Slides and Course Notes for Jacobs University*

Michael Kohlhase
Jacobs University, Bremen
`http://kwarc.info/kohlhase`

November 16, 2015

Abstract

We present a document class from which we can generate both course slides and course notes in a transparent way. Furthermore, we present a set of LaTeXML bindings for these, so that we can also generate OMDoc-based course materials, e.g. for inclusion in the ActiveMath system.

Contents

---

*Version ? (last revised ?)

1 Introduction

This Document class is derived from `beamer.cls` [**beamerclass:on**], specializes it with Jacobs stuff and adds a notes version that is more suited to printing than the one supplied by `beamer.cls`.

2 The User Interface

The `mikoslides` class takes the notion of a slide frame from Till Tantau's excellent `beamer` class and adapts its notion of frames for use in the $\mathcal{S}$TEXand OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `mikoslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the `mikoslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

2.1 Package Options

EdN:1

The `mikoslides` class takes a variety of class options:[1]

slides
a
- The options `slides`nd `notes`notes switch between slides mode and notes mode (see Section 2.2).

sectocframes
- If the option `sectocframes` is given, then special frames with section table of contents are produced headers [2]

EdN:2

showmeta
- `showmeta`. If this is set, then the metadata keys are shown (see [**Kohlhase:metakeys:ctan**] for details and customization options).

frameimages
- If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames.

2.2 Notes and Slides

frame
note
Slides are represented with the `frame` just like in the `beamer` class, see [**Tantau:ugbc**] for details. The `mikoslides` class adds the `note` environment for encapsulating the course note fragments.[1]

⚠ Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else LATEX becomes confused and throws error messages that are difficult to decipher.

---

[1]EDNOTE: leaving out noproblems for the moment until we decide what to do with it.

[2]EDNOTE: document the functionality

[1]MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive LaTeX trickery. Hints to the author are welcome.

```
\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 1: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 1.

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add sTEXnotes. In this case we can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where ⟨opt⟩ are the options of `\includegraphics` from the `graphicx` package [**CarRah:tpp99**] and ⟨path⟩ is the file path (extension can be left off like in `\includegraphics`).

\frameimage

2.3 Header and Footer Lines

2.4 Colors and Highlighting

\textwarning   The `\textwarning` macro generates a warning sign: ⚠

2.5 Front Matter, Titles, etc

2.6 Miscellaneous

3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTEXGitHub repository [**sTeX:github:on**].

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined

by the time the heading is formatted. Need to look into how the headings are made. This is a problem of the underlying `omdoc` package.

4   The Implementation

The `mikoslides` package generates two files: the LaTeX package (all the code between ⟨*package⟩ and ⟨/package⟩) and the LaTeXML bindings (between ⟨*ltxml⟩ and ⟨/ltxml⟩). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

The general preamble for LaTeXML:

```
1 ⟨∗ltxml.cls | ltxml.sty⟩
2 # -*- CPERL -*-
3 package LaTeXML::Package::Pool;
4 use strict;
5 use LaTeXML::Package;
6 ⟨/ltxml.cls | ltxml.sty⟩
```

4.1   Class and Package Options

We define some Package Options and switches for the `mikoslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `mikoslides` package.

```
 7 ⟨∗cls⟩
 8 \newif\ifnotes\notesfalse
 9 \DeclareOption{notes}{\notestrue\PassOptionsToPackage{\CurrentOption}{mikoslides}}
10 \DeclareOption{slides}{\notesfalse\PassOptionsToPackage{\CurrentOption}{mikoslides}}
11 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{omdoc}
12                              \PassOptionsToClass{\CurrentOption}{beamer}
13                              \PassOptionsToPackage{\CurrentOption}{mikoslides}}
14 \ProcessOptions
15 ⟨/cls⟩
16 ⟨∗ltxml.cls⟩
17 DefConditional('\if@mikoslides');
18 DeclareOption('mh', sub {Digest(T_CS('\@mikoslidestrue')); });
19 DeclareOption('notes', sub{PassOptions('mikoslides','sty',ToString(Digest(T_CS('\CurrentOption'
20 DeclareOption('slides', sub{PassOptions('mikoslides','sty',ToString(Digest(T_CS('\CurrentOption
21 DeclareOption(undef, sub {PassOptions('omdoc','cls',ToString(Digest(T_CS('\CurrentOption'))))); 
22                                        PassOptions('mikoslides','sty',ToString(Digest(T_CS('
23                                        PassOptions('beamer','cls',ToString(Digest(T_CS('\Cur
24 ProcessOptions();
25 ⟨/ltxml.cls⟩
```

now we do the same for the `mikoslides` package. Note that we also have to define the same switches[3], since we might use `mikoslides.sty` in a different class.

EdN:3

```
26 ⟨∗package⟩
```

---

[3] EDNOTE: MK: we may think about making all of them internal

```
27 \newif\if@mikoslides@mh@\@mikoslides@mh@false
28 \DeclareOption{mh}{\@mikoslides@mh@true}
29 \newif\ifnotes\notesfalse
30 \DeclareOption{notes}{\notestrue}
31 \DeclareOption{slides}{\notesfalse}
32 \newif\ifsectocframes\sectocframesfalse
33 \DeclareOption{sectocframes}{\sectocframestrue}
34 \newif\ifframeimages\frameimagesfalse
35 \DeclareOption{frameimages}{\frameimagestrue}
36 \newif\if@part\@partfalse
37 \DeclareOption{report}{\@parttrue\PassOptionsToPackage{\CurrentOption}{omdoc}}
38 \DeclareOption{book}{\@parttrue\PassOptionsToPackage{\CurrentOption}{omdoc}}
39 \newif\ifproblems\problemstrue
40 \DeclareOption{noproblems}{\problemsfalse}
41 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}
42                          \PassOptionsToPackage{\CurrentOption}{tikzinput}}
43 \ProcessOptions
44 ⟨/package⟩
45 ⟨∗ltxml.sty⟩
46 DeclareOption('notes', '');
47 DeclareOption('slides', '');
48 DeclareOption('noproblems', '');
49 DeclareOption('sectocframes', '');
50 DeclareOption('frameimages', '');
51 DeclareOption(undef, sub {PassOptions('stex','sty',ToString(Digest(T_CS('\CurrentOption'))));
52                          PassOptions('tikzinput','sty',ToString(Digest(T_CS('\
53 ProcessOptions();
54 RawTeX('\newif\ifnotes\notesfalse');
55 RawTeX('\newif\ifproblems\problemsfalse');
56 ⟨/ltxml.sty⟩
```

Depending on the options, we either load the `article`-based `omdoc` or the `beamer` class. In the first case, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the STEX packages.

```
57 ⟨∗cls⟩
58 \ifnotes
59   \LoadClass{omdoc}
60   \RequirePackage{a4wide}
61   \RequirePackage{marginnote}
62   \RequirePackage{mdframed}
63   \RequirePackage[notheorems,noamsthm,noxcolor]{beamerarticle}
64 \else
65   \LoadClass[notheorems,noamsthm,10pt]{beamer}
66   \newcounter{Item}
67   \newcounter{paragraph}
68   \newcounter{subparagraph}
69   \newcounter{Hfootnote}
70   \usetheme{Jacobs}
```

```
71 \fi
72 \RequirePackage{mikoslides}
73 ⟨/cls⟩
74 ⟨*ltxml.cls⟩
75 LoadClass('omdoc');
76 RequirePackage('mikoslides');
77 if(IfCondition(T_CS('\if@mikoslides'))) {RequirePackage('mikoslides-mh');}
78  DefConstructor('\usetheme{}','');
79 ⟨/ltxml.cls⟩
```

now, we load the remaining packages for both versions.

```
80 ⟨*package⟩
81 \if@mikoslides@mh@\RequirePackage{mikoslides-mh}\fi
82 \RequirePackage{stex}
83 \RequirePackage{smglom}
84 \RequirePackage{tikzinput}
85 \RequirePackage{latexml}
86 \RequirePackage{amssymb}
87 \RequirePackage{amsmath}
88 \RequirePackage{comment}
89 \RequirePackage{textcomp}
90 \RequirePackage{url}
91 ⟨/package⟩
92 ⟨*ltxml.sty⟩
93 RequirePackage('stex');
94 RequirePackage('smglom');
95 RequirePackage('tikzinput', options => ['image']);
96 RequirePackage('latexml');
97 RequirePackage('amssymb');
98 RequirePackage('amsmath');
99 RequirePackage('graphicx');
100 RequirePackage('url');
101 ⟨/ltxml.sty⟩
```

4.2 Notes and Slides

We define the sizes of slides in the notes. Somehow, we cannot get by with the
same here.

```
102 ⟨*package⟩
103 \newcounter{slide}
104 \newlength{\slidewidth}\setlength{\slidewidth}{12.8cm}
105 \newlength{\slideheight}\setlength{\slideheight}{9cm}
106 ⟨/package⟩
107 ⟨*ltxml.sty⟩
108 DefRegister('\slidewidth'     => Dimension('13.6cm'));
109 DefRegister('\slideheight'     => Dimension('9cm'));
110 ⟨/ltxml.sty⟩
```

note The note environment is used to leave out text in the slides mode. It does not
have a counterpart in OMDoc. So for course notes, we define the note environment

to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```
111 ⟨∗package⟩
112 \ifnotes%
113   \renewenvironment{note}{\ignorespaces}{}%
114 \else%
115   \excludecomment{note}%
116 \fi%
117 ⟨/package⟩
118 ⟨∗ltxml.sty⟩
119 DefEnvironment('{note}','#body');
120 ⟨/ltxml.sty⟩
```

We start by giving the LATEXML binding for the `frame` environment from the `beamer` class. We first set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
121 ⟨∗package⟩
122 \ifnotes
123   \newlength{\slideframewidth}
124   \setlength{\slideframewidth}{1.5pt}
```

`frame`   We first define the keys.

```
125   \addmetakey{frame}{label}
126   \addmetakey[yes]{frame}{allowframebreaks}
127   \addmetakey{frame}{allowdisplaybreaks}
128   \addmetakey[yes]{frame}{fragile}
129   \addmetakey[yes]{frame}{shrink}
130   \addmetakey[yes]{frame}{squeeze}
```

We redefine the `itemize` environment so that it looks more like the one in `beamer` with Jacobs theme. We create the box with the `mdframed` environment from the equinymous package. Then we define the environment, read them, and construct the slide number and label.

```
131 \renewenvironment{frame}[1][]{%
132   \metasetkeys{frame}{#1}%
133   \stepcounter{slide}%
134   \def\@currentlabel{\theslide}%
135   \ifx\frame@label\@empty%
136   \else%
137     \label{\frame@label}%
138   \fi%
```

We redefine the `itemize` environment so that it looks more like the one in `beamer` with Jacobs theme.

```
139     \def\itemize@level{outer}%
140     \def\itemize@outer{outer}%
141     \def\itemize@inner{inner}%
142     \renewcommand\newpage{}%
143     \renewcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}%
```

7

```
144    \renewenvironment{itemize}{%
145      \ifx\itemize@level\itemize@outer%
146        \def\itemize@label{$\rhd$}%
147      \fi%
148      \ifx\itemize@level\itemize@inner%
149        \def\itemize@label{$\scriptstyle\rhd$}%
150      \fi%
151      \begin{list}%
152      {\itemize@label}%
153      {\setlength{\labelsep}{.3em}%
154       \setlength{\labelwidth}{.5em}%
155       \setlength{\leftmargin}{1.5em}%
156      }%
157      \edef\itemize@level{\itemize@inner}%
158    }{%
159      \end{list}%
160    }%
```

We create the box with the `mdframed` environment from the equinymous package.

```
161    \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=\s
162  }{%
163    \medskip\miko@slidelabel\end{mdframed}%
164  }%
165 ⟨/package⟩
166 ⟨∗ltxml.sty⟩
167 DefEnvironment('{frame}[]',
168    "<omdoc:omgroup layout='slide'>"
169    .  "#body\n"
170    ."</omdoc:omgroup>\n\n",
171  afterDigestBegin=>sub {
172    $_[1]->setProperty(theory=>LookupValue('current_module')); });
173 ⟨/ltxml.sty⟩#$
```

Now, we need to redefine the frametitle (we are still in course notes mode).

\frametitle

```
174 ⟨∗package⟩
175    \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}%
176 \fi %ifnotes
177 ⟨/package⟩
178 ⟨∗ltxml.sty⟩
179 DefConstructor('\frametitle{}',
180    "\n<omdoc:metadata><dc:title>#1</dc:title></omdoc:metadata>");
181 ⟨/ltxml.sty⟩
```

\frameimage We have to make sure that the width is overwritten, for that we check the
EdN:4   \Gin@ewidth macro from the graphicx package[4]

```
182 ⟨∗package⟩
```

---

[4]EDNOTE: MK@DG; we need to do that in the LaTeXML binding as well!

```
183 \newrobustcmd\frameimage[2][]{%
184   \stepcounter{slide}%
185   \ifframeimages%
186     \def\Gin@ewidth{}\setkeys{Gin}{#1}%
187     \ifnotes%
188     \else%
189       \vfill%
190     \fi%
191     \ifx\Gin@ewidth\@empty%
192       \mycgraphics[width=\slidewidth,#1]{#2}\else\mycgraphics[#1]{#2}%
193     \fi%
194     \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
195     \ifnotes%
196     \else%
197       \vfill%
198     \fi%
199   \fi%
200 }% ifframeimages
201 ⟨/package⟩
202 ⟨∗ltxml.sty⟩
203 DefMacro('\frameimage[]{}','\@frameimage{\includegrahics[#1,width=\slidewidth]{#2}}');
204 DefConstructor('\@frameimage{}',"<omdoc:omgroup layout='slide'>#1</omdoc:omgroup>\n");
205 ⟨/ltxml.sty⟩
```

4.3   Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

\setslidelogo The default logo is the logo of Jacobs University. Customization can be done by \setslidelogo{⟨logo name⟩}.

```
206 ⟨∗package⟩
207 \newlength{\slidelogoheight}
208 \ifnotes%
209   \setlength{\slidelogoheight}{.4cm}%
210 \else%
211   \setlength{\slidelogoheight}{1cm}%
212 \fi%
213 \newsavebox{\slidelogo}%
214 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{jacobs-logo}}%
215 \newrobustcmd{\setslidelogo}[1]{%
216   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}%
217 }%
```

\setsource \source stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. \setsource{⟨name⟩} can change the writer's name.

```
218 \def\source{Michael Kohlhase}% customize locally
```

```
219 \newrobustcmd{\setsource}[1]{\def\source{#1}}%
```

\setlicensing Now, we set up the copyright and licensing. By default we use the Creative Commons Attribuition-ShareAlike license to strengthen the public domain. If package hyperref is loaded, then we can attach a hyperlink to the license logo. \setlicensing[⟨*url*⟩]{⟨*logo name*⟩} is used for customization, where ⟨*url*⟩ is optional.

```
220 \def\copyrightnotice{\footnotesize\copyright:\hspace{.3ex}{\source}}%
221 \newsavebox{\cclogo}%
222 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}%
223 \newif\ifcchref\cchreffalse%
224 \AtBeginDocument{%
225   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
226 }%
227 \def\licensing{%
228   \ifcchref%
229     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}%
230   \else%
231     {\usebox{\cclogo}}%
232   \fi%
233 }%
234 \newrobustcmd{\setlicensing}[2][]{%
235   \def\@url{#1}%
236   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}%
237   \ifx\@url\@empty%
238     \def\licensing{{\usebox{\cclogo}}}%
239   \else%
240     \def\licensing{%
241   \ifcchref%
242     \href{#1}{\usebox{\cclogo}}%
243   \else%
244     {\usebox{\cclogo}}%
245   \fi%
246     }%
247   \fi%
248 }%
```

EdN:5 \slidelabel Now, we set up the slide label for the article mode.[5]

```
249 \newrobustcmd\miko@slidelabel{%
250   \vbox to \slidelogoheight{%
251     \vss\hbox to \slidewidth%
252     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}%
253   }%
254 }%
```

---

[5]EDNOTE: see that we can use the themes for the slides some day. This is all fake.

4.4 Colors and Highlighting

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to to is to adapt the green so that it is dark enough for most beamers

```
255 \AtBeginDocument{%
256   \definecolor{green}{rgb}{0,.5,0}%
257   \definecolor{purple}{cmyk}{.3,1,0,.17}%
258 }%
```

We customize the \defemph, \notemph, and \stDMemph macros with colors for the use in the statements package. Furthermore we customize the \@@lec macro for the appearance of line end comments in \lec.

```
259 % \def\STpresent#1{\textcolor{blue}{#1}}
260 \def\defemph#1{{\textcolor{magenta}{#1}}}
261 \def\notemph#1{{\textcolor{magenta}{#1}}}
262 \def\stDMemph#1{{\textcolor{blue}{#1}}}
263 \def\@@lec#1{(\textcolor{green}{#1})}
264 ⟨/package⟩
265 ⟨∗ltxml.sty⟩
266 #DefMacro('\defemph{}','{\textcolor{magenta}{#1}}');
267 #DefMacro('\notemph{}','{\textcolor{magenta}{#1}}');
268 ⟨/ltxml.sty⟩
```

I like to use the dangerous bend symbol for warnings, so we provide it here.

\textwarning as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```
269 ⟨∗package⟩
270 \pgfdeclareimage[width=.9em]{miko@small@dbend}{dangerous-bend}
271 \def\smalltextwarning{%
272   \pgfuseimage{miko@small@dbend}%
273   \xspace%
274 }%
275 \pgfdeclareimage[width=1.5em]{miko@dbend}{dangerous-bend}
276 \newrobustcmd\textwarning{%
277   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}%
278   \xspace%
279 }%
280 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}%
281 \newrobustcmd\bigtextwarning{%
282   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}%
283   \xspace%
284 }%
285 ⟨/package⟩
286 ⟨∗ltxml.sty⟩
287 DefMacro('\textwarning','\@textwarning\xspace');
288 DefConstructor('\@textwarning',"");
289 ⟨/ltxml.sty⟩
```

4.5 Front Matter, Titles, etc

We need to redefine the frontmatter macros inherited from the `beamer` class for LaTeXML, since there they take an optional argument.

```
290 ⟨∗ltxml.sty⟩
291 DefMacro('\title[]{}', '\@add@frontmatter{ltx:title}{#1}');
292 DefMacro('\date[]{}', '\@add@frontmatter{ltx:date}[role=creation]{#1}');
293 DefMacro('\author[]{}', sub { andSplit(T_CS('\@author'),$_[1]); });#$
294 ⟨/ltxml.sty⟩

295 %        Must be first command on slide to make positioning work.
296 ⟨∗package⟩
297 \newrobustcmd\putgraphicsat[3]{%
298   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}%
299 }%
300 \newrobustcmd\putat[2]{%
301   \begin{picture}(0,0)\put(#1){#2}\end{picture}%
302 }%
```

4.6 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define a set of counters

```
303 \ifsectocframes%
304   \if@part%
305     \newcounter{mpart}%
306     \newcounter{mchapter}%
307     \newcounter{msection}[mchapter]%
308   \else%
309     \newcounter{msection}%
310   \fi%
311   \newcounter{msubsection}[msection]%
312   \newcounter{msubsubsection}[msubsection]%
313   \newcounter{msubsubsubsection}[msubsubsection]%
314 \fi% ifsectocframes
```

and then

```
315 \ifnotes\else% only in slides
316   \renewenvironment{omgroup}[2][]{%
317     \metasetkeys{omgroup}{#1}\sref@target%
318     \advance\section@level by 1%
319     \ifsectocframes%
320     \begin{frame}%
321     \vfill\Large\centering%
322     \red{%
323      \ifcase\section@level\or%
324         \stepcounter{mpart}Part \Roman{mpart}\or%
325         \stepcounter{mchapter}Chapter \arabic{mchapter}\or
326         \stepcounter{msection}\if@part\arabic{mchapter}.\fi\arabic{msection}\or
327         \stepcounter{msubsection}\if@part\arabic{mchapter}.\fi\arabic{msection}.\arabic{msubsec
```

```
328        \stepcounter{msubsubsection}\if@part\arabic{mchapter}.\fi\arabic{msection}.\arabic{msub
329        \stepcounter{msubsubsubsection}\if@part\arabic{mchapter}.\fi\arabic{msection}.\arabic{m
330      \fi% end ifcase
331      \quad #2%
332    }%
333    \vfill%
334    \end{frame}%
335    \fi %ifsectocframes
336  }
337  {\advance\section@level by -1}%
338 \fi% ifnotes
339 ⟨/package⟩
```

4.7 Miscellaneous

The following fixes an error I do not understand, this has something to do with beamer compatibility, which has similar definitions but only up to 1.

```
340 ⟨∗package⟩
341 \expandafter\def\csname Parent2\endcsname{}
342 %     \begin{macrocode}
343 %
344 % We need to disregard the columns macros introduced by the |beamer| class
345 %     \begin{macrocode}
346 \ifnotes%
347   \renewenvironment{columns}{%
348     \par\noindent%
349     \begin{minipage}%
350     \slidewidth\centering\leavevmode%
351   }{%
352     \end{minipage}\par\noindent%
353   }%
354   \newsavebox\columnbox%
355   \renewenvironment{column}[1]{%
356     \begin{lrbox}{\columnbox}\begin{minipage}{#1}%
357   }{%
358     \end{minipage}\end{lrbox}\usebox\columnbox%
359   }%
360 \fi%
361 ⟨/package⟩
362 ⟨∗ltxml.sty⟩
363 DefEnvironment('{columns}',"#body");
364 DefEnvironment('{column}{}',"#body");
```

We also need to deal with overlay specifications introduced by the beamer class.[6]

[7]

───────────────

[6]EDNOTE: this is just to keep latexml quiet, no real functionality here.
[7]EDNOTE: Deyan: We reuse the CMP itemizations defined in the omdoc.cls.ltxml binding, adjusting the parameters to be overlay-sensitive

13

```
365 DefConstructor('\uncover','#1');
366 #Define a Beamer Overlay Parameter type
367  DefParameterType('BeamerOverlay', sub {
368    my ($gullet) = @_;
369    my $tok = $gullet->readXToken;
370    if (ref $tok && ToString($tok) eq '<') {
371      $gullet->readUntil(T_OTHER('>'));
372    } else {
373      $gullet->unread($tok) if ref $tok;
374      undef; }},
375      reversion=> sub {
376  (T_OTHER('<'), $_[0]->revert, T_OTHER('>'));
377         });
378
379 #Take the "from" field of the overlay range
380 sub overlayFrom {
381    return "" unless defined $_[0];
382    my $overlay=ToString($_[0]); $overlay =~ /^(\d+)/; $1;}
383
384 #Reuse the CMP itemizations, only adjust the \item constructors.
385 DefMacro('\beamer@group@item[] OptionalBeamerOverlay IfBeginFollows', sub {
386    my($gullet,$tag,$overlay,$needwrapper)=@_;
387    $overlay=$overlay||T_OTHER("");
388    ( T_CS('\group@item@maybe@unwrap'),
389      ($needwrapper ? (Invocation(T_CS('\beamer@group@item@wrap'),$tag,$overlay)->unlist) : ()) )
390 DefConstructor('\beamer@group@item@wrap {} OptionalBeamerOverlay',
391        "<omdoc:omtext ?#2(overlay='&overlayFrom(#2)')()>"
392        . "?#1(<dc:title>#1</dc:title>)()"
393              . "<omdoc:CMP>",
394        beforeDigest=>sub {
395  Let('\group@item@maybe@unwrap','\group@item@unwrap');
396 #$_[0]->bgroup;
397 return; },
398        properties=>sub{ RefStepItemCounter(); });
399 #DefConstructor('\beamer@itemize@item[] OptionalBeamerOverlay',
400 #        "<omdoc:li ?#2(overlay='&overlayFrom(#2)')() >"
401 #      . "?#1(<dc:title>#1</dc:title>)()",
402 #        properties=>sub{ RefStepItemCounter(); });
403 DefConstructor('\beamer@enumerate@item[] OptionalBeamerOverlay',
404        "<omdoc:li ?#2(overlay='&overlayFrom(#2)')() >"
405      . "?#1(<dc:title>#1</dc:title>)()",
406        properties=>sub{ RefStepItemCounter(); });
407 DefConstructor('\beamer@description@item[] OptionalBeamerOverlay',
408        "<omdoc:di ?#2(overlay='&overlayFrom(#2)')() >"
409      . "?#1(<omdoc:dt>#1</omdoc:dt>)()<omdoc:dd>", # trust di and dt to autoclose
410        properties=>sub{ RefStepItemCounter(); });
411 ⟨/ltxml.sty⟩#$
```

Now, some things that are imported from the pgf and beamer packages:

```
412 ⟨∗ltxml.sty⟩
```

```
413 DefMacro('\putgraphicsat{}{}{}','\mygraphics[#2]{#3}');
414 DefMacro('\putat{}{}','#2');
415 ⟨/ltxml.sty⟩

416 ⟨∗package⟩
417 \ifproblems%
418    \newenvironment{problems}{}{}%
419 \else%
420    \excludecomment{problems}%
421 \fi%
422 ⟨/package⟩
423 ⟨∗ltxml.sty⟩
424 DefEnvironment('{problems}','#body');
425 ⟨/ltxml.sty⟩
```

4.8   Finale

Finally, we set the slide body font to the sans serif, and we terminate the L^AT_EXML bindings file with a success mark for perl.

```
426 ⟨package⟩\ifnotes\else\sf\fi
427 ⟨ltxml.sty | ltxml.cls⟩1;
```