Class 5: Exporting Slides and Handouts

Table of Contents / Agenda

1 Exporting Content in Org Mode

Org mode provides powerful export functionalities, which enable users to convert Org markup content to a variety of other formats. The outputs have proper formatting while maintaining the original structure and markup. The Org-Coursepack is set up to use reveal.js for slides and LaTeX for handouts. Users can adapt the current setting to accomodate other output formats.

General information about exporting can be found in the Org manual. Hence, this lecture focuses on describing how exporting is set up in the Org-Coursepack, introducing pre-built export functionalities of the Org-Coursepack, and offering useful exporting tips for instructors.

1.1 Setting Export Scope to `Subtree`

The export command, `org-export-dispatch` (`C-c C-e`), takes the user to the `Org Export Dispatcher` interface. Here, the user can select whether to export the buffer (i.e., the whole file) or only a subtree. An instructor may use the former to create a course booklet and the latter to export slides for a full or part of a lecture. The default scope of Org mode is `Buffer`, but a user can put the following snippet in the Emacs init file to set the default scope to `Subtree`.

```
;; set the default export scope to subtree
(setf org-export-initial-scope 'subtree)
```

1.2 Export Settings

Org mode allows users to specify export settings at both buffer and subtree levels.

1.2.1 Buffer-Level Settings

For buffer-level settings, one can use the `#+OPTIONS:` statement. For example, including the following line in an Org file will include a table of contents for that file in the exported document:

```
#+OPTIONS: toc:t
```

Similarly including the following line will add numbers in front of the headings.

```
#+OPTIONS: num:t
```

1.2.2 Subtree-Level Settings

Export settings can be specified at the subtree level with `:PROPERTIES:`. Simply add `:EXPORT_:` as a prefix to each option. For example, the title of the document can be set with `:EXPORT_TITLE:`. To specify multiple settings (e.g., items for LaTeX preamble), one can add + to the property name to append an additional value. For example,

```
:PROPERTIES:
:EXPORT_LATEX_HEADER+: \usepackage{titling}
:EXPORT_LATEX_HEADER+: \usepackage{multicol}
:END:
```

2 Clickable Links and Keybindings for Exporting

2.1 Buttons

We include export buttons (in the form of links written in Emacs-lisp) in the properties of any exportable subtree in the Org-Coursepack (e.g., lectures, syllabi, and exams).

For example, each lecture headline comes with the `:EXPORT_TO:` property, which includes buttons such as `reveal.js` and `LaTeX`. These buttons will export files to their respective format, using `Subtree` as the export scope.

After exporting, users can click on buttons in the `:VIEW_OUTPUT:` property (e.g., `HTML` or `PDF`) to open the exported files.

2.2 Key Bindings

Using the command=org-export-dispatch= (`C-c C-e`) when exporting a content allows users to later repeat the last export action for that same content using the prefix argument (`C-u`). This is a convenient feature when exporting the same content multiple times.

> Note that the cursor shoud be in the correct position (heading, properties, or spaces between the heading and its first subheading) of the subtree being exported. When using `C-u C-c C-e` for repeated export, however, the cursor position does *not* matter as long as the same buffer which contains the last exported subtree is open.

To bind `C-u C-c C-e` to a key (`F5` in this example), include the following Emacs-lisp code in into the init file:

```
;; bind f5 to keyboard macro of export-last-subtree
(fset 'export-last-subtree
      "\C-u\C-c\C-e")

(eval-after-load "org"
  '(progn
     (define-key org-mode-map (kbd "<f5>") 'export-last-subtree)))
```

3 Exporting Slides: with reveal.js

- See the org-reveal documentation for instructions on installation and usage.

4 Exporting Handouts: with LaTeX

LaTeX export is extensively supported by Org mode. We refer users to the Org manual for the in-depth instructions.

The following snippet shows the basic setup for our LaTeX output, where the `koma-article` class is added to `org-latex-classes` and the `minted` package is used for syntax highlighting. Currently, Python is the only language added to `org-latex-minted-langs`. Users can add to `org-latex-minted-langs` any other languages they want processed with the `minted` package.

> Note that we manually added the `minted` package to LaTeX preambles as opposed to adding it to `org-latex-packages-alist`. This was to allow for flexible specifications of the `outputdir` option.

```
(eval-after-load 'ox '(require 'ox-koma-letter))

(eval-after-load 'ox '(add-to-list 'org-latex-classes
                                   '("koma-article"
                                     "\\documentclass{scrartcl}"
                                     ("\\section{%s}" . "\\section*{%s}")
                                     ("\\subsection{%s}" . "\\subsection*{%s}")
```

```
                                        ("\\subsubsection{%s}"
                                         . "\\subsubsection*{%s}")
                                        ("\\paragraph{%s}" . "\\paragraph*{%s}")
                                        ("\\subparagraph{%s}"
                                         . "\\subparagraph*{%s}"))) )

(require 'ox-latex)
(setq org-latex-listings 'minted)

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

(add-to-list 'org-latex-minted-langs '(python "python"))
```

4.1 Inserting a Page Break Before a Heading in LaTeX Export

Users can add a page break in the LaTeX export by inserting `#+LATEX: \clearpage`. Importantly, adding the following code into the init file automatically inserts a page break before any subtree that has a `:newpage:` tag.

```
(defun org/get-headline-string-element  (headline backend info)
  "Return the org element representation of an element.

  Won't work on ~verb~/=code=-only headers"
  (let ((prop-point (next-property-change 0 headline)))
    (if prop-point (plist-get (text-properties-at prop-point headline) :parent))))

(defun org/ensure-latex-clearpage (headline backend info)
  (when (org-export-derived-backend-p backend 'latex)
    (let ((elmnt (org/get-headline-string-element headline backend info)))
      (when (member "newpage" (org-element-property :tags elmnt))
        (concat "\\clearpage\n" headline)))))

(eval-after-load 'ox '(add-to-list
                       'org-export-filter-headline-functions
                       'org/ensure-latex-clearpage))
```

5 Selective Export

By using raw code and custom Emacs-lisp scripts, users can flexibly choose which content to show/hide, depending on output format. For example, instructors may want to show images in slides but not in handouts, or they may want to include supplementary notes in handouts but not in slides.

5.1 Tagging a Subtree as Slide or Handout Only

With the code below in added to your init file, you can use the `:slideonly:` or `:handoutonly:` tags to selectively include a subtree in either a slide output or handout output, respectively. Currently LaTeX and `rst` backends is set as a handout output, and reveal.js and beamer backends are set as slide outputs.

For example,

```
* This subtree will only be exported in slide output      :slideonly:
- Content
* This subtree will only be exported in handout output    :handoutonly:
- Content
```

```
(defun org/parse-headings (backend)
  "Remove every headline with certain tags in the
  current buffer. BACKEND is the export back-end being used, as
  a symbol.

  "

  (if (member backend '(latex rst))
      (org-map-entries
       (lambda ()
         (progn
           (org-narrow-to-subtree)
           (org-cut-subtree)
           (widen)
           ))
       "+slideonly"))

  (if (member backend '(reveal beamer))
      (org-map-entries
       (lambda ()
         (progn
           (org-narrow-to-subtree)
           (org-cut-subtree)
           (widen)
           ))
       "+handoutonly"))

)

(add-hook 'org-export-before-parsing-hook 'org/parse-headings)
```

5.2 Hiding Specific Content

To hide content when exporting to HTML-based format outputs (slides), use raw HTML tags `<span hidden>` and `</span>`. See the example below.

```
#+REVEAL_HTML: <span hidden>
This will not be shown in reveal.js output
#+REVEAL_HTML: </span>
```

Similarly, any content placed between `\iffalse` and `\fi` will not be rendered in LaTeX outputs (handouts). See the example below.

```
#+LATEX: \iffalse
This will not be shown in LaTeX output
#+LATEX: \fi
```