> TODO -> DONE -> Non-TODO … (with C-c C-t).

Lists

You can create bullets with the - (dash), numbered bullets or lists with #. (number[dot]). Tabbing multiple times changes the indentation of the bullet, but only if there is no text yet. Tabbing multiple levels can get confusing, but if you remember to <TAB> before typing - or #., things tend to work out.

Ordered Lists

1. item 1
2. item 2
    1. sub 1
    2. sub 2
        1. subsub 1

Unordered Lists

Start with -, +, or *.

- hello
- item 2
- item 3
    - subitem 1
    - sub
        - subsub

Description Lists

Start with -, +, or * and followed by ::

```
Matlab is a funny language.
  - Scope :: Scope doesn't work as expected, and messes everything up
    when loops mix variables up in recursive functions.
  - Namespaces :: You wish.
  - Header Files :: Nope.
```

gives

Matlab is a funny language.

Scope
: Scope doesn't work as expected, and messes everything up when loops mix variables up in recursive functions.

Namespaces
: You wish.

Header Files
: Nope.

Text Formatting

Individual words can be **bolded** with stars, *italicized* with forward slashes, and underlined with underscores. Also, there is `verbatim` and ~~strike-through~~.

```
*bold* /italics/ _underline_ ~verbatim~ +strike-through+
```

Tables

Tables can be created from any line that starts with '|' (pipe). A line that starts with '|-' (pipe dash) is considered a horizontal seperator; <TAB> after inserting it to expand the separator.

this	is	a	sample	table
move	from	one	col	to
the	other	with	<TAB>	this column

Some tips:

- <TAB> realigns the columns, and moves to the next column. If at end of line, it will create a new row.
- C-c C-c will realign the columns without moving or creating new ones.
- <RET> moves to the next row.

Table Formatting

When exporting, tables are drawn without cell borders or frames. Here are useful table formatting commands which you place immediately before a table.

This is a table with lines around and between cells

this	is	a	sample	table
move	from	one	col	to
the	other	with	<TAB>	this column

Literal Examples

Literal examples are snippets of text or source code that need to be unformatted on org export. Put this text between #+begin-example and +end_example.

Source Code

Org-mode is great for inserting snippets of code. This can be done with the

```
#+BEGIN_SRC [major-mode-name] [options]
#+END_SRC
```

[major-mode-name] adds a lot of extra functionality to these code blocks within your org file. It's also good for whoever is reading to know what the language is.

For example:

```
1: int main() {
2: return 0;
3: }
```

Useful options and commands.

Options:

- `-n` : add line numbers to the source code.
- `+n` : add line numbers but continue from last SRC snippet.
- `-r` : remove labels from source code

Commands:

- `C-c ['] ` : Allows you to edit the source code at point in its native mode. Opens a new buffer where you exit after editing by the command again.
- `C-c l` : calls org-store-link when editing with ~C-c '~. This creates a label at the line currently being edited. It can be retrieved later with `C-c C-l`.

Easy templates

It gets tiring to use the long tags if you're doing this a lot. These are shortcuts to generate the above snippet borders. To insert, type a '<' followed by a template selector and then <TAB>.

Template selectors:

```
s    #+begin_src ... #+end_src
e    #+begin_example ... #+end_example
q    #+begin_quote ... #+end_quote
v    #+begin_verse ... #+end_verse
c    #+begin_center ... #+end_center
l    #+begin_latex ... #+end_latex
L    #+latex:
h    #+begin_html ... #+end_html
H    #+html:
a    #+begin_ascii ... #+end_ascii
A    #+ascii:
i    #+index: line
I    #+include: line
```

Linking

Linking files together

You can link files together with

```
[[file:filename][name-of-link]]
```

This will preserve links after export to HTML as well.

Linking sections within a document

You can create a link to a section within your document with the section name in brackets. The second option lets you call the link something else.

```
[[section-title]]
[[section-title][link-title]]
```

Exporting (C-c C-e)

You can export org files to any of the supported formats with C-c C-e [option].

C-c C-e will show you the options available in a separate buffer. Common ones are text, html, and latex. You can get PDFs by converting to latex, but you need to have a LaTeX environment installed on your system to use this.

Export Options (C-c C-e t)

When org files are exported, certain variables are set automatically but can be manually set by you. To change a variable's value, use

```
#+VARIABLE-NAME: value
```

anywhere in your document. At the top is probably best.

To paste a template of all export options in your org document so that you can set them immediately, use (org-insert-export-options-template), or C-c C-e t.

This will print

```
#+TITLE:       the title to be shown (default is the buffer name)
#+AUTHOR:      the author (default taken from user-full-name)
#+DATE:        a date, an Org timestamp120, or a format string for format-time-stri
#+EMAIL:       his/her email address (default from user-mail-address)
#+DESCRIPTION: the page description, e.g. for the XHTML meta tag
#+KEYWORDS:    the page keywords, e.g. for the XHTML meta tag
#+LANGUAGE:    language for HTML, e.g. ‘en’ (org-export-default-language)
#+TEXT:        Some descriptive text to be inserted at the beginning.
#+TEXT:        Several lines may be given.
#+OPTIONS:     H:2 num:t toc:t \n:nil @:t ::t |:t ^:t f:t TeX:t ...
#+BIND:        lisp-var lisp-val, e.g.: org-export-latex-low-levels itemize
               You need to confirm using these, or configure org-export-allow-BIND
#+LINK_UP:     the ``up'' link of an exported page
#+LINK_HOME:   the ``home'' link of an exported page
#+LATEX_HEADER: extra line(s) for the LaTeX header, like \usepackage{xyz}
#+EXPORT_SELECT_TAGS:   Tags that select a tree for export
#+EXPORT_EXCLUDE_TAGS:  Tags that exclude a tree from export
#+XSLT:        the XSLT stylesheet used by DocBook exporter to generate FO file
```

within your document, but with your actual variable values instead of descriptions. Change any of the variables to the value of your choice. These descriptions are provided here for your reference.

The #+OPTIONS variable is very useful for specific tweaks and will effect both HTML and LaTeX exports.

```
H:         set the number of headline levels for export
num:       turn on/off section-numbers
toc:       turn on/off table of contents, or set level limit (integer)
\n:        turn on/off line-break-preservation (DOES NOT WORK)
```

```
@:         turn on/off quoted HTML tags
::         turn on/off fixed-width sections
|:         turn on/off tables
^:         turn on/off TeX-like syntax for sub- and superscripts.  If
you write "^:{}", a_{b} will be interpreted, but
the simple a_b will be left as it is.
-:         turn on/off conversion of special strings.
f:         turn on/off footnotes like this[1].
todo:      turn on/off inclusion of TODO keywords into exported text
tasks:     turn on/off inclusion of tasks (TODO items), can be nil to remove
all tasks, todo to remove DONE tasks, or list of kwds to keep
pri:       turn on/off priority cookies
tags:      turn on/off inclusion of tags, may also be not-in-toc
<:         turn on/off inclusion of any time/date stamps like DEADLINES
*:         turn on/off emphasized text (bold, italic, underlined)
TeX:       turn on/off simple TeX macros in plain text
LaTeX:     configure export of LaTeX fragments.  Default auto
skip:      turn on/off skipping the text before the first heading
author:    turn on/off inclusion of author name/email into exported file
email:     turn on/off inclusion of author email into exported file
creator:   turn on/off inclusion of creator info into exported file
timestamp: turn on/off inclusion creation time into exported file
d:         turn on/off inclusion of drawers
```

Variables are set with `nil` or `t`, and sometimes take an argument. For example

```
#+OPTIONS:     H:2 num:t toc:t \n:nil @:t ::t |:t ^:t f:t TeX:t ...
```

Title

The title is taken from the first non-comment line in the file. If there is none, ie your file starts with a heading, it uses the filename. The title can be manually set using

```
#+TITLE: This is the title of the document
```

Table of Contents

Normally inserted after the first headline, you can set manually by inserting `TABLE-OF-CONTENTS` inside square-brackets ([]) seperately on a line.

Some useful options are:

```
#+OPTIONS: toc:2          (only to two levels in TOC)
#+OPTIONS: toc:nil        (no TOC at all)
```

Including Other Files

You can include files in an org document, which will be rendered on export. This is done using the single line command

```
#+INCLUDE: "file-path" [type] [src-language]
```

type can be either (quote, example, or src). if type is src, then src-language can be any major-mode language.

HTML Export

Stylesheet

You can include a stylesheet by using

```
#+STYLE:    <link rel="stylesheet" type="text/css" href="../stylesheet.css" />
```

The styles used by org-mode to export specific parts of your document can be set in your stylesheet with the following variables:

```
p.author            author information, including email
p.date              publishing date
p.creator           creator info, about org mode version
.title              document title
.todo               TODO keywords, all not-done states
.done               the DONE keywords, all states that count as done
.WAITING            each TODO keyword also uses a class named after itself
.timestamp          timestamp
.timestamp-kwd      keyword associated with a timestamp, like SCHEDULED
.timestamp-wrapper  span around keyword plus timestamp
.tag                tag in a headline
._HOME              each tag uses itself as a class, "@" replaced by "_"
.target             target for links
.linenr             the line number in a code example
.code-highlighted   for highlighting referenced code lines
div.outline-N       div for outline level N (headline plus text))
div.outline-text-N  extra div for text at outline level N
.section-number-N   section number in headlines, different for each level
div.figure          how to format an inlined image
pre.src             formatted source code
pre.example         normal example
p.verse             verse paragraph
div.footnotes       footnote section headline
p.footnote          footnote definition paragraph, containing a footnote
.footref            a footnote reference number (always a <sup>)
.footnum            footnote number in footnote definition (always <sup>)
```

Site Navigation

You can verily easily add "Up" and "Home" buttons on each page with

```
#+LINK_UP:
#+LINK_HOME: index.html
```

Hyperlinks

Hyperlinks are managed by links and automatically converted to their html counterparts. See Linking Files.

Publishing

Publishing is simply automating your export pipeline. When you have many files that require export, it is a good idea to create a publishing script for this.

Publishing Script

To enable project publishing with org-mode, create a local file in the project directory with the instructions. In our example we have named it `org_publish.el`.

```
(defconst home (file-name-directory (or load-file-name buffer-file-name)))

(require 'org-publish)
(setq org-publish-project-alist
      `(
        ;; add all the components here
        ;; *notes* - publishes org files to html
        ("main-page"
         :base-directory ,(concat home "org/")
         :base-extension "org"  ; Filename suffix without dot
         :publishing-directory ,(concat home "html/")
         :recursive t           ; DONT include subdirectories
         :publishing-function org-publish-org-to-html
         :headline-levels 4             ; Just the default for this project.
         :style "<link rel=\"stylesheet\" type=\"text/css\" href=\"../css/styleshee
         :auto-preamble t
         :auto-sitemap t                ; generate automagically
         :sitemap-sort-folders last
         :sitemap-title "Main Sitemap"
         )

        ;; The meeting notes to be published
        ("meeting-notes"
         :base-directory ,(concat home "org/meeting-notes/")
         :base-extension "org"
         :publishing-directory ,(concat home "html/meeting-notes/")
         :recursive nil
         :publishing-function org-publish-org-to-html
         :headline-levels 4
         :style "<link rel=\"stylesheet\" type=\"text/css\" href=\"../../css/styles
         :auto-preamble t
         :auto-sitemap t
         :sitemap-title "Meeting Notes Directory"
         )

        ;; *static* - copies files to directories
        ("org-static"
         :base-directory ,(concat home "org/")
         :base-extension "css\\|js\\|png\\|jpg\\|gif\\|pdf\\|mp3\\|ogg\\|swf"
         :publishing-directory ,(concat home "html/")
         :recursive t
         :publishing-function org-publish-attachment
         )
```

```
;; *publish* with M-x org-publish-project RET emacsclub RET
("emacsclub" :components ("main-page" "meeting-notes" "org-static"))
))
```

Date: 2012-10-18 17:12:02 EDT

Author: Mishal Awadah

Org version 7.8.11 with Emacs version 24

Validate XHTML 1.0