Org mode syntax

Quick reference card

Fabrice Niessen

February 9, 2023

Contents

Framework	Org mode 9
Bug tracker	`https://github.com/fniessen/refcard-org-mode/issues`
Source	`https://github.com/fniessen/refcard-org-mode`

Summary

You will learn to:

- write your docs in Org mode
- create tables
- create custom code blocks
- and much more!

Abstract

This is an Org mode document, using the `.org` extension (supported by GitHub).

Org mode is an easy-to-write *plain text* formatting syntax for authoring notes, articles, LaTeX documents, books, Web pages, Beamer slide decks and much more!

This is a cheat sheet for **Org mode 8** (because of some markup syntax changes since Org mode 7), using ReadTheOrg CSS.

Reading through all the documentation is highly recommended, but for the truly impatient, following are some quick steps to get started.

Reference card

Contents

Document header

Title and author line:

```
#+TITLE:     Org mode syntax examples
#+AUTHOR:    Fabrice Niessen

My document provides...
```

It's a good practice to also include an email line following the author line.

```
#+EMAIL:     john.doe@example.com
```

Document settings

Document description

```
#+DESCRIPTION: This document catalogs a set of tips and tricks for composing documents
#+KEYWORDS:  org-mode, syntax, quick reference, cheat sheet, recommended practices, la
#+LANGUAGE:  en
```

Section numbering

```
#+OPTIONS:   H:4

#+OPTIONS:   num:nil
```

Table of contents

Set the `toc` attribute to activate an auto-generated table of contents (limited to its 2 first levels) at the top of document.

```
#+OPTIONS:   toc:2

#+OPTIONS:   p:t
```

The `ALT_TITLE` property allows to set an alternate title (shorter, for example) for a given headline in the table of contents and other running heads.

To locally insert the TOC at some random place, use the `#+TOC: headlines [n]` feature; for example:

```
#+TOC: headlines 2
```

List of figures

`#+TOC: figures` is not implemented yet in the HTML backend.

List of tables

`#+TOC: tables` is already implemented in the HTML backend.

List of equations

Section titles (headings)

```
* Biggest heading (level 1)

New chapter.

** Bigger heading (level 2)

New section.

*** Big heading (level 3)

New sub-section.

**** Heading (level 4)

New sub-sub-section.
```

Numbered headings

You can create numbered headings up to a certain level by setting an option:

```
#+OPTIONS: H:4
```

Paragraphs

Normal

```
A single newline has no effect.
This line is part of the same paragraph.

But an empty line

demarcates paragraphs.
```

Line breaks

```
By entering two consecutive backslashes, \\
you can force a line break
without starting a new paragraph.
```

Horizontal rules

```
For an horizontal line, insert at least 5 dashes: this is some text above an
horizontal rule
-----
and some text below it.
```

Text width

One morning, when Gregor Samsa woke from troubled dreams, he found himself transformed in his bed into a horrible vermin. He lay on his armour-like back, and if he lifted his head a little he could see his brown belly, slightly domed and divided by arches into stiff sections. The bedding was hardly able to cover it and seemed ready to slide off any moment. His many legs, pitifully thin compared with the size of the rest of him, waved about helplessly as he looked.

Formatting text

Text effects.

Bold and italic

```
/Emphasize/ (italics), *strongly* (bold), and */very strongly/* (bold italics).
```

Markup elements can be nested:

```
This is /italic text which contains _underlined text_ within it/, whereas _this is
normal underlined text_.
```

Markup can span across multiple lines, by default **no more than 2**:

```
*This
is not
bold.*
```

Org mode does not interpret a marker surrounded by alphanumeric characters as an emphasis marker. So, you can't (easily) emphasize just part of a word:

```
Not feas*ible*.
```

Monospace, superscript and subscript

Other elements to use sparingly are:

```
- monospaced typewriter font for ~inline code~
- monospaced typewriter font for =verbatim text=
- +deleted text+ (vs. _inserted text_)
- text with super^{script}, such as 2^{10}
- text with sub_{script}, such as H_{2}O
```

Smart punctuation

If the XXX option is specified, Org mode will produce typographically correct output, converting straight quotes to curly quotes, `--` to em-dashes, `--` to en-dashes, and `...` to ellipses.

Lists

Org markup allows you to create **bulleted** or **numbered** lists. It allows any combination of the two list types.

Unordered lists

Itemized lists are marked with bullets. Create them with a minus or a plus sign.

They are convenient to organize data, and make the document prettier, and easier to read.

```
- Item with some lengthy text wrapping hopefully across several lines. We add
  a few words to really show the line wrapping.
- Bullet.
  + Bullet.
    * Bullet.
```

Checklists

```
- [X] Checked.
- [-] Half-checked.
- [ ] Not checked.
- Normal list item.
```

Ordered lists

Enumerated lists are marked with numbers or letters:

```
1. Arabic (decimal) numbered list item. We add a few words to show the line
   wrapping.
   A. Upper case alpha (letter) numbered list item.
      a. Lower alpha.
      b. Lower alpha.
   B. Upper alpha.
2. Number.
```

You can have ordered lists with jumping numbers:

```
2. [@2] We start with point number 2.
3. Automatically numbered item.
```

Definition lists

Labeled, multi-line lists.

```
- First term to define ::
     Definition of the first term. We add a few words to show the line wrapping,
     to see what happens when you have long lines.

- Second term ::
     Explication of the second term with *inline markup*.

     In many paragraphs.
```

Separating lists

Adjacent lists sometimes like to fuse. To force the start of a new list, offset the two lists by an empty line comment:

```
- apples
- oranges
- bananas

# Comment.

- carrots
- tomatoes
- celery
```

Tables

Tables are one of the most refined areas of the Org mode syntax. They are very easy to create and to read.

Simple table

```
| Cell in column 1, row 1 | Cell in column 2, row 1 |
| Cell in column 1, row 2 | Cell in column 2, row 2 |
```

Org tables have cells of at most one line long: there is no such thing as a multi-line table cell in Org.

Column formatting

Columns are automatically aligned:

- Number-rich columns to the right, and
- String-rich columns to the left.

Table with aligned cells

If you want to override the automatic alignment, use `<r>`, `<c>` or `<l>`.

```
#+CAPTION: Table with aligned columns
|          <r> | <c>          | <l>          |
|            1 | 2            | 3            |
|        Right | Center       | Left         |
| xxxxxxxxxxxx | xxxxxxxxxxxx | xxxxxxxxxxxx |
```

Table with column size adjusted

Header row

You can create tables with an header row (by using an horizontal line of dashes to separate it from the rest of the table).

```
#+CAPTION: Table with an header row
| Name of column 1 | Name of column 2 | Name of column 3 |
|------------------+------------------+------------------|
| Top left         | Top middle       |                  |
|                  |                  | Right            |
| Bottom left      | Bottom middle    |                  |
```

A very long table

To test "sticky table headers"...

Name of column 1	Name of column 2	Name of column 3
Top left	Top middle	
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		Right
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
Bottom left	Bottom middle	

Table placement

```
#+ATTR_LATEX: :center nil
| a | b |
| 1 | 2 |
```

XXX Different from the following:

```
| a | b |
```

```
| 1 | 2 |
```

Align tables on the page

Left

Here is a table on the left side:

```
#+LATEX: \noindent
#+ATTR_LATEX: :center nil
| a | b | c |
|---+---+---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
#+LATEX: \hfill
```

The `noindent` just gets rid of the indentation of the first line of a paragraph which in this case is the table. The `hfill` adds infinite stretch after the table, so it pushes the table to the left.

Center

Here is a centered table:

```
| a | b | c |
|---+---+---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
```

Right

And here's a table on the right side:

```
#+LATEX: \hfill
#+ATTR_LATEX: :center nil
| a | b | c |
|---+---+---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
```

Here the `hfill` adds infinite stretch before the table, so it pushes the table to the right.

Table size

```
#+ATTR_HTML: :width 100%
| Cell in column 1, row 1 | Cell in column 2, row 1 |
| Cell in column 1, row 2 | Cell in column 2, row 2 |
```

CSV

You can fill a table from a CSV file using R commands.

Links

```
* Links
  :PROPERTIES:
  :CUSTOM_ID: links
  :END:
```

> This document is available in plain text, HTML and PDF.
> The links are delimited by double square brackets.

External links

```
See http://www.pirilampo.org (automatic!) and the
[[http://orgmode.org/][Org mode Web site]].
```

Relative links

```
[[../README.html][Home]]
```

Email links

```
[[mailto:john.doe@example.com][email John Doe]]
```

Image links

To get image links, put a link to a file in the description.

```
Clicking on the image [[http://orgmode.org/][file:images/org-mode-unicorn.png]]
leads to the Org mode home page.
```

Internal links

Inline anchors

Anchors are used to specify hypertext link targets.

```
<<anchor>> Inline anchors make arbitrary content referenceable.
```

Internal cross references

Links generally point to an headline.

```
See chapter [[#links][Links]].
```

To add a link to a figure (e.g., "See Figure 1"), just do:

```
#+name: fig
#+caption: caption
[[file:fig.png]]

See figure [[fig]].
```

You can also create a hypertext link to a document anchor in the current document *or in another document*.

```
See:
- Location [[anchor][cross reference]].
- Section [[id:0d2b0cb2-116c-4a61-a076-4c641faf4346][Internal links]]
```

Extensions that define new hyperlinks targets

Images

You can insert **image** files of different **formats** to a document:

	HTML	PDF
gif	yes	
jpeg	yes	
png	yes	
bmp	(depends on browser support)	

Inline picture

```
#+caption: Org mode logo
[[file:images/org-mode-unicorn.png]]

Click to see the [[file:images/org-mode-unicorn.png][Unicorn picture]].
```

Image alignment (using positioning)

Books usually align/float images on the right/left of the contents.

Image is left aligned

Image is right aligned

Image is centered

[̂b]

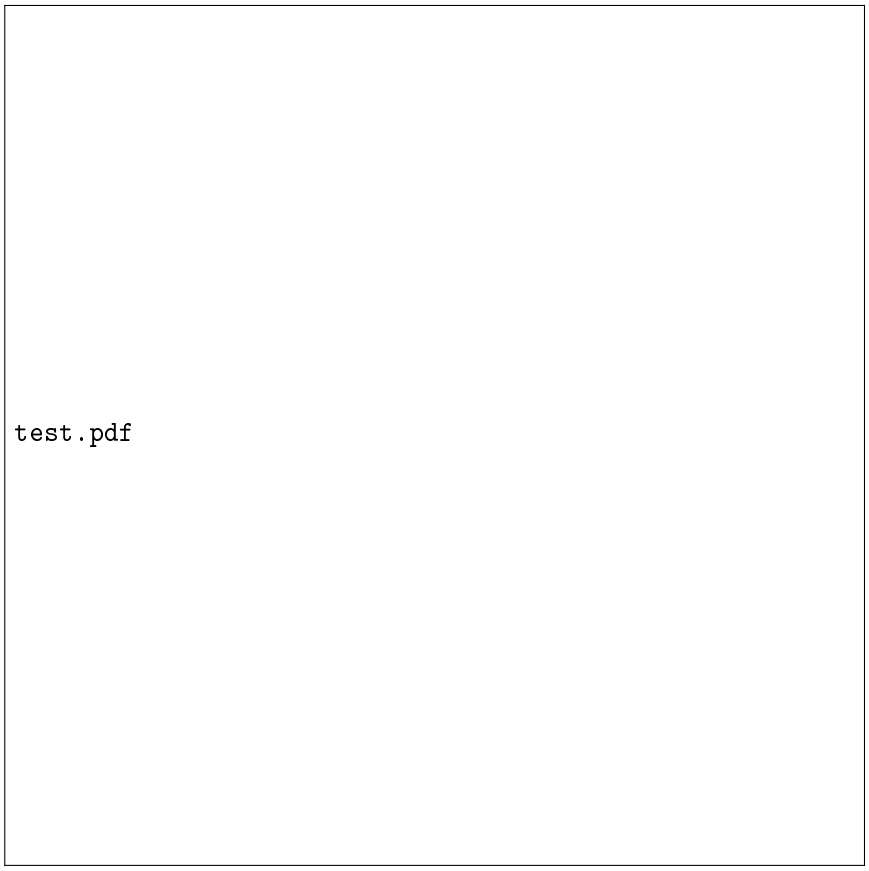


Image attributes and values

XXX Available HTML image tags include ...

Attribute	Value(s)
`:alt`	Alternate text
`:height`	
`:width`	User defined size in pixels
`:align`	
`:border`	
`:bordercolor`	
`:hspace`	
`:vspace`	
`:title`	User defined text

```
#+ATTR_LaTeX: :width 0.25\linewidth
[[file:images/org-mode-unicorn.png]]
```

Place images side by side: XXX

Figures

To define images that will be **treated as book illustrations** (figures) and automatically labeled and numbered, use XXX.

Videos

Videos can't be added directly.

Though, you can add an image with a link to the video like this:

```
[[http://www.youtube.com/watch?v=DnSGSiXYuOk][file:../bigblow.png]]
```

Admonitions

Admonitions (contextual backgrounds) are statements taken out of the content's flow and labeled with a title.

Common admonitions are:

1. `note`
2. `warning`

3. `tip`
4. `caution`
5. `important`

(Most themes style only `note` and `warning` specially.)

Base admonitions

Note

A note box is displayed as follows:

```
#+begin_note
This is a useful note.
#+end_note
```

Warning

A warning box is displayed as follows:

```
#+begin_warning
Be careful!  Check that you have...
#+end_warning
```

Tip

A tip box is displayed as follows:

```
#+begin_tip
Try doing it this way...
#+end_tip
```

Caution

```
#+begin_caution
Caution
#+end_caution
```

Important

```
#+begin_important
Important
#+end_important
```

Additional admonitions

Attention

```
#+begin_attention
Attention
#+end_attention
```

Hint

```
#+begin_hint
Hint
#+end_hint
```

Error

```
#+begin_error
Error
#+end_error
```

Danger

```
#+begin_danger
Danger
#+end_danger
```

SeeAlso (Sphinx additional)

```
#+begin_seealso
- [[http://en.wikipedia.org/wiki/Apple][Apples]] ::
     A kind of [[http://en.wikipedia.org/wiki/Fruit][fruit]].
#+end_seealso
```

Todo admonition

Simple box ("inline task"):
#+begin$_{\text{src}}$ org

- **TODO** Do this task Description of inline task.
- END #+end$_{\text{src}}$
- **TODO** Do this task Description of inline task.

- END or:

  #+begin$_{\text{src}}$ org

- WAIT [#B] Do also this other task PHONE
- END #+end$_{\text{src}}$

  Admonitiontodo

  Alternatively to the inline tasks (for creating "TODO" annotations), if you want such notes not to show up in the published version, drawers may also do the job, e.g.:

  ...

  You can then control what drawers are exported with `org-export-with-drawers` (or per document with d OPTIONS item).

Centered text

```
#+begin_left
This text is \\
aligned to the left!
#+end_left

#+begin_center
This text is \\
centered!
#+end_center

#+begin_right
This text is \\
aligned to the right!
#+end_right
```

Sidebar

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non

```
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

#+begin_sidebar
Org mode was first released by Carsten Dominik in 2004 as an outlining and
project planning tool.  Further development turned it into a general tool that
can be used to author professional documents like LaTeX.
#+end_sidebar

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac
turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor
sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies
mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien
ullamcorper pharetra. Vestibulum erat wisi...

Phasellus ut libero. Nulla in libero non enim tristique sollicitudin. Ut
tempor. Phasellus pellentesque augue eget ante. Mauris malesuada. Donec sit
amet diam sit amet dolor placerat blandit. Morbi enim purus, imperdiet in,
molestie sit amet, pellentesque eu, mauris. In vel erat vel ipsum bibendum
commodo. Curabitur accumsan. Nam sed metus. Etiam tristique bibendum justo.
```

Example

You can have `example` blocks.

```
: 10/17/97   9:04         <DIR>    bin
: 10/16/97  14:11         <DIR>    DOS
: 10/16/97  14:46         <DIR>    TEMP
: 10/16/97  14:37         <DIR>    WINNT
: 10/16/97  14:25             119  AUTOEXEC.BAT
:  2/13/94   6:21          54,619  COMMAND.COM
```

or

```
#+begin_example
10/17/97   9:04         <DIR>    bin
10/16/97  14:11         <DIR>    DOS
10/16/97  14:46         <DIR>    TEMP
10/16/97  14:37         <DIR>    WINNT
10/16/97  14:25             119  AUTOEXEC.BAT
 2/13/94   6:21          54,619  COMMAND.COM
#+end_example
```

Prose excerpts

Quote

Use the `quote` block for content that **doesn't require the preservation of line breaks**.

```
#+begin_quote
Let us change our traditional attitude to the construction of programs:
Instead of imagining that our main task is to instruct a computer what to do,
let us concentrate rather on explaining to human beings what we want a
computer to do.

The practitioner of literate programming can be regarded as an essayist, whose
main concern is with exposition and excellence of style. Such an author, with
thesaurus in hand, chooses the names of variables carefully and explains what
each variable means. He or she strives for a program that is comprehensible
because its concepts have been introduced in an order that is best for human
understanding, using a mixture of formal and informal methods that reinforce
each other.

-- Donald Knuth
#+end_quote
```

A short one:

```
#+begin_quote
Everything should be made as simple as possible,
but not any simpler. -- Albert Einstein
#+end_quote
```

Verse

In a `verse` environment, there is an **implicit line break at the end of each line**, and **indentation** is preserved:

```
#+begin_verse
Everything should be made as simple as possible,
but not any simpler. -- Albert Einstein
#+end_verse
```

Typically used for quoting passages of an email message:

```
#+begin_verse
>>  The meeting has been postponed to next Friday.
>
> Has the deadline for the report been moved too?

Yes.  And chekout http://www.doodle.com/ for rescheduling the meeting.

In the text body,
   indentation is
preserved.
#+end_verse
```

Block quote with optional attribution line

epigraph

Block quotes with their own class attribute

highlights
: pull-quote
: Blockquote

Non-breaking space

Insert the Unicode character `00A0` to add a non-breaking space.
: FIXME Or add/use an Org entity? Or use tilde?

Comments

```
It's possible to add comments in the document.

# This Org comment here won't be displayed.
```

Org doesn't support **comments inside paragraphs** since a comment ends a paragraph. However, you can mimic inline comments with export snippets, e.g., `@@comment:...@@`.

If you have tables (for example) that you want to ignore during export, one possibility is to use **comment blocks** or `:noexport:` subtrees. Another possibility is to **use** non-exported **drawers** (see #+OPTIONS: d:).

If you want to ignore that part only during export, but still want to use keep it active in the buffer, I suggest to use a drawer, with an appropriate 'org-export-with-drawers' value, e.g.,

```
#+OPTIONS: d:(not "NOEXPORT")
```

Substitutions

General replacements

```
#+MACRO: longtext this very very long text

Insert {{{longtext}}} wherever required.
```

Insert this very very long text wherever required.

Styled references

```
#+MACRO: color @@html:<span style="color: $1">$2</span>@@

{{{color(blue, This text is colored in blue.)}}}

{{{color(red, This other text is in red.)}}}
```

Find more macros on GitHub.

Special characters

We also use substitutions to include some of the widely used Unicode characters (like ©, converted from text characters to its typographically correct entity).

Accents

```
- \Agrave \Aacute
```

Punctuation

```
- Dash: \ndash \mdash
- Marks: \iexcl \iquest
- Quotations: \laquo \raquo
- Miscellaneous: \para \ordf
```

Commercial symbols

```
- Property marks: \copy \reg
- Currency: \cent \EUR \yen \pound
```

Greek characters

```
The Greek letters \alpha, \beta, and \gamma are used to denote angles.
```

Math characters

```
- Science: \pm \div
- Arrows: \to \rarr \larr \harr \rArr \lArr \hArr
- Function names: \arccos \cos
- Signs and symbols: \bull \star
```

Misc

```
- Zero-width non-joiner: \zwnj
# Smilies: \smiley \sad
- Suits: \clubs \spades
```

You can insert a real "zero-width space" Unicode character by pressing `C-x 8 RET zero width space RET` or `C-x 8 RET 200b RET`.

Source code

Inline code

```
Reference code like ~variables~ or ~functions~ inline.
```

You can also evaluate code inline as follows: 1 + 1 is `1 + 1`.

Code blocks (with syntax highlighting)

The source code blocks support syntax highlighting:

```
/*
 * Application that displays a "Hello" message to the standard output.
 */
int main(int arc, char **argv)
{
```

```
  printf("Hello, %s!\n", (argc>1) ? argv[1] : "World");
  return 0;
}
```

```
(defvar hello "Hello")

(defun hello (name &optional greeting)
  (message "%s %s" (or greeting "Hello") name))

(setq tab-width 4)
```

You need `htmlize.el` in your `load-path`, for the HTML export.

Source mode

The following language strings are currently recognized:

Awk, C, R, Calc, Clojure, CSS, Ditaa, Dot, Emacs Lisp, Forth, Fortran, Gnuplot, Haskell, Java, Javascript, LaTeX, Lilypond, Lisp, Makefile, Maxima, Matlab, Ocaml, Octave, Org, Perl, Pico Lisp, PlantUML, Python, Ruby, Sass, Scala, Scheme, Screen, Shell Script, Sql, Sqlite, Stan.

Line break

Code block with long lines:

```
testing testing testing testing testing testing testing testing testing testing
0        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456
```

For PDF (LaTeX), one solution is to surround the code block such as:

```
print("This block is in scriptsize")
```

Line numbers

Both in `example` and in `src` snippets, you can add a `-n` switch to the end of the `begin` line to get the lines numbered:

```
1  (defun org-xor (a b)
2    "Exclusive or."
```

If you use a `+n` switch, the numbering from the previous numbered snippet will be continued in the current one:

```
3  (if a (not b) b))
```

Callouts

In literal examples, Org will interpret strings like `(ref:name)` as labels, and use them as targets for special hyperlinks like `[[(name)]]` (i.e., the reference name enclosed in single parenthesis). In HTML, hovering the mouse over such a link will remote-highlight the corresponding code line, which is kind of cool.

You can also add a `-r` switch which removes the labels from the source code. With the `-n` switch, links to these references will be labeled by the line numbers from the code listing, otherwise links will use the labels with no parentheses. Here is an example:

```
1  (save-excursion                  ;
2    (goto-char (point-min)))       ;
```

In line 1, we remember the current position. Line 2 jumps to `point-min`.

Math

You can embed LaTeX math formatting in Org mode files.

Inline math expressions

For **inline math** expressions, use the parentheses notation `\(...\)`:

```
The formula \(a^2 + b^2 = c^2\) has been discovered by Pythagoras.

Let \(a=\sin(x) + \cos(x)\). Then \(a^2 = 2\sin(x)\cos(x)\) because \(\sin^2x +
\cos^2x = 1\).
```

It's *not* advised to use the `$...$` construct (both for Org and for MathJax).

Don't forget that `$` is also a valid currency symbol!

Math expressions in display mode

For mathematical expressions which you want to make **stand out, centered on their own lines**, use `\[...\]`:

```
The /Euler theorem/:
```

```
\[
\int_0^\infty e^{-x^2} dx = {{\sqrt{\pi}} \over {2}}
\]

LaTeX allows to inline such ~\[...\]~ constructs (/quadratic formula/):
\[ \frac{-b \pm \sqrt{b^2 - 4 a c}}{2a} \]
```

Double dollar signs (`$$`) should not be used.

```
\[
\left( \int_{0}^{\infty} \frac{\sin x}{\sqrt x}\,\mathrm{d}x \
right)^{2} -
\prod_{k=1}^{\infty} \frac{4k^{2}}{4k^{2}-1} +
\frac{\lambda}{2n}\sum_{k=1} ^{n} \theta_{k} ^{2} x^{n} = 0
\]
```

The equation may be wrong, but it's a nice one!

Equation numbers

Differently from `$...$` and `\(...\)`, an equation environment produces a **numbered** equation to which you can add a label and reference the equation by (label) name in other parts of the text. This is not possibly with unnumbered math environments (`$$`, ...).

```
The /Pythagoras theorem/:

#+name: pythag
\begin{equation}
a^2 + b^2 = c^2
\end{equation}

See equation [[pythag]].

# The /sinus theorem/ can be written as the equation:
#
# \begin{equation}
# \label{eqn:sinalpha}
# \frac{\sin\alpha}{a}=\frac{\sin\beta}{b}
# \end{equation}
#
# See equation [[eqn:sinalpha]].
```

Only captioned equations are numbered.
Other alternatives: use

- `\begin{equation*}` or
- `\begin{displaymath}` (= the verbose form of the `\[...\]` construct).

`M-q` does not fill those.

Miscellaneous effects

Include Org files

You can include another Org file and skip its title by using the `:lines` argument to `#+INCLUDE`:

```
#+INCLUDE: "chapter1.org" :lines "2-"
```

File inclusion, through INCLUDE keywords, is an **export-only feature**.

Raw HTML

You can include raw HTML in your Org documents and it will get kept as HTML when it's exported.

Text can be preformatted (in a fixed-width font). It is especially useful for more advanced stuff like images or tables where you need more control of the HTML options than Org mode actually gives you.

Similarly, you can incorporate JS or do anything else you can do in a Web page (such as importing a CSS file).

Native DIV blocks

You can create named classes (to get style control from your CSS) with:

```
#+begin_myclass
This text is wrapped in a myclass DIV...
#+end_myclass
```

You can also add interactive elements to the HTML such as interactive R plots.

Finally, you can include an HTML file verbatim (during export) with:

```
#+INCLUDE: "file.html" export html
```

Don't edit the exported HTML file!

Raw LaTeX

You can also use raw LaTeX. XXX

Text can be preformatted (in a fixed-width font).

Useful extensions

Todo extension

Dates

Timestamps: *[2014-01-16 dj.]* and *<2014-01-16 dj.>*.

TODO We need to achieve...

DONE Buy GTD book ONLINE

By default, `DONE` actions will be collapsed.

Note that I should probably implement that default behavior only for `ARCHIVE`'d items.

TODO Read GTD book

SCHEDULED: *<2014-09-11 dj.>*

By default, **all** (active) entries will be expanded at page load, so that their contents is visible.

That can be changed by adding such a line (into your Org document):

```
#+HTML_HEAD: <script> var HS_STARTUP_FOLDED = true; </script>
```

TODO Apply GTD methodoloy

DEADLINE: *<2014-12-01 dl.>*

This section will be collapsed when loading the page because the entry has the value `hsCollapsed` for the property `:HTML_CONTAINER_CLASS:`.

Powerful, no?

Some note COMPUTER:WRITE

You can add tags to any entry, and hightlight all entries having some specific tag by clicking on the buttons made accessible to you in the "Dashboard".

Weekly review COMPUTER

Now, you can even make your weekly review in the HTML export... Press the `r` key to start entering the "review mode" where all but one active entry are collapsed, so that you can really focus on one item at a time!

Bigblow extension

The string `fixme` (in **upper case**) gets replaced by a "Fix Me!" image:

`FIXME Delete this...`

Graphs with Graphviz

To enable the Graphviz extension, we have to add it to the extensions list in the `org-babel-load-languages` variable.

```
(add-to-list 'org-babel-load-languages '(dot . t))
(org-babel-do-load-languages 'org-babel-load-languages org-babel-load-languages)
```

It uses directly the `dot` command to process DOT language.

Undirected

```
#+begin_src dot :file images/graph.png :cmdline -Tpng
graph foo {
        "bar" -- "baz";
}
#+end_src
```

Directed

Graphs with R

The output from the **execution** of programs, scripts or commands can be inserted in the document itself, allowing you to work in the *reproducible research* mindset.

To enable the Graphviz extension, we have to add it to the extensions list in the `org-babel-load-languages` variable.

```
(add-to-list 'org-babel-load-languages '(R . t)) ; Requires R and ess-mode.
(org-babel-do-load-languages 'org-babel-load-languages org-babel-load-languages)
```

It uses directly the `R` command to process R language.

Example

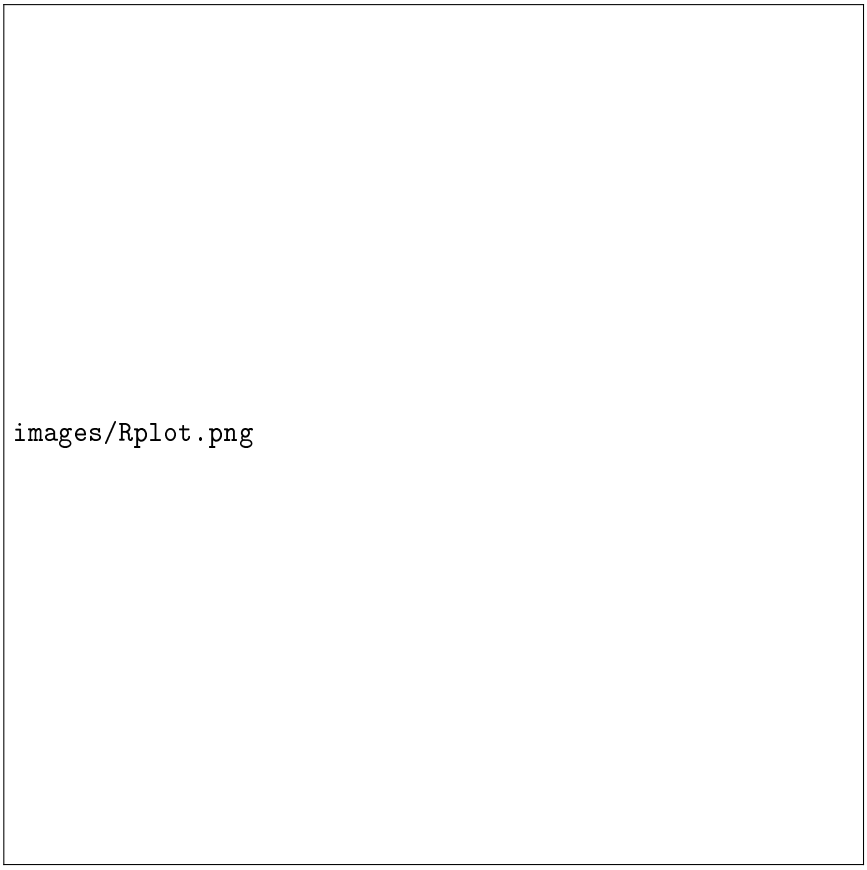
Data to be charted:

Month	Degrees
01	3.8
02	4.1
03	6.3
04	9.0
05	11.9
06	15.1
07	17.1
08	17.4
09	15.7
10	11.8
11	7.7
12	4.8

Code:

```
plot(data, type="b", bty="l", col=c("#ABD249"), las=1, lwd=4)
grid(nx=NULL, ny=NULL, col=c("#E8E8E8"), lwd=1)
legend("bottom", legend=c("Degrees"), col=c("#ABD249"), pch=c(19))
```

The resulting chart:



Citations

Cross-referenced to bibliography.

Appendix

Special sections.

Index

Index (or list of acronyms).

- Write index entries

Note that multi-entry terms generate separate index entries.

- Place the index at the desired location
- Produce the index by updating `org-latex-pdf-process`

Bibliography

The bibliography...

- Eric Steven Raymond. The Art of Unix Programming. Addison-Wesley. ISBN 0-13-142901-9.

Glossary

Glossaries are optional. Glossaries entries are an example of definition lists.

A glossary term The corresponding (indented) definition.

A second glossary term The corresponding (indented) definition.

Contributing

Issues

Report issues and suggest features and improvements on the GitHub issue tracker.

Patches

I love contributions! Patches under any form are always welcome!

Donations

If you use the refcard-org-mode project and feel it is making your life better and easier, you can show your appreciation and help support future development by making a donation through PayPal. Thank you!

Regardless of the donations, refcard-org-mode will always be free both as in beer and as in speech.

License

Author: Fabrice Niessen

Keywords: org-mode refcard