Org Beamer quick reference card

Fabrice Niessen

March 28, 2014

Plan

Description

Welcome to Org Beamer reference card. It contains the reference documentation that describes how to write presentations using Org mode 8+ and the LaTeX Beamer class.
That allows you to produce high quality PDF files which are going to look on every computer exactly the way they looked on your computer.

Objectives

Preparing presentations with Org mode is very different from preparing them with WYSWYG programs such as PowerPoint, Impress or Keynote, as they are created like any other Org mode document.
The obvious advantage of this approach is that you don't have to know LaTeX in order to use Org mode's Beamer export and create presentations.
Org mode presentations contain headings at different levels. By default, headings at the first outline level will become titles of the different slides (called frames in Beamer), and deeper levels will be used as structural environments.

Requirements

A working LaTeX installation is required for exporting to PDF. If it is not yet installed on your system, install TeX Live (for example).

Beamer back-end (for Org export engine)

Type:

```
M-x load-library RET ox-beamer RET
```

to load the Beamer back-end library, and to obtain extra commands in the LaTeX export menu:

`C-c C-e l B`	As LaTeX buffer (Beamer).
`C-c C-e l b`	As LaTeX file (Beamer).
`C-c C-e l P`	As PDF file (Beamer).
`C-c C-e l O`	As PDF file and open (Beamer).

Editing support

Type:

```
M-x org-beamer-mode RET
```

to load the minor mode `org-beamer-mode` easing the edition of the document structure (through the key binding `C-c C-b`, which offers fast selection of a Beamer environment).

You can also turn it on with:

```
#+STARTUP: beamer
```

in your document.

Appearance of the presentation

```
#+BEAMER_THEME: Boadilla
```

is equivalent (for Boadilla) to:

```
#+BEAMER_COLOR_THEME: dolphin
#+BEAMER_FONT_THEME:  default
#+BEAMER_INNER_THEME: [shadow]rounded
#+BEAMER_OUTER_THEME: infolines
```

LaTeX preamble

Append any line of code in the LaTeX preamble with:

```
#+BEAMER_HEADER: \usepackage{...}
```

It will go in the `[EXTRA]` placeholder of the header associated to the `beamer` LaTeX class (see `org-latex-classes`).

Affiliated keywords

The Beamer back-end reads both

- `#+ATTR_LATEX:` and
- `#+ATTR_BEAMER:`

affiliated keywords.

Creating a title page

Creating a table of contents

If you set the `H` option from the `#+OPTIONS:` keyword such as:

```
#+OPTIONS: H:2
```

then:

- Top-level headlines become sections listed in the table of contents (created by default), and
- Second-level headlines become the frames.

In many themes, sections and subsections appear in the sidebar or headline.

Creating a simple frame

```
* Introduction

** A title
   #+BEAMER: \framesubtitle{A subtitle}

Some content.
```

The subtitle does not have an Org syntax because it's specific to the Beamer back-end only.

Create a handout

againframe

appendix

You can add an appendix to your talk by using the `\appendix` command. You should put frames and perhaps whole subsections into the appendix that you do not intend to show during your presentation, but which might be useful to answer a question. The `\appendix` command essentially just starts a new part named `\appendixname`. However, it also sets up certain hyperlinks. Like other parts, the appendix is kept separate from your actual talk.

column

columns

frame

- Headlines become frames when their level is equal to `org-beamer-frame-level` (or `H` value in the `OPTIONS` line).
- Though, if a headline in the current tree has a `BEAMER_env` property set to either `frame` or `fullframe`, its level overrides the variable.

- A `fullframe` is a `frame` with an ignored title
  - `frametitle` is set to the empty string

- A headline with an `ignoreheading` environment will have its contents only inserted in the output.
  - Contents is not inserted in any `frame` environment...
- This special value is useful to have data between frames, or to properly close a `column` environment.

note

noteNH

Environment specification (BEAMER_env)

Use a different environment in current block.

`structureenv` environment

- For highlighting text.
- To help the audience see the structure of your presentation.

Paragraph Heading.

`block` environment

Answered Questions

How many primes are there?

Open Questions

Is every even number the sum of two primes?

`alertblock` environment

- Inserts a block whose title is highlighted.
- Behaves like the `block` environment otherwise.

Wrong theorem

1=2.

`exampleblock` environment

- Inserts a block that is supposed to be an example.
- Behaves like the `block` environment otherwise.

Example

The set *{1,2,3,5}* has four elements.

theorem environment

- Inserts a theorem.

Theorem

There is no largest prime number.

`theorem` environment

- Inserts a theorem.
- Simpler solution
  - More readable
  - Less powerful: you can't nest blocks of the same type with this syntax

Theorem

There is no largest prime number.

`definition` environment

- Behaves like the `theorem` environment, except that the theorem style `definition` is used.
- In this style, the body of a `theorem` is typeset in an upright font.

Definition (definition)

Contents of definition

`example` environment

- Behaves like the theorem environment, except that the theorem style `example` is used.
- A side-effect of using this theorem style is that the contents is put in an `exampleblock` instead of a `block`.

Example (Example)

Contents of example

example environment

- Simpler solution:

```
Contents of example
```

proof environment

- Typesets a proof.

proof.

- Suppose p were the largest prime number.

- But $q + 1$ is greater than 1, thus divisible by some prime number not in the first p numbers. $\square$

proof environment

- Typesets a proof.

proof.

- Suppose p were the largest prime number.
- Let q be the product of the first p numbers.

- But $q + 1$ is greater than 1, thus divisible by some prime number not in the first p numbers. $\square$

proof environment

- Typesets a proof.

proof.

- Suppose p were the largest prime number.
- Let q be the product of the first p numbers.
- Then $q + 1$ is not divisible by any of them.
- But $q + 1$ is greater than 1, thus divisible by some prime number not in the first p numbers. □

beamercolorbox environment

- Create colored boxes.

Text

`verse` environment

Contents of verse

`verse` environment

- Simpler solution:

  Contents of verse

quotation environment

- Use quote or quotation to typeset quoted text.
- quotation has paragraph indentation.

    Contents of quotation

quote environment

- Use `quote` or `quotation` to typeset quoted text.
- `quote` hasn't paragraph indentation.

> *Contents of quote*

quote environment

- Use quote or quotation to typeset quoted text.
- quote hasn't paragraph indentation.
- Simpler solution:

  Contents of quote

Verbatim

```
int main (void)
{
  std::vector<bool> is_prime (100, true);
  for (int i = 2; i < 100; i++)
    if (is_prime[i])
      {
        std::cout << i << " ";
        for (int j = i; j < 100;
             is_prime [j] = false, j+=i);
      }
  return 0;
}
```

Extra environments

For simple environments, use:
I think we should changes some environment placeholders:

- Introduce %r which would stand for the raw headline (without any processing)
- %H and %U would use the raw headline text instead.

The previous definition would become:
WDYT?

- Environment options may be given using the BEAMER_opt property. They will be enclosed in square brackets and inserted where %o appears in the environment definition. (with an example, but I can't think of one now)
- Additional arguments may be written into the environment's headline, and inserted into the LaTeX string using %r (raw headline text, no processing).

Overlay specification (BEAMER_act)

Set overlay specifications in current block to create dynamic effects (*multiple slides*, called *overlays*, for a single frame).
Headlines support the BEAMER_ACT property:

```
* Headline
  :PROPERTIES:
  :BEAMER_ACT: [+-]
  :END:

  # Diff with [<+->]?

  - Item
  - Item
```

It is translated as an overlay/action specification, or a default overlay specification when enclosed within square brackets.
Dynamic lists are possible on a case by case basis :

```
#+ATTR_BEAMER: :overlay +-
- Item 1
- Item 2
```

```
- @@beamer:<1->@@ Item 1
- @@beamer:<2->@@ Item 2
```

Option specification (BEAMER_opt)

Insert optional arguments for the current frame (or in current block???).

Options for the current frame (opt)

- Headlines support `BEAMER_OPT` properties.

I'd still like to see something more like a "for-dummies" explanation of passing options and arguments to LaTeX entities. I'm not saying the documentation is woefully inadequate (hardly that – Suvayu's page got me rather far, and I got stuck on a couple of details). My experience was: it never would have occurred to me on my own to use the headline text for LaTeX code, and if there was a hint anywhere in the docs to suggest that this would be the way to go, I didn't find it. That's a conceptual leap that passed me by.

- This is for frames, and for environments within a frame
- It specifies options for the current frame or block, and will automatically be enclosed within square brackets.
- `fragile` option is added automatically
- You might want to put `allowframebreaks=0.9` there

Frame structure (Explicit page breaking) I

If the text does not fit on a single slide, all you have to do to automatically break up the frame into several frames, is set the option `allowframebreaks`:

```
** A long "frame" with breaks
   :PROPERTIES:
   :BEAMER_opt: allowframebreaks,label=
   :END:
```

Until the Beamer issue #265 is solved, we need to unset the framelabel as shown above (~label~=).

Splitting a frame into multiple columns

BMCOL

TODO Specify 5cm

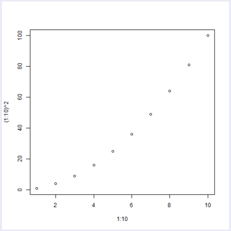
BMCOL

Two lines.

One line (but aligned).

Multiple columns

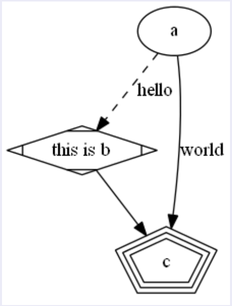
Two lines.

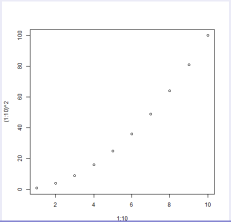


Result of an evaluation on two columns

Using graphics

Figures





Skip proof

nil

Summary

For further reading

A. Salomaa.
Formal Languages.
Academic Press, 1973.

For further reading

A. Salomaa.
Formal Languages.
Academic Press, 1973.

E. Dijkstra.
Smoothsort, an alternative for sorting in situ.
Science of Computer Programming, 1(3):223–233, 1982.

For further reading

A. Salomaa.
Formal Languages.
Academic Press, 1973.

E. Dijkstra.
Smoothsort, an alternative for sorting in situ.
Science of Computer Programming, 1(3):223–233, 1982.

E. Feldman and J. Owings, Jr.
A class of universal linear bounded automata.
Information Sciences, 6:187–190, 1973.

For further reading

A. Salomaa.
Formal Languages.
Academic Press, 1973.

E. Dijkstra.
Smoothsort, an alternative for sorting in situ.
Science of Computer Programming, 1(3):223–233, 1982.

E. Feldman and J. Owings, Jr.
A class of universal linear bounded automata.
Information Sciences, 6:187–190, 1973.

P. Jančar, F. Mráz, M. Plátek, and J. Vogel.
Restarting automata.
FCT Conference 1995, LNCS 985, pages 282–292. 1995.

Proof details

Text omitted in main talk.

More details

Even more additional material.

Abbreviations