

# Slides and Course Notes for Jacobs University\*

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

April 6, 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  $\text{\LaTeX}$ XML 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 `STEX` 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:<sup>1</sup>

- `slides` • The options `slidesnd` and `notesnotes` switch between slides mode and notes mode (see Section ??).
- `a` • If the option `sectocframes` is given, then special frames with section table of contents are produced headers<sup>2</sup>
- `sectocframes` • `showmeta`. If this is set, then the metadata keys are shown (see [`Kohlhase:metakeys:ctan`] for details and customization options).
- `showmeta` • If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames.
- `frameimages`

### 2.2 Notes and Slides

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

 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.

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

<sup>2</sup>EDNOTE: document the functionality

<sup>1</sup>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 ??.

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 `TeX`notes. 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

## 2.7 Support for MathHub

Much of the `TeX`content is hosted on MathHub (<http://MathHub.info>), a portal and archive for flexiformal mathematics. MathHub offers GIT repositories (public and private escrow) for mathematical documentation projects, online and offline authoring and document development infrastructure, and a rich, interactive reading interface. The `modules` package supports repository-sensitive operations on MathHub.

Note that **MathHub** has two-level repository names of the form  $\langle group \rangle / \langle repo \rangle$ , where  $\langle group \rangle$  is a **MathHub**-unique repository group and  $\langle repo \rangle$  a repository name that is  $\langle group \rangle$ -unique. The file and directory structure of a repository is arbitrary – except that it starts with the directory **source** because they are Math Archives in the sense of [HorIacJuc:cscpnrr11]. But this structure can be hidden from the  $\TeX$ author with **MathHub**-enabled versions of the **modules** macros.

**\mhframeimage** The **\mhframeimage** macro is a variant of **\frameimage** with repository support. Instead of writing

```
\defpath{MathHub}{/user/foo/lmh/MathHub}
\frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that **\MathHub** is defined as above)

```
\mhframeimage[fooMH/bar]{baz/foobar}
```

Note that the **\mhframeimage** form is more semantic, which allows more advanced document management features in **MathHub**.

If **baz/foobar** is the “current module”, i.e. if we are on the **MathHub** path **...MathHub/fooMH/bar...**, then stating the repository in the first optional argument is redundant, so we can just use

```
\mhframeimage{baz/foobar}
```

Of course, neither  $\LaTeX$  nor  $\LaTeX$ ML know about the repositories when they are called from a file system, so we can use the **\mhcurrentrepos** macro from the **modules** package to tell them. But this is only needed to initialize the infrastructure in the driver file. In particular, we do not need to set it in each module, since the **\importmhmodule** macro sets the current repository automatically.

**Caveat** if you want to use the **MathHub** support macros (let’s call them **mh**-variants), then every time a module is imported or a document fragment is included from another repos, the **mh**-variant **\importmhmodule** must be used, so that the “current repository” is set accordingly. To be exact, we only need to use **mh**-variants, if the imported module or included document fragment use **mh**-variants.

### 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  $\TeX$ TRAC [sTeX:online].

1. the class should be divided into concerns. [sTeX:online], issue 1684
2. when option **book** or **report** is given together with **sectocframes** chapter-level omgroups generate a spurious slide with a bare heading. This has something to do with the fact that beamer does not support **\chapter**

## 4 The Implementation

The `mikoslides` package generates two files: the  $\text{\LaTeX}$  package (all the code between `\package` and `\endpackage`) and the  $\text{\LaTeX}$ XML bindings (between `\beginltxml` 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.

### 4.1 Initialization and Class Options

For the  $\text{\LaTeX}$ XML bindings, we make sure the right perl packages are loaded.

```
1 \beginltxml
2 # -*- PERL -*-
3 package LaTeXXML::Package::Pool;
4 use strict;
5 use LaTeXXML::Package;
6 \endltxml
```

For  $\text{\LaTeX}$  we define some Package Options and switches for the `mikoslides` class and activate them by passing them on to `beamer.cls` the appropriate packages.

```
7 \begin{cls}
8 \DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
9 \DeclareOption{defindex}{\PassOptionsToPackage{\CurrentOption}{statements}}
10 \newif\ifnotes\notesfalse
11 \newif\ifsectocframes\sectocframesfalse
12 \newif\ifframeimages\frameimagesfalse
13 \newif\ifproblems\problemstrue
14 \DeclareOption{notes}{\notestrue}
15 \DeclareOption{slides}{\notesfalse}
16 \DeclareOption{nopproblems}{\problemsfalse}
17 \DeclareOption{sectocframes}{\sectocframestrue}
18 \DeclareOption{frameimages}{\frameimagestrue}
```

the next two define the `frontmatter` environment so that the later `\renewcommand` does not lead to trouble.

```
19 \newif\if@part\@partfalse
20 \DeclareOption{report}{\@parttrue\PassOptionsToClass{\CurrentOption}{omdoc}}
21 \DeclareOption{book}{\@parttrue\PassOptionsToClass{\CurrentOption}{omdoc}}
22 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{omdoc}
23               \PassOptionsToClass{\CurrentOption}{beamer}}
24 \ProcessOptions
25 \end{cls}
26 \beginltxml
27 RawTeX('\newif\ifnotes\notesfalse');
28 RawTeX('\newif\ifproblems\problemsfalse');
29 \endltxml
```

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  $\text{\LaTeX}$  packages.

```

30 <*cls>
31 \ifnotes
32 \LoadClass{omdoc}
33 \RequirePackage{a4wide}
34 \RequirePackage{marginnote}
35 \RequirePackage{mdframed}
36 \RequirePackage[notheorems,noamsthm,noxcolor]{beamerarticle}
37 \else
38 \LoadClass[notheorems,noamsthm,10pt]{beamer}
39 \newcounter{Item}
40 \newcounter{paragraph}
41 \newcounter{subparagraph}
42 \newcounter{Hfootnote}
43 \usetheme{Jacobs}
44 \fi
45 </cls>
46 <*ltxml>
47 LoadClass('omdoc');
48 RequirePackage('tikzinput');
49 DefConstructor('\usetheme{','}');
50 </ltxml>

```

EdN:3

now, we load the remaining packages for both versions. <sup>3</sup>

```

51 <*cls>
52 \RequirePackage{stex}
53 \RequirePackage{tikzinput}
54 \RequirePackage{latexml}
55 \RequirePackage{amssymb}
56 \usepgflibrary{shapes}
57 \usetikzlibrary{arrows}
58 \usetikzlibrary{positioning}
59 \usetikzlibrary{tikzmark}%experimental/beta but very useful
60 \usetikzlibrary{fit}
61 \RequirePackage{url}
62 \RequirePackage{amsmath}
63 \RequirePackage{comment}
64 \RequirePackage{standalone}
65 \RequirePackage{textcomp}
66 </cls>
67 <*ltxml>
68 RequirePackage('stex');
69 RequirePackage('latexml');
70 RequirePackage('amssymb');
71 RequirePackage('graphicx');

```

---

<sup>3</sup>EDNOTE: MK: eventually (when `tikz` support is fully realized in  $\text{\LaTeXML}$ ) get rid of the `standalone` package

```

72 \RequirePackage('tikz');
73 \RequirePackage('url');
74 \RequirePackage('amsmath');
75 \end{document}

```

## 4.2 Notes and Slides

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

```

76 \setlength{\slideheight}{9cm}
77 \setlength{\slidewidth}{12.8cm}
78 \newlength{\slideheight}\setlength{\slideheight}{9cm}
79 \newlength{\slidewidth}\setlength{\slidewidth}{12.8cm}
80 \end{document}
81 \end{document}
82 \DefRegister('slidewidth' => Dimension('13.6cm'));
83 \DefRegister('slideheight' => Dimension('9cm'));
84 \end{document}

```

**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.

```

85 \setlength{\slideheight}{9cm}
86 \ifnotes\renewenvironment{note}{\ignorespaces}{\else\excludecomment{note}\fi}
87 \end{document}
88 \end{document}
89 \DefEnvironment('{note}','body');
90 \end{document}

```

We start by giving the L<sup>A</sup>T<sub>E</sub>X 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.

```

91 \setlength{\slideframewidth}{1.5pt}
92 \ifnotes
93 \newlength{\slideframewidth}\setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

94 \addmetakey{frame}{label}
95 \addmetakey[yes]{frame}{allowframebreaks}
96 \addmetakey{frame}{allowdisplaybreaks}
97 \addmetakey[yes]{frame}{fragile}
98 \addmetakey[yes]{frame}{shrink}
99 \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.

```

100 \renewenvironment{frame}[1] []%
101 {\metasetkeys{frame}{#1}%
102 \stepcounter{slide}\def\@currentlabel{\theslide}%
103 \ifx\frame@label\empty\else\label{\frame@label}\fi

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

104 \def\itemize@level{outer}%
105 \def\itemize@outer{outer}%
106 \def\itemize@inner{inner}%
107 \renewcommand\newpage{}%
108 \renewcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}%
109 \renewenvironment{itemize}%
110 {\ifx\itemize@level\itemize@outer\def\itemize@label{\$ \rhd$}\fi%
111 \ifx\itemize@level\itemize@inner\def\itemize@label{\$ \scriptstyle\rhd$}\fi%
112 \begin{list}%
113   {\itemize@label}%
114   {\setlength{\labelsep}{.3em}\setlength{\labelwidth}{.5em}\setlength{\leftmargin}{1.5em}}%
115   \edef\itemize@level{\itemize@inner}}%
116 {\end{list}}

We create the box with the mdframed environment from the equinymous package.
117 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,
118 userdefinedwidth=\slidewidth,align=center]\sf}
119 {\medskip\miko@slidelabel\end{mdframed}}
120 \end{cls}
121 \ltxml
122 DefEnvironment('frame'[],
123   "<omdoc:omgroup layout='slide'>"
124   . "#body\n"
125   . "</omdoc:omgroup>\n\n",
126   afterDigestBegin=>sub {
127     $_[1]->setProperty(theory=>LookupValue('current_module')); });
128 \ltxml)#$

```

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

\frametitle

```

129 \end{cls}
130 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
131 \fi
132 \end{cls}
133 \ltxml
134 DefConstructor('\frametitle{}',
135   "\n<omdoc:metadata><dc:title>#1</dc:title></omdoc:metadata>");
136 \ltxml)

```

\frameimage

We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package<sup>4</sup>

<sup>4</sup>EDNOTE: MK@DG; we need to do that in the LaTeXML binding as well!



```

137 <*cls>
138 \newcommand\frameimage[2] [] {\stepcounter{slide}%
139 \ifframeimages%
140 \def\Gin@ewidth{}\setkeys{Gin}{#1}%
141 \ifnotes\else\vfill\fi%
142 \ifx\Gin@ewidth\@empty\mycgraphics[width=\slidewidth,#1]{#2}\else\mycgraphics[#1]{#2}\fi
143 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
144 \ifnotes\else\vfill\fi%
145 \fi}%\ifframeimages
146 </cls>
147 <*txml>
148 DefMacro(' \frameimage[] {}', '@frameimage{\includegraphics[#1,width=\slidewidth]{#2}}');
149 DefConstructor('@frameimage{}', "<omdoc:omgroup layout='slide'>#1</omdoc:omgroup>\n");
150 </txml>

```

### 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.

```

151 <*cls>
152 \newlength{\slidelogoheight}
153 \ifnotes\setlength{\slidelogoheight}{.4cm}\else\setlength{\slidelogoheight}{1cm}\fi
154 \newsavebox{\slidelogo}\sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{jacobs-logo}}

```

Now, we set up the copyright and licensing, the copyright remains with the author, but we use the Creative Commons Attribution-ShareAlike license to strengthen den public domain. Here the problem is that we want a hyperref on the CC logo, if hyperref is loaded, and otherwise not. As hyperref is always loaded, we have to find out at the beginning of the document whether it is, set up a switch, and later in the footer line decide what to do.

```

155 \def\source{Michael Kohlhase}% customize locally
156 \def\copyrightnotice{\footnotesize\copyright:\hspace{.3ex}{\source}}
157 \newsavebox{\cclogo}\sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
158 \newif\ifcchref\cchreffalse
159 \AtBeginDocument{\ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}}
160 \def\licensing{\ifcchref\href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}

```

EdN:5

Now, we set up the slide label for the article mode<sup>5</sup>

\slidelabel

```

161 \newcommand\miko@slidelabel%
162 {\vbox to \slidelogoheight{\vss\hbox to \slidewidth%
163 {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}}}

```

---

<sup>5</sup>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 do is to adapt the green so that it is dark enough for most beamers

```
164 \AtBeginDocument{\definecolor{green}{rgb}{0,.5,0}\definecolor{purple}{cmyk}{.3,1,0,.17}}
```

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`.

```
165 % \def\STpresent#1{\textcolor{blue}{#1}}
166 \def\defemph#1{{\textcolor{magenta}{#1}}}
167 \def\notemph#1{{\textcolor{magenta}{#1}}}
168 \def\stdMemph#1{{\textcolor{blue}{#1}}}
169 \def\@@lec#1{(\textcolor{green}{#1})}
170 \<cls>
171 \<*txml>
172 #DefMacro('defemph{ }', '{\textcolor{magenta}{#1}}');
173 #DefMacro('notemph{ }', '{\textcolor{magenta}{#1}}');
174 \</txml>
```

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.

```
175 \<cls>
176 \pgfdeclareimage[width=.9em]{miko@small@dbend}{dangerous-bend}
177 \def\smalltextwarning{\pgfuseimage{miko@small@dbend}\xspace}
178 \pgfdeclareimage[width=1.5em]{miko@dbend}{dangerous-bend}
179 \newrobustcmd\textwarning{\raisebox{-.05cm}{\pgfuseimage{miko@dbend}}\xspace}
180 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
181 \newrobustcmd\bigtextwarning{\raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}\xspace}
182 \</cls>
183 \<*txml>
184 DefMacro('textwarning', '@textwarning\xspace');
185 DefConstructor('@textwarning', "");
186 \</txml>
```

## 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.

```
187 \<*txml>
188 DefMacro('title[]{}', '@add@frontmatter{ltx:title}{#1}');
189 DefMacro('date[]{}', '@add@frontmatter{ltx:date}[role=creation]{#1}');
190 DefMacro('author[]{}', sub { andSplit(T_CS('@author'),$_[1]); });#$
191 \</txml>
```

Now, we specialize the slide environment that we have implemented above or inherited from `seminar.cls` for some abbreviations, e.g. separator slides and title slides.

```

192 <*cls>
193 \ifnotes\newcommand\titleframe{\maketitle}\else
194 \newcommand\titleframe{\begin{frame}\titlepage\end{frame}}\fi
195 \newenvironment{titleframewith}{\begin{frame}\titlepage}{\end{frame}}
196 \newenvironment{ttitle}{\begin{center}\LARGE\begin{tabular}{|c|}\hline}%
197 {\hline\end{tabular}\end{center}\vspace{1ex minus 1ex}}
198 \newenvironment{ttitlejoint}[1]%
199 {\newbox\boxwith\setbox\boxwith\hbox{\begin{tabular}{c}{\em joint work with}\#1\end{tabular}}}%
200 \begin{center}\LARGE\begin{tabular}{c}\color{red}}%
201 {\box\boxwith\end{tabular}\end{center}%
202 \vspace{1ex minus 1ex}}
203 </cls>
204 <*ltxml>
205 DefConstructor('\titleframe',"<omdoc:ignore>titleframe elided here</omdoc:ignore>");
206 DefEnvironment('{titleframewith}',
207               "<omdoc:ignore>begin elided titleframe</omdoc:ignore>"
208               . "#body"
209               . "<omdoc:ignore>end elided titleframe</omdoc:ignore>");
210 DefEnvironment('{titleslide}', "");
211 DefEnvironment('{titleslide}', "<omdoc:omgroup>#body</omdoc:omgroup>");
212 DefEnvironment('{ttitle}', "\n<dc:title>#body</dc:title>");
213 </ltxml>

214 %      Must be first command on slide to make positioning work.
215 <*cls>
216 \newcommand\putgraphicsat[3]{%
217   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}}
218 \newcommand\putat[2]{\begin{picture}(0,0)\put(#1){#2}\end{picture}}
219 </cls>

```

## 4.6 Sectioning

If the `sectocframes` option is set, then we make section frames.

```

220 <*cls>
221 \ifsectocframes
222 \if@part\newcounter{mpart}
223 \newcounter{mchapter}
224 \newcounter{msection}[mchapter]
225 \else
226 \newcounter{msection}
227 \fi
228 \newcounter{msubsection}[msection]
229 \newcounter{msubsubsection}[msubsection]
230 \newcounter{msubsubsubsection}[msubsubsection]
231 \ifnotes\else% only in slides
232 \renewcommand\at@begin@omgroup[3][]{\begin{frame}%

```

```

233 \vfill\Large\centering
234 \red{\ifcase\section@level\or
235 \stepcounter{mpart}Part \Roman{mpart}\or%
236 \stepcounter{mchapter}Chapter \arabic{mchapter}\or
237 \stepcounter{msection}\if@part\arabic{mchapter}.\fi\arabic{msection}\or
238 \stepcounter{msubsection}\if@part\arabic{mchapter}.\fi\arabic{msection}.\arabic{msubsection}\or
239 \stepcounter{msubsubsection}\if@part\arabic{mchapter}.\fi\arabic{msection}.\arabic{msubsection}
240 \stepcounter{msubsubsubsection}\if@part\arabic{mchapter}.\fi\arabic{msection}.\arabic{msubsecti
241 \quad #3}\vfill
242 \end{frame}}
243 \fi% ifnotes
244 \fi% ifsectocframes
245 \</cls>

```

## 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.

```

246 \<cls>
247 \expandafter\def\csname Parent2\endcsname{
248 % \begin{macrocode}
249 %
250 % We need to disregard the columns macros introduced by the |beamer| class
251 % \begin{macrocode}
252 \ifnotes
253 \renewenvironment{columns}%
254 {\par\noindent\begin{minipage}\slidewidth\centering\leavevmode}%
255 {\end{minipage}\par\noindent}
256 \newsavebox\columnbox
257 \renewenvironment{column}[1]%
258 {\begin{lrbox}{\columnbox}\begin{minipage}{#1}}%
259 {\end{minipage}\end{lrbox}\usebox\columnbox}
260 \fi
261 \</cls>
262 \<!xml>
263 DefEnvironment('{columns}','#body');
264 DefEnvironment('{column}{}','#body');

```

We also need to deal with overlay specifications introduced by the beamer class.<sup>6</sup>

```

265 DefConstructor('\uncover', '#1');
266 #Define a Beamer Overlay Parameter type
267 DefParameterType('BeamerOverlay', sub {
268 my ($gullet) = @_ ;
269 my $tok = $gullet->readXToken;

```

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

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

```

270   if (ref $tok && ToString($tok) eq '<') {
271     $gullet->readUntil(T_OTHER('>'));
272   } else {
273     $gullet->unread($tok) if ref $tok;
274     undef; }},
275     reversion=> sub {
276 (T_OTHER('<'), $_[0]->revert, T_OTHER('>'));
277     });
278
279 #Take the "from" field of the overlay range
280 sub overlayFrom {
281   return "" unless defined $_[0];
282   my $overlay=ToString($_[0]); $overlay =~ /\d+/; $1;
283
284 #Reuse the CMP itemizations, only adjust the \item constructors.
285 DefMacro('\beamer@group@item[] OptionalBeamerOverlay IfBeginFollows', sub {
286   my($gullet,$tag,$overlay,$needwrapper)=@_;
287   $overlay=$overlay||T_OTHER("");
288   ( T_CS('\group@item@maybe@unwrap'),
289     ($needwrapper ? (Invocation(T_CS('\beamer@group@item@wrap'),$tag,$overlay)->unlist) : ()))
290 DefConstructor('\beamer@group@item@wrap {} OptionalBeamerOverlay',
291   "<omdoc:omtext ?#2(overlay='&overlayFrom(#2)')()>"
292   . "?#1(<dc:title>#1</dc:title>())"
293   . "<omdoc:CMP>",
294   beforeDigest=>sub {
295 Let('\group@item@maybe@unwrap','\group@item@unwrap');
296   $_[0]->bgroup;
297 return; },
298   properties=>sub{ RefStepItemCounter(); });
299 #DefConstructor('\beamer@itemize@item[] OptionalBeamerOverlay',
300 #   "<omdoc:li ?#2(overlay='&overlayFrom(#2)')() >"
301 #   . "?#1(<dc:title>#1</dc:title>())",
302 #   properties=>sub{ RefStepItemCounter(); });
303 DefConstructor('\beamer@enumerate@item[] OptionalBeamerOverlay',
304   "<omdoc:li ?#2(overlay='&overlayFrom(#2)')() >"
305   . "?#1(<dc:title>#1</dc:title>())",
306   properties=>sub{ RefStepItemCounter(); });
307 DefConstructor('\beamer@description@item[] OptionalBeamerOverlay',
308   "<omdoc:di ?#2(overlay='&overlayFrom(#2)')() >"
309   . "?#1(<omdoc:dt>#1</omdoc:dt>())<omdoc:dd>", # trust di and dt to autoclose
310   properties=>sub{ RefStepItemCounter(); });
311 </lxml>#$

```

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

```

312 <*lxml>
313 DefMacro('\putgraphicsat{}{}{}','\mygraphics[#2]{#3}');
314 DefMacro('\putat{}{}','\#2');
315 </lxml>
316 <*cls>
317 \ifproblems\newenvironment{problems}{}{}\else\excludecomment{problems}\fi

```

```

318 </cls>
319 <*ltxml>
320 DefEnvironment('{problems}', '#body');
321 </ltxml>

```

## 4.8 Support for MathHub

`\mhframeimage` Use the current value of `\mh@currentrepos` or the value of the `mhrepos` key if it is given in `\frameimage`.

```

322 <cls>\addmetakey{Gin}{mhrepos}
323 <ltxml>DefKeyVal('Gin', 'mhrepos', 'Semiverbatim');
324 <ltxml>RawTeX('
325 <*ltxml | cls>
326 \newcommand\mhframeimage[2][\metasetkeys{Gin}{#1}%
327 \edef\mh@@repos{\mh@currentrepos}%
328 \ifx\Gin@mhrepos\empty\frameimage[#1]{\MathHub{\mh@currentrepos/source/#2}}}%
329 \else\frameimage[#1]{\MathHub{\Gin@mhrepos/source/#2}}\fi}
330 </ltxml | cls>
331 <ltxml>');

```

## 4.9 Finale

Finally, we set the slide body font to the sans serif, and we terminate the  $\text{\LaTeX}$ ML bindings file with a success mark for perl.

```

332 <cls>\ifnotes\else\sffamily
333 <ltxml>1;

```