How to teach using org-mode for fun and profit

Olivier Berger

(2018-04-23)

ORG-TEACHING - EXAMPLE LESSON

Contents

1 Introduction

This is a demo document about the codename `org-teaching` framework, which aims at managing teaching material using Org-mode.

2 Org-mode powa

Attention, this framework heavily relies on:

- org-mode (version 9 at the time of writing)
- and the org-reveal exporter for `reveal.js`.

> Since Org-mode is plain text, you may be able to edit contents without Emacs and org-mode, but I'd recommend learning org-mode for serious work ;-)

3 About this handbook

This PDF handbook is one variant of the same teaching material, also available as a slides deck. Note that the layout is formatted as a portrait handbook, including speaker notes.

This section may contain content that is best viewed in (LaTeX) PDF export of org-mode.

4 Features

4.1 Writing teaching material in org-mode

The goal is to be able to edit a single file (namely `lesson.org`) which will contain, in a **single source**, all the content of a lesson, written with org-mode syntax.

From this single source, several documents are generated :

- **slides** (as a dynamic Web document) for overhead presentation
- a **handbook** that contains the same information (or more) and can be handed to the students for work outside the classroom (a PDF file)
- optionaly, another **handbook** for the teaching team, to provide additional instructions (also a PDF file)

> The student handbook's LaTeX formatting has a summarized content table and nice looking (in principle) title page. Customize at will.
> The teachers handbook contains a less nice-looking format, includes a detailed table of contents, and has a watermark. Again, customize at will and submit improvements.

4.2 Frugal org-reveal/reveal.js slides

Pretty much all features of `reveal.js`, supported by the org-mode reveal.js exporter (org-reveal), should be supported too.

An example `org-reveal` document is available here for inspiration (it's the export of org-reveal's `README.org`, actually).

> If you're already familiar with reveal.js, you may have noticed that the current settings adopted for our slides generation are quite frugal: no fancy 3D effects and likes.
> It's a matter of taste : I didn't want to show off, and prefer to give students a clear content on the projector behind me.

4.3 Structure of the sections / slides

I'm using the 3 levels of outlining / sectioning so that the content can be sectioned in the same way in `lesson.org` and appear appropriately in the slides and handbook, with these principles:

1. First level outlines define main sections of the document.
2. Second level outlines are the main "horizontal" slides that will be played with page up/down
3. Third level outlines may be used for additional content ("vertical" slides) that may be skipped for the presentation, but is still accessible with cursor keys.

> The first level outlines can be rendered as a "separating" slides which may get a different `reveal_background` and `class="center"` slide layout, but that isn't automatic. See Section separators.

4.4 Presenter notes / content for the handbook

org-reveal's *Speaker notes* may be added to the slides (and will only appear on dual-screen presentation after having pressed '*s*': standard reveal.js feature).

They will be masked for the audience, but will, by default, appear in the handbook given to the students.

```
#+BEGIN_NOTES
This is a note
#+END_NOTES
```

4.5 Masking content for some audiences

I've implemented some "easy ways" to preserve some of the content of the same `lesson.org` source for certain outputs (using org exporter's standard `EXCLUDE_TAGS`):

***Slides only* material** that won't be embedded in the handbook : surprise stuff for live audience, or HTML-only hacks;

***Teachers only* material** secret knowledge that only adults need to know (for instance), which won't be exported;

***Handbook only* material** stuff that only fits in the handbook, and/or only exports as LaTeX and not HTML.

> The choice to reveal or not some details to the students is quite arbitrary and depends on your pedagogical approach. I'm not advisable in this matter. YMMV.

4.6 Stuff only meant for presentation

Tagging a section/slide with `:slidesonly:` means it isn't exported in the handbooks.

Below is an example (or not)…

4.6.1 Regular slide (no tag on heading line)

There should be no "Only in the slides" after this section, in the handbooks, as it has been tagged with `slidesonly`.

4.7 Stuff only meant for teachers

Tagging a section/slide with `:teachersonly:` means it isn't exported in the students handbook (nor in the slides).

Below is an example…

4.7.1 Regular slide (no tag on heading line)

There should be no "Only for teachers" after this section, in the slides or in the students handbook, as it has been tagged with `teachersonly`.

4.8 Notes only for the teachers

This slide/section contains notes, but only part of it is displayed in the presentation notes included in the handbook. Special notes and are kept only for the teachers handbook.

We use an org-mode drawer for that (additional benefit is that the content is folded by default in emacs, as it may be verbose and/or "sensitive") :

```
#+BEGIN_NOTES
This part of the note can be viewed by the students in the handbook.

:TEACHERSONLY:
Not this one
:END:
#+END_NOTES
```

This part of the note can be viewed by the students in the handbook, but not the rest.

4.9 Stuff only in the handbooks

Just like sections are for slides only, others can be for the handbook only, using the `handbookonly` tag. This may be useful for **Annex** sections for instance, or for stuff that the HTML exporter won't like, with inline LaTeX.

4.10 Code colorization

Code is colorized / highlighted in the slides :-)

Nice when like me, you're teaching Computer Science stuff

4.11 Misc org-mode

4.11.1 Babel powa

As you're using org-mode, its `babel` components are available, to embed source code in the same `lesson.org` source, and manage executable code and teaching material at once.

Look for *literate programing* instructions in the org-mode docs to know more.

4.11.2 Jumping to slide number

Included is the use of the reveal.js jump plugin to allow jumping directly to slides # by entering a number and hitting RETURN. Quite handy while writing and testing slides.

4.12 Missing features ?

Please try and talk to me to suggest new stuff and/or provide patches ;)

> See the teacher's handbook for some ideas

5 Authoring

5.1 Modify only the lesson.org

Only one file should be edited for writing the lesson's material : `lesson.org`

Only exception is modification of some configurations for title pages and other bits that shouldn't change much in time (see next section).

5.2 Use Emacs org-mode exporters or the Docker container

You have 2 options to generate the different formats:

- either manualy use org-mode exporters from inside Emacs
- or use the Docker container for automation / reproducibility

> You're welcome to suggest improvements. But I'm not an Elisp hacker, so I may not be able to maintain them. At the moment, the intent is to rely on the original org-reveal only, as much as possible.

5.3 Use docker

You may use the `olberger/docker-org-export` docker container image I've prepared, to make org-mode exports. Or you may rebuild it yourself (see below).

5.3.1 Build the Docker container image

This is recommended to avoid man in the middle, IMHO:

```
cd docker
docker build -t obergixlocal/docker-org-export .
```

5.3.2 Run the container

Use the provided `docker/docker-org-export` script, which relies on the `olberger/docker-org-export` container image. See how `Makefile` does it.

5.4 Configuration

Each `lesson.org` needs some configuration :

- Configure `org-reveal-title-slide` in `slides.org`.
- Configure in the headers elements like:
  - *header* (`\lhead{...}` and `\rhead{...}`)
  - and *footer* (`\lfoot{...}` and `\rfoot{...}`)

  ex: `#+LaTeX_HEADER: \rhead{...}` in `handbook.org` and `teacher-handbook.org`.

> These may be better handled, but some limitations of the exporters or my lack of knowledge/time have prevented a better result so far. Improvements much welcome.

5.5 Generating final documents

We're using the standard exporters so each output format will be exported from its corresponding umbrella `.org` source :

slides open `slides.org`, then `C-c C-e R ...` for `org-reveal` export (to `slides.html`), provided that you have loaded org-reveal in Emacs

handbook open `handbook.org`, then `C-c C-e l ...` for LaTeX export (to `handbook.pdf`)

teacher handbook open `teacher-handbook.org`, then `C-c C-e l ...` for LaTeX export (to `teacher-handbook.pdf`)

5.5.1 Exporting slides to HTML with org-reveal

Depending on how you installed org-reveal (Git submodules or otherwise), `org-reveal` may already be available.

If not yet, load it with `M-x load-file` from the location of its Git submodule (`elisp/org-reveal/ox-reveal.el` by default).

> I'm not sure which solution is better : org-reveal from Git (hence the Git submodule) or from an Emacs package. Please report.

5.5.2 Printing slides

I've tested DeckTape using a Docker container containing `PhantomJS` and `decktape` to convert the slides to a single PDF document.

See the provided decktape.sh script that runs the container, bind-mounting the working dir into the container, so that input and output files can be found.

Note that I used a rebuilt Docker image, reusing the DeckTape Dockerfile, rebuilding with something alongside:

```
docker build -t obergixlocal/decktape .
```

5.6 Known Issues

5.6.1 Firefox issues ?

We have experienced issues with presentations made on some versions of Firefox, which are known by reveal.js maintainer... maybe best viewed in chrome.

You may prefer to have a PDF variant of the slides (see Printing slides) in case.

6 How it works / Installation

6.1 Use the source (Luke)

See the contents of the files... but be wary that it's sometimes messy and incrementally obtained.

Emacs is your buddy.

Git clone from `https://gitlab.com/olberger/org-teaching.git` (see the Gitlab project)

6.1.1 Git submodules

The repository contains Git submodules for :

- `reveal.js/`
- `elisp/org-reveal`
- reveal.js's jump plugin (`reveal.js-jump-plugin/`)

So :

```
git submodule init
git submodule update
```

You may prefer to install them another way (ELPA repo, CDN, etc.)

> Refer to org-reveal's documentation for more details.

6.2 Customize slides appearance

6.2.1 Reveal.js settings

See the org-reveal settings set in the sources and the docs for a detailed explanation.

I'm using the following for a "frugal" look close to what powerpoint or beamer (?) could look like :

```
#+REVEAL_HLEVEL: 2
#+REVEAL_THEME: simple
#+REVEAL_TRANS: fade
#+REVEAL_SPEED: fast
#+REVEAL_MARGIN: 0.0
#+REVEAL_EXTRA_CSS: ./presentation.css
#+REVEAL_ROOT: ./reveal.js

#+OPTIONS: reveal_center:nil
```

6.2.2 Section separators

The highest level sections include the following properties below the heading line, to customize the look of the slide.

```
:PROPERTIES:
:REVEAL_EXTRA_ATTR: class="center"
:reveal_background: #dbdbed
:END:
```

This is intended to provide some visual sense of the transitions between sections. Please adapt and report.

6.2.3 Title screen picture (logos, etc.)

I'm not yet sure how much may be achieved with HTML and CSS for the title page of the slides deck, so I've relied on the embedding of a background image that will contain the logos and additional graphics.

```
#+REVEAL_TITLE_SLIDE_BACKGROUND: ./media/title-slide-background.png
```

I'm quite sure this could be improved.

7 Annex

7.1 Thanks

- All contributors to org-mode (special kudos to Carsten Dominik and Bastien Guerry)
- Yujie Wen for `org-reveal`
- Hakim El Hattab for `reveal.js`
- My colleagues at Telecom SudParis who had to teach with this tool without much rehersal
- Our students who endured it for a brand new course (and included bugs)

7.2 Feedback

I may be contacted from my Web page or via the Gitlab project.

7.3 Usage reports

7.3.1 2016-2018 at Telecom SudParis

Created and used for 2 editions of teaching "Web Architecture and Applications" in the CSC4101 module at Telecom SudParis (Olivier Berger and colleagues)