

Slides and Course Notes for Jacobs University*

Michael Kohlhasse
Jacobs University, Bremen
<http://kwarc.info/kohlhasse>

November 17, 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 L^AT_EX XML bindings for these, so that we can also generate OMDoc-based course materials, e.g. for inclusion in the ACTIVEMATH system.

Contents

1	Introduction	2
2	The User Interface	2
2.1	Package Options	2
2.2	Notes and Slides	2
2.3	Header and Footer Lines	3
2.4	Colors and Highlighting	3
2.5	Front Matter, Titles, etc	3
2.6	Miscellaneous	3
3	Limitations	3
4	The Implementation	4
4.1	Class and Package Options	4
4.2	Notes and Slides	6
4.3	Header and Footer Lines	9
4.4	Colors and Highlighting	10
4.5	Front Matter, Titles, etc	11
4.6	Sectioning	12
4.7	Miscellaneous	13
4.8	Finale	15

*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 \TeX and 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

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

- The options `slides` and `notes` switch between slides mode and notes mode (see Section 2.2).
- If the option `sectocframes` is given, then special frames with section table of contents are produced headers²
- `showmeta`. If this is set, then the metadata keys are shown (see [`Kohlhase:metakeys:ctan`] for details and customization options).
- If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames.

2.2 Notes and Slides

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.

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

²EDNOTE: document the functionality

¹MK: 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 `\endpackage`) and the \LaTeX XML bindings (between `\ltxml` and `\endltxml`). 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 \LaTeX XML:

```
1 \ltxml.cls | ltxml.sty
2 # -*- PERL -*-
3 package LaTeXXML::Package::Pool;
4 use strict;
5 use LaTeXXML::Package;
6 \endltxml.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}{\notestruet\PassOptionsToPackage{\CurrentOption}{mikoslides}}
10 \DeclareOption{slides}{\notestruet\PassOptionsToPackage{\CurrentOption}{mikoslides}}
11 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{omdoc}
12                                     \PassOptionsToClass{\CurrentOption}{beamer}
13                                     \PassOptionsToPackage{\CurrentOption}{mikoslides}}
14 \ProcessOptions
15 \endcls
16 \ltxml.cls
17 \DeclareOption(undef, sub {PassOptions('omdoc','cls',ToString(Digest(T_CS('\CurrentOption'))));
18                                     PassOptions('mikoslides','sty',ToString(Digest(T_CS('
19 \ProcessOptions();
20 \endltxml.cls
```

now we do the same for the `mikoslides` package. Note that we also have to define the same switches³, since we might use `mikoslides.sty` in a different class.

```
21 \package
22 \newif\if@mikoslides@mh@\@mikoslides@mh@false
23 \DeclareOption{mh}{\@mikoslides@mh@true}
24 \newif\ifnotes\notesfalse
25 \DeclareOption{notes}{\notestruet}
26 \DeclareOption{slides}{\notestruet}
```

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

```

27 \newif\ifsectocframes\sectocframesfalse
28 \DeclareOption{sectocframes}{\sectocframestrue}
29 \newif\ifframeimages\frameimagesfalse
30 \DeclareOption{frameimages}{\frameimagestrue}
31 \newif\if@part\@partfalse
32 \DeclareOption{report}{\@parttrue\PassOptionsToPackage{\CurrentOption}{omdoc}}
33 \DeclareOption{book}{\@parttrue\PassOptionsToPackage{\CurrentOption}{omdoc}}
34 \newif\ifproblems\problemstrue
35 \DeclareOption{nopproblems}{\problemsfalse}
36 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}
37 \PassOptionsToPackage{\CurrentOption}{tikzinput}}
38 \ProcessOptions
39 </package>
40 <*txml.sty>
41 DeclareOption('notes', '');
42 DeclareOption('slides', '');
43 DeclareOption('nopproblems', '');
44 DeclareOption('sectocframes', '');
45 DeclareOption('frameimages', '');
46 DeclareOption('mh', sub {RequirePackage('mikoslides-mh')});
47 DeclareOption(undef, sub {PassOptions('stex','sty',ToString(Digest(T_CS('CurrentOption'))));
48 PassOptions('tikzinput','sty',ToString(Digest(T_CS('
49 ProcessOptions();
50 RawTeX('\newif\ifnotes\notesfalse');
51 RawTeX('\newif\ifproblems\problemsfalse');
52 </txml.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. On the `LATEXML` side we just load the `omdoc` class and provide the `\usetheme` macro that would otherwise from the `beamer` class.

```

53 <*cls>
54 \ifnotes
55 \LoadClass{omdoc}
56 \RequirePackage{a4wide}
57 \RequirePackage{marginnote}
58 \RequirePackage{mdframed}
59 \RequirePackage[notheorems,noamsthm,noxcolor]{beamerarticle}
60 \else
61 \LoadClass[notheorems,noamsthm,10pt]{beamer}
62 \newcounter{Item}
63 \newcounter{paragraph}
64 \newcounter{subparagraph}
65 \newcounter{Hfootnote}
66 \usetheme{Jacobs}
67 \fi
68 \RequirePackage{mikoslides}
69 </cls>

```

```

70 <*ltxml.cls>
71 LoadClass('omdoc');
72 RequirePackage('mikoslides');
73 DefConstructor('\usetheme{','}');
74 </ltxml.cls>

    now, we load the remaining packages for both versions.

75 <*package>
76 \if@mikoslides@mh@RequirePackage{mikoslides-mh}\fi
77 \RequirePackage{stex}
78 \RequirePackage{smglom}
79 \RequirePackage{tikzinput}
80 \RequirePackage{latexml}
81 \RequirePackage{amssymb}
82 \RequirePackage{amsmath}
83 \RequirePackage{comment}
84 \RequirePackage{textcomp}
85 \RequirePackage{url}
86 </package>
87 <*ltxml.sty>
88 RequirePackage('stex');
89 RequirePackage('smglom');
90 RequirePackage('tikzinput', options => ['image']);
91 RequirePackage('latexml');
92 RequirePackage('amssymb');
93 RequirePackage('amsmath');
94 RequirePackage('graphicx');
95 RequirePackage('url');
96 </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.

```

97 <*package>
98 \newcounter{slide}
99 \newlength{\slidewidth}\setlength{\slidewidth}{12.8cm}
100 \newlength{\slideheight}\setlength{\slideheight}{9cm}
101 </package>
102 <*ltxml.sty>
103 DefRegister('\slidewidth'    => Dimension('13.6cm'));
104 DefRegister('\slideheight'  => Dimension('9cm'));
105 </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.

```

106 <*package>

```

```

107 \ifnotes%
108   \renewenvironment{note}{\ignorespaces}{}%
109 \else%
110   \excludcomment{note}%
111 \fi%
112 </package>
113 <*lxml.sty>
114 DefEnvironment('{note}', '#body');
115 </lxml.sty>

```

We start by giving the L^AT_EX 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.

```

116 <*package>
117 \ifnotes
118   \newlength{\slideframewidth}
119   \setlength{\slideframewidth}{1.5pt}

```

`frame` We first define the keys.

```

120   \addmetakey{frame}{label}
121   \addmetakey[yes]{frame}{allowframebreaks}
122   \addmetakey{frame}{allowdisplaybreaks}
123   \addmetakey[yes]{frame}{fragile}
124   \addmetakey[yes]{frame}{shrink}
125   \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.

```

126   \renewenvironment{frame}[1][]{%
127     \metasetkeys{frame}{#1}%
128     \stepcounter{slide}%
129     \def\@currentlabel{\theslide}%
130     \ifx\frame@label\@empty%
131     \else%
132       \label{\frame@label}%
133     \fi%

```

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

```

134   \def\itemize@level{outer}%
135   \def\itemize@outer{outer}%
136   \def\itemize@inner{inner}%
137   \renewcommand\newpage{}%
138   \renewcommand\metakeys@showkeys[2]{\marginnote{{\scriptsize ##2}}}%
139   \renewenvironment{itemize}{%
140     \ifx\itemize@level\itemize@outer%
141       \def\itemize@label{\$ \rhd $}%
142     \fi%

```

```

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

```

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

```

156     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=\s
157 ]{%
158     \medskip\miko@slidelabel\end{mdframed}%
159 }%
160 \end{package}
161 \end{*lxml.sty}
162 DefEnvironment('frame')[],
163     "<omdoc:omgroup layout='slide'>"
164     . "#body\n"
165     . "</omdoc:omgroup>\n\n",
166     afterDigestBegin=>sub {
167         $_[1]->setProperty(theory=>LookupValue('current_module')); });
168 \end{*lxml.sty}##

```

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

```

\frametitle
169 \begin{package}
170     \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip%
171 \end{package}
172 \end{package}
173 \end{*lxml.sty}
174 DefConstructor('\frametitle{}',
175     "\n<omdoc:metadata><dc:title>#1</dc:title></omdoc:metadata>");
176 \end{*lxml.sty}

```

EdN:4

`\frameimage` We have to make sure that the width is overwritten, for that we check the `\Gin@ewidth` macro from the `graphicx` package⁴

```

177 \begin{package}
178 \newrobustcmd\frameimage[2][]{%
179     \stepcounter{slide}%
180     \ifframeimages%
181         \def\Gin@ewidth{}\setkeys{Gin}{#1}%

```

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


```

182 \ifnotes%
183 \else%
184 \vfill%
185 \fi%
186 \ifx\Gin@ewidth\@empty%
187 \mycgraphics[width=\slidewidth,#1]{#2}\else\mycgraphics[#1]{#2}%
188 \fi%
189 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
190 \ifnotes%
191 \else%
192 \vfill%
193 \fi%
194 \fi%
195 }% iframeimages
196 \end{package}
197 \end{*ltxml.sty}
198 DefMacro('frameimage[]{}','\@frameimage{\includegraphics[#1,width=\slidewidth]{#2}}');
199 DefConstructor('\@frameimage{}','<omdoc:omgroup layout='slide'>#1</omdoc:omgroup>\n");
200 \end{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>}`.

```

201 \end{package}
202 \newlength{\slidelogoheight}
203 \ifnotes%
204 \setlength{\slidelogoheight}{.4cm}%
205 \else%
206 \setlength{\slidelogoheight}{1cm}%
207 \fi%
208 \newsavebox{\slidelogo}%
209 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{jacobs-logo}}%
210 \newrobustcmd{\setslidelogo}[1]{%
211 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}%
212 }%

```

`\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.

```

213 \def\source{Michael Kohlhase}% customize locally
214 \newrobustcmd{\setsource}[1]{\def\source{#1}}%

```

`\setlicensing` Now, we set up the copyright and licensing. By default we use the Creative Commons Attribution-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.

```

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

```

EdN:5

`\slidelabel` Now, we set up the slide label for the `article` mode.⁵

```

244 \newrobustcmd\miko@slidelabel{%
245   \vbox to \slidelogoheight{%
246     \vss\hbox to \slidewidth%
247       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}%
248   }%
249 }%

```

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 do is to adapt the green so that it is dark enough for most beamers

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

```

250 \AtBeginDocument{%
251   \definecolor{green}{rgb}{0,.5,0}%
252   \definecolor{purple}{cmk}{.3,1,0,.17}%
253 }%

```

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`.

```

254 % \def\STpresent#1{\textcolor{blue}{#1}}
255 \def\defemph#1{{\textcolor{magenta}{#1}}}
256 \def\notemph#1{{\textcolor{magenta}{#1}}}
257 \def\stDMemph#1{{\textcolor{blue}{#1}}}
258 \def\@@lec#1{(\textcolor{green}{#1})}
259 \end{package}
260 \ltxml.sty
261 #DefMacro('defemph','{\textcolor{magenta}{#1}}');
262 #DefMacro('notemph','{\textcolor{magenta}{#1}}');
263 \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.

```

264 \begin{package}
265 \pgfdeclareimage[width=.9em]{miko@small@dbend}{dangerous-bend}
266 \def\smalltextwarning{%
267   \pgfuseimage{miko@small@dbend}%
268   \xspace%
269 }%
270 \pgfdeclareimage[width=1.5em]{miko@dbend}{dangerous-bend}
271 \newrobustcmd\textwarning{%
272   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}%
273   \xspace%
274 }%
275 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}%
276 \newrobustcmd\bigtextwarning{%
277   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}%
278   \xspace%
279 }%
280 \end{package}
281 \ltxml.sty
282 DefMacro('textwarning','\@textwarning\xspace');
283 DefConstructor('@textwarning',"");
284 \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.

```

285 \ltxml.sty

```

```

286 DefMacro('\title[]{}', '\@add@frontmatter{ltx:title}{#1}');
287 DefMacro('\date[]{}', '\@add@frontmatter{ltx:date}[role=creation]{#1}');
288 DefMacro('\author[]{}', sub { andSplit(T_CS('\@author'),$_[1]); });#$
289 </ltxml.sty>

290 %      Must be first command on slide to make positioning work.
291 <*package>
292 \newrobustcmd\putgraphicsat[3]{%
293   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}%
294 }%
295 \newrobustcmd\putat[2]{%
296   \begin{picture}(0,0)\put(#1){#2}\end{picture}%
297 }%

```

4.6 Sectioning

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

```

298 \ifsectocframes%
299   \if@part%
300     \newcounter{mpart}%
301     \newcounter{mchapter}%
302     \newcounter{msection}[mchapter]%
303   \else%
304     \newcounter{msection}%
305   \fi%
306   \newcounter{msubsection}[msection]%
307   \newcounter{msubsubsection}[msubsection]%
308   \newcounter{msubsubsubsection}[msubsubsection]%
309 \fi% ifsectocframes

and then

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

```

```

328     \vfill%
329     \end{frame}%
330     \fi %ifsectocframes
331 }
332 {\advance\section@level by -1}%
333 \fi% ifnotes
334 \</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.

```

335 <*package>
336 \expandafter\def\csname Parent2\endcsname{
337 %     \begin{macrocode}
338 %
339 % We need to disregard the columns macros introduced by the |beamer| class
340 %     \begin{macrocode}
341 \ifnotes%
342 \renewenvironment{columns}{%
343     \par\noindent%
344     \begin{minipage}%
345     \slidewidth\centering\leavevmode%
346 }{%
347     \end{minipage}\par\noindent%
348 }%
349 \newsavebox\columnbox%
350 \renewenvironment{column}[1]{%
351     \begin{lrbox}{\columnbox}\begin{minipage}{#1}%
352 }{%
353     \end{minipage}\end{lrbox}\usebox\columnbox%
354 }%
355 \fi%
356 \</package>
357 <*ltxml.sty>
358 DefEnvironment('{columns}',"#body");
359 DefEnvironment('{column}{}',"#body");

```

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

7

```

360 DefConstructor('\uncover', '#1');
361 #Define a Beamer Overlay Parameter type
362 DefParameterType('BeamerOverlay', sub {
363     my ($gullet) = @_ ;
364     my $tok = $gullet->readXToken;

```

⁶EDNOTE: this is just to keep latexml quiet, no real functionality here.

⁷EDNOTE: Deyan: We reuse the CMP itemizations defined in the omdoc.cls.ltxml binding, adjusting the parameters to be overlay-sensitive

EdN:6
EdN:7

```

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

```

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

```

407 <*ltxml.sty>
408 DefMacro('\putgraphicsat{}{}{}','\mygraphics[#2]{#3}');
409 DefMacro('\putat{}{}','\#2');
410 </ltxml.sty>
411 <*package>
412 \ifproblems%

```

```

413 \newenvironment{problems}{}{}%
414 \else%
415 \excludecomment{problems}%
416 \fi%
417 \</package>
418 \<*ltxml.sty>
419 DefEnvironment('{problems}', '#body');
420 \</ltxml.sty>

```

4.8 Finale

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

```

421 \<package>\ifnotes\else\sf\fi
422 \<ltxml.sty | ltxml.cls>1;

```