

PORTABLE DOCUMENT MODES

VO.5.1 2026-09-27

Extensible mode graphs for L^AT_EX documents

Matthias WERNER¹

<https://github.com/werner-matthias/osglecture>

L^AT_EX packages such as [beamer](#)² and [ltx-talk](#)³ offer the concept of *modes*, in which, depending on the document mode, only specific parts of the document are rendered.

This package extends this concept and manages document modes as directed acyclic graphs. It complements the mode interfaces of [beamer](#) and [ltx-talk](#), but also works with standard document classes. It is part of the [osglecture](#) bundle, but can also be used independently of it.

Table of Contents

1. Quick start	2	6. Filter modes	7
		6.1. Registering exceptions	8
2. Modes and inheritance	2	6.2. Additional package for ltx-talk	9
2.1. Custom and simultaneous modes	3	7. Defining mode-dependent com-	
2.2. When to declare modes	3	mands	10
3. Selecting content and values	3	8. Limitations	11
3.1. Mode-dependent values	4	A. Bundle integration: a custom	
4. Configuration reference	5	document type	12
4.1. Package options	5	A.1. Registering the target	12
4.2. Declaration and selection	5	A.2. Declaring mode and behavior	12
5. Backend interaction	6	A.3. Setting up the document profile	13
5.1. Making custom commands		Index	14
overlay-aware	7		

1. matthias.werner@informatik.tu-chemnitz.de

1. Quick start

Users of Beamer or [ltx-talk](#) will recognize mode specifications such as `<presentation>` or `<handout>`. [osglecture-modes](#) makes document modes available in other classes as well and allows custom modes with multiple parent modes. One source can thus contain slides, a handout, and a script.

```

1 \documentclass{article}
2 \usepackage[mode=script]{osglecture-modes}
3 \begin{document}
4 Common text.
5 \lecturemode<print>{Text for printed documents.}
6 \lecturemode<presentation>{Text for presentations.}
7 \IfLectureModeTF{script}{Detailed explanation.}{Short explanation.}
8 \end{document}

```

The example prints the common text, the text for printed documents, and the detailed explanation. Presentation text is omitted: `script` activates `print`, but not `presentation`. Selecting `mode=handout` would print both variants and the short explanation instead. The package option does not change the document class.

The core package needs a current L^AT_EX kernel. It works independently of [osglecture](#), OLLM, and project directories. Normally, load it after the document class. Without an explicit mode selection it detects Beamer and [ltx-talk](#); for other classes it selects `article`. Only the portable text filters in Section 6 require LuaL^AT_EX.

2. Modes and inheritance

Selecting a mode also activates its parents, their parents, and so on. Multiple parents are allowed, cycles are not. Unlike a simple tree, this lets `handout` belong to both `presentation` and `print`. The following modes are predefined; the table lists their direct parents.

Mode	Parents
<code>presentation</code>	—
<code>beamer</code> , <code>projector</code> , <code>trans</code>	<code>presentation</code>
<code>handout</code>	<code>presentation</code> , <code>print</code>
<code>longform</code> , <code>print</code>	—
<code>article</code>	<code>longform</code> , <code>print</code>
<code>script</code>	<code>article</code>
<code>web</code>	<code>longform</code>
<code>*</code>	—

`slides` is an alias for `projector`, not a separate mode. `all` is an always-true query. Mode `*` is used for text filtering (Section 6).

2.1. Custom and simultaneous modes

```

1 \documentclass{article}
2 \usepackage{osglecture-modes}
3 \DeclareLectureMode[parents={article,print}]{reading}
4 \DeclareLectureModeAlias{reader}{reading}
5 \SetLectureModes{reader}
6 \FinalizeLectureModes
7 \begin{document}
8 \lecturemode<reading>{Reading edition.}
9 \lecturemode<longform>{Extended explanation.}
10 \end{document}

```

`\SetLectureModes` replaces the previous selection; `\AddLectureModes` extends it. Both accept comma-separated lists. For instance, selecting `slides,web` activates `projector`, `presentation`, `web`, and `longform` together. This does not change the document class or its native output options.

2.2. When to declare modes

Declare modes, parents, and aliases and select the active modes before the first query.

`\FinalizeLectureModes` validates the entire graph, including inactive modes, and closes this configuration. The first query using `\IfLectureModeTF`, `\IfLectureModeT`, `\IfLectureModeF`, or a `\lecturemode` instruction with a specification does this automatically if necessary; finalization takes place before `\begin{document}` at the latest. Afterwards, further declarations and changes to the mode selection are no longer allowed. Activating previously declared filter modes in the document is a separate operation; see Section 6.

3. Selecting content and values

`\lecturemode<expression>{<content>}`

Prints the content if the expression is true. Without a specification the content is always printed.

`\IfLectureModeTF{<expression>}{<true>}{<false>}`

Executes the second or third argument according to the query result.

`\IfLectureModeT{<expression>}{<true>}`

Executes the second argument only if the query is true.

`\IfLectureModeF{<expression>}{<false>}`

Executes the second argument only if the query is false.

These queries accept a mode name or several names separated by `|` or commas (disjunction). Aliases are allowed; unknown names cause an error. `all` always matches. A single `|`, `overlay`

3. Selecting content and values

numbers, negation, and conjunction are not part of this query syntax. Nest queries to express combined conditions. Native overlay commands use different rules for commas and |; see Section 5.

```
1 \lecturemode<article||handout>{Material for readers.}
2 \IfLectureModeT{print}{\IfLectureModeF{handout}{Extended printed text.}}
```

3.1. Mode-dependent values

`\ModeValue{<specification>}`

Expands to a selected value. The specification consists of entries of the form `mode=value`, separated by |. Use `default=value` or an entry without = for the fallback. With no match and no fallback the result is empty.

```
1 % With graphicx loaded:
2 \includegraphics[width=\ModeValue{5cm|handout=8cm}]{plot}
```

The order of entries in the specification does not determine the result. `\ActiveLectureModes` supplies the priority: parents precede their children, selected modes are traversed in selection order, and each mode occurs only once. The first mode with an entry wins; default is last. Use canonical names here, for example `projector` rather than the alias `slides`.

```
1 % mode=script: ActiveLectureModes = longform,print,article,script
2 \ModeValue{script=8cm|print=6cm|default=5cm} % -> 6cm
3 \ModeValue{script=8cm|default=5cm}           % -> 8cm
```

`\ModeValue` is expandable and can be used in length and register assignments. It does not finalize the graph itself: call `\FinalizeLectureModes` first when using it in the preamble. Values may contain further equals signs and colons; | serves as the separator. Use at most one fallback entry.

4. Configuration reference

4.1. Package options

`mode = {⟨name⟩}` Default: auto
 Selects a mode already known at load time. Declare custom modes after loading and select them with `\SetLectureModes`. With `auto`, Beamer selects `handout`, `beamer` in `handout` mode and `slides`, `beamer` otherwise; `ltx-talk` selects `handout`, `article`, or `slides` as appropriate. Other classes select `article`.

`full = true|false` Default: true
 Enables integration into existing mode and overlay commands. `full=false` leaves these native commands unchanged. Portable queries remain available. Missing commands may still be supplied.

`patch-overlays = true|false` Default: true
 The same setting as `full`; if specified more than once, the last setting wins.

`ignorenonframetext = true|false` Default: false
 Enables filtering with `*` at the start of the document. Naming the option without a value means `true`.

`raw-skip-envs = {⟨list⟩}`
 Replaces the list of environments to skip verbatim while filtering. Default: `verbatim`, `Verbatim`, `lstlisting`, `minted`, `comment`. See Section 6.

`deferred = true|false` Default: false
 For class authors: suppresses automatic mode selection and overlay integration at load time. The class selects the mode itself and calls the activation command described below after loading its base class.

4.2. Declaration and selection

`\DeclareLectureMode[⟨parents=⟨list⟩⟩]{⟨mode⟩}`
 Declares a mode, optionally with comma-separated parents. `parent` is a synonym for `parents`. Parents must be declared by finalization; use their canonical names.

`\DeclareLectureModeParents{⟨mode⟩}{⟨list⟩}`
 Replaces the complete parent list rather than appending to it. Also declares the mode if needed.

`\DeclareLectureModeAlias{⟨alias⟩}{⟨mode⟩}`
 Declares an alternative name. Alias chains are allowed; alias cycles and unknown final targets are not.

`\SetLectureModes{⟨list⟩}`
 Replaces the previous selection with the given list.

`\AddLectureModes{<list>}`

Extends the selection. Modes must already be declared; aliases are resolved and duplicates removed.

`\FinalizeLectureModes`

Validates and finalizes the graph. Repeated calls are allowed.

`\LectureModes`

Expands to the comma-separated selection without inherited parents and with aliases resolved.

`\ActiveLectureModes`

Expands to the active list including parents. Before finalization this list has not yet been computed. Neither list command finalizes the graph. The order also determines `\ModeValue`.

`\LectureModesActivateOverlays`

Integrates overlay commands after loading the base class. Call this once only when using `deferred=true`. It does not select a mode.

5. Backend interaction

If the loaded class has no `\mode` command, the package provides its portable inline form. If a native implementation already exists, the package handles only inline expressions composed of registered modes. The patched overlay commands and decoders of Beamer and `ltx-talk` likewise sort an angle-bracket specification into the following kinds:

- *portable* – a `|`-separated disjunction of registered modes (or `all`). Evaluated entirely by the package; works without any overlay backend.
- *mode-qualified* – every `|`-separated segment is a registered mode, a `mode:tail` form, or a purely numeric overlay atom. The package evaluates the portable parts and recombines the surviving native tails into a single backend specification. This lets `<slides:2-handout:0>` be decided portably while only the matching overlay range reaches the backend.
- *native* – everything else: backend mode names, `|` mode scoping over unknown modes, action specifications and native mode switches with unknown names. Delegated to the backend unchanged.

For backend commands, `|` and `,` inside `<...>` are always backend syntax; `||` is the only portable disjunction operator, so an expression that mixes portable modes with native overlay atoms must signal this with `||` or the `mode:tail` form. When the loaded class has no overlay backend, a native or unrecognised specification degrades to *show*. All of these patches can be turned off with `patch-overlays=false`.

5.1. Making custom commands overlay-aware

`\MakeOverlayAwareCommand` [*options*] {*command*}

Adds an angle-bracket specification immediately after an existing command. Registered modes are evaluated portably; genuine overlays are evaluated by the available backend.

`arguments` = {*signature*}

Default: empty

Argument signature of the original command in `\NewDocumentCommand` syntax, for example `m` or `s o`. It must describe all arguments following the angle specification so they are consumed correctly even when the command is omitted.

`default-mode` = {*mode*}

Default: empty

Mode specification for calls without an angle argument. Without this option, a call without an angle argument executes unchanged.

`fallback` = `flatten`|`omit`|`error`

Default: `flatten`

Behavior for genuine overlays without a native backend: `flatten` executes the command, `omit` omits it and its arguments, and `error` reports an error.

```
1 \NewDocumentCommand{\Remark}{m}{\textbf{#1}}
2 \MakeOverlayAwareCommand[arguments=m]{\Remark}
3 % In the document:
4 \Remark<print>{For the printed edition.}
```

The angle specification precedes the original command's star and optional arguments, for example:

```
1 \MakeOverlayAwareCommand
2   [arguments={s o},default-mode=slides,fallback=flatten]{\}
3 Text\<slides>[3ex]
```

6. Filter modes

`osglecture-modes` generalizes Beamer's own `\mode*` into a portable mechanism: `*` is itself just a pre-installed mode (see Section 2), and the package does not distinguish between it and any other, self-declared *filter mode*. While a filter mode is active, everything outside registered *unit environments* (by default `frame`, if present) is dropped, with registered exceptions. This works equally under Beamer, `ltx-talk`, and with ordinary classes such as `article` – an exam with task, solution, and grading environments needs neither Beamer nor `ltx-talk`.

6. Filter modes

The actual filtering needs Lua^AT_EX and runs through the native backend where possible: if a native `\mode*` exists (Beamer) *and* no exceptions beyond its fixed allow-list are registered, `*` delegates there unchanged – no Lua^AT_EX, no behavior change for existing Beamer documents. As soon as custom exceptions are registered, or a filter mode other than `*` is used, the package’s own portable scanner necessarily takes over, since no backend can know about an arbitrary, self-declared allow-list.

```
1 \usepackage[ignorenonframetext]{osglecture-modes}
```

The package option `ignorenonframetext` (default when named bare: true) activates the built-in `* mode` – equivalent to a `\mode*` at the start of the document. No package option is needed for any other filter mode: it is declared like any other mode (`\DeclareLectureMode`) and activated directly in the document via the switch form `\mode<name>` – without a content argument; `\mode*` is only the syntax kept for Beamer compatibility with `\mode<*>`. `raw-skip-envs=<list>` replaces the built-in list of environments to skip verbatim. The defaults are `verbatim`, `Verbatim`, `lstlisting`, `minted`, and `comment`. Include custom verbatim environments in this list if they occur in filtered parts of the document.

Everything outside a registered unit environment is dropped, with built-in exceptions: `frame` and `frame*` themselves (if present), `\againframe` (if present) as a standalone command for repeating a unit environment, structural commands including `\appendix`, `\lecturemode`, `\mode`, and `\maketitle` pass through the scanner unchanged; the structural commands also preserve an immediately following `\label`, while a `\label` elsewhere in dropped text vanishes along with it. The optional, still *ltx-talk*-specific adapter `osglecture-modes-ltxtalk` adds `\lecture` as a further exception and resumes filtering after every `\lecture`.

When portable filtering starts

In the current package, the mode switch enables filtering, but actual skipping starts only after `\maketitle`. With the additional package for *ltx-talk*, it also starts after `\lecture`. Earlier text is processed normally. A document without such a call is not filtered by the switch alone. This applies to the portable filter; delegation to Beamer retains its native behavior.

6.1. Registering exceptions

`\DeclareLectureKeepCommand[<mode=<name>>]{<command>}`

Keeps an already defined command with exactly one mandatory argument, such as a custom input command. The command and its argument are executed normally.

`\DeclareLectureKeepLabelCommand[<mode=<name>>]{<command>}`

As above, but only immediately after a structural command. This lets a custom label command replace the built-in `\label` without preserving labels in discarded text.

6. Filter modes

`\DeclareLectureKeepEnvironment`[$\langle mode = \langle name \rangle \rangle$]{ $\langle list \rangle$ }

Keeps the environments in a comma-separated list. They must already be defined. Their content is processed normally; filtering resumes after each such environment ends.

Without `mode=`, the exception applies in every filter mode. With `mode= $\langle name \rangle$` , it applies only when that mode is active. Inheritance counts too: a selected mode with multiple filter modes as parents combines their exceptions. Register exceptions in the preamble before activating filtering.

```
1 \documentclass{article}
2 \usepackage{osglecture-modes} % compile with LuaLaTeX
3 \NewDocumentEnvironment{task}{}{}{}
4 \DeclareLectureKeepEnvironment{task}
5 \DeclareLectureMode{solutions}
6 \NewDocumentEnvironment{solution}{}{}{}
7 \DeclareLectureKeepEnvironment[mode=solutions]{solution}
8 \title{Exercises}
9 \author{}
10 \date{}
11 \begin{document}
12 \mode<solutions>
13 \maketitle
14 This text is skipped.
15 \begin{task}An exercise.\end{task}
16 \begin{solution}Its solution.\end{solution}
17 \end{document}
```

The example prints the title, exercise, and solution. Replacing `\mode<solutions>` with `\mode*` keeps the title and exercise. Without an active filter, text outside the environments is processed normally too; registering exceptions alone does not hide anything. A switch activates the named mode additionally rather than replacing the active mode set. Multiple activated filter modes therefore combine their exceptions. The package currently provides no portable deactivation command.

6.2. Additional package for ltx-talk

```
1 \documentclass{ltx-talk}
2 \usepackage[ignorenonframetext]{osglecture-modes-ltxtalk}
```

The additional package loads the core package and adds `\lecture` as an exception. It accepts

`mode`, `ignorenonframetext`, and `raw-skip-envs`. If you need other core options, load `osglecture-modes` first with those options. Register custom environments and commands as above.

7. Defining mode-dependent commands

For simple conditional content, `\IfLectureModeTF` suffices. To give a public command a different definition depending on the document mode, use behavior classes instead. These group modes that share a command definition, such as `dynamic` for slides and `longform` for articles and scripts. Behavior classes and document modes have separate namespaces; identical names do not establish an automatic connection. The standalone package does not predeclare behavior classes.

```

1 \documentclass{article}
2 \usepackage[mode=script]{osglecture-modes}
3 \DeclareModeBehaviorClass{dynamic}
4 \DeclareModeBehaviorClass{longform}
5 \AssignLectureModeBehavior{slides}{dynamic}
6 \AssignLectureModeBehavior{script,article}{longform}
7 \NewDocumentCommand{\BriefRemark}{m}{\textbf{#1}}
8 \NewDocumentCommand{\FullRemark}{m}{\begin{quote}#1\end{quote}}
9 \DeclareModeAwareCommand{\Remark}{dynamic, longform}
10 \DeclareModeAwareCommandImplementation
11   {\Remark}{dynamic}{\BriefRemark}
12 \DeclareModeAwareCommandImplementation
13   {\Remark}{longform}{\FullRemark}
14 \begin{document}
15 \Remark{This is set as a quotation in script mode.}
16 \end{document}

```

`\DeclareModeBehaviorClass{<class>}`

Declares a behavior class.

`\AssignLectureModeBehavior{<modes>}{<class>}`

Assigns a previously declared behavior class to a comma-separated list of modes. Aliases are allowed. Assignment is not inherited from parent modes: assign the modes actually selected.

`\DeclareModeAwareCommand{<command>}{<classes>}`

Declares a new public command and the comma-separated list of behavior classes for which definitions are required. An existing command is not overwritten.

`\DeclareModeAwareCommandImplementation{<command>}{<class>}{<implementation command>}`

Registers the implementation command for a behavior class. That command owns the complete argument signature. Use signatures compatible with the shared source for all variants.

8. Limitations

`\FinalizeModeAwareCommands`

Checks all required definitions, including behavior classes inactive in the current run, and binds public commands to their selected definitions. Also finalizes the mode graph if necessary.

Selection considers the modes chosen with `\SetLectureModes` or `\AddLectureModes`. At least one must supply a behavior class; multiple assigned modes must not select conflicting classes. Finalization occurs automatically on first use of such a command, or before `\begin{document}` at the latest. Afterwards, no further declarations or assignments for mode-dependent commands are possible. Activating filter modes later does not reselect command definitions.

8. Limitations

- A registered mode may prefix a native overlay tail (`<slides:2->`), be staggered per mode with `|` (`<slides:2-|handout:0>`), or be `||`-combined with one. A segment carrying an unregistered mode name makes the whole specification native, and a disjunction with a bare `|` or `,` across the boundary between a portable mode and a native atom (`<presentation|2->`) is not available as a portable *or*; use `||` for that.
- Load the base class before `oslecture-modes`. Classes loading their base class later can use `deferred=true`.
- If a base class lacks `\only`, `\uncover`, `\visible`, `\onslide`, `\invisible`, `\alt`, `\pause`, or `\temporal`, `oslecture-modes` provides a portable version of the *command*. The corresponding environments (`onlyenv`, `uncoverenv`, `visibleenv`, `invisibleenv`) remain undefined there; shared source should use the command form in that case.

A. Bundle integration: a custom document type

This section is for users of [osglecture](#) and OLLM. These steps are not required for standalone use of [osglecture-modes](#). We add studyguide, a printable long-form document based on [scrbook](#). The OLLM target specifies the document type, which is selected as a mode. The document profile specifies the base class and document setup instead.

A.1. Registering the target

Create `targets/studyguide.toml` in a configured OLLM definition path:

```
1 schema = 1
2 kind = "target"
3 name = "studyguide"
4 version = "1.0"
5 doctype = "studyguide"
6 family = "publication"
7 unit_scopes = ["a", "as"]
```

Then enable the target in `ollmconfig.toml`. The definition path must be configured through local OLLM settings or an installed bundle preset.

```
1 [targets.studyguide]
2 languages = ["de", "en"]
```

A.2. Declaring mode and behavior

Create `osglecture-mode-studyguide.def` where TeX can find it. The [osglecture](#) class already supplies the longform behavior class.

```
1 \ProvidesExplFile{osglecture-mode-studyguide.def}
2 {2026-09-27}{v0.5.1}{studyguide mode setup}
3 \DeclareLectureMode[parents={longform,print}]{studyguide}
4 \AssignLectureModeBehavior{studyguide}{longform}
```

Queries for studyguide, longform, and print now match. If commands need behavior different from existing long-form documents, declare a separate behavior class and supply the required command definitions as described in [Section 7](#).

A.3. Setting up the document profile

Create `osglecture-profile-studyguide-scrbook.def`:

```
1 \ProvidesExplFile{osglecture-profile-studyguide-scrbook.def}
2 {2026-09-27}{v0.5.1}{studyguide scrbook profile}
3 \DeclareOsgLectureProfile{studyguide-scrbook}
4 {
5   backend = scrbook,
6   class = scrbook,
7   doctypes = {script,article,studyguide},
8   mode-setup-file = osglecture-mode-studyguide.def,
9   setup-file = osglecture-studyguide.code.tex
10 }
11 \DeclareOsgLectureProfileClassOptions
12 {studyguide-scrbook}{studyguide}{oneside}
```

`mode-setup-file` contains mode declarations and assignments. Also provide `osglecture-studyguide.code.tex`, which contains setup and formatting applied after loading the base class. Including `script` and `article` keeps these targets available when replacing the project's long-form profile. Select the profile in the project or user configuration:

```
1 [latex.defaults]
2 script_profile = "studyguide-scrbook"
```

The class loads the mode setup file before selecting the new mode; no change to the class core is needed. Finally, build each enabled language: target, document type, and profile must agree, mode queries must return the expected results, and the mode-dependent commands used must be defined. Also check a document with explicit doctype and profile outside an OLLM project. The [osglecture](#) and OLLM manuals describe the full profile and target configuration.

Index

A	
<code>\ActiveLectureModes</code>	4, 6
<code>\AddLectureModes</code>	3, 6, 11
<code>\againframe</code>	8
<code>\alt</code>	11
<code>\appendix</code>	8
<code>arguments</code>	7
<code>\AssignLectureModeBehavior</code>	10
B	
<code>beamer</code> (package)	1
<code>\begin</code>	3, 11
C	
<code>comment</code> (environment)	5, 8
D	
<code>\DeclareLectureKeepCommand</code>	8
<code>\DeclareLectureKeepEnvironment</code>	9
<code>\DeclareLectureKeepLabelCommand</code>	8
<code>\DeclareLectureMode</code>	5, 8
<code>\DeclareLectureModeAlias</code>	5
<code>\DeclareLectureModeParents</code>	5
<code>\DeclareModeAwareCommand</code>	10
<code>\DeclareModeAwareCommandImplementation</code>	10
<code>\DeclareModeBehaviorClass</code>	10
<code>default-mode</code>	7
<code>deferred</code>	5
F	
<code>fallback</code>	7
<code>\FinalizeLectureModes</code>	3 f., 6
<code>\FinalizeModeAwareCommands</code>	11
<code>frame</code> (environment)	7 f.
<code>frame*</code> (environment)	8
<code>full</code>	5
I	
<code>\IfLectureModeF</code>	3
<code>\IfLectureModeT</code>	3
<code>\IfLectureModeTF</code>	3, 10
<code>ignorenonframetext</code>	5, 10
<code>\invisible</code>	11
<code>invisibleenv</code> (environment)	11
L	
<code>\label</code>	8
<code>\lecture</code>	8 f.
<code>\lecturemode</code>	3, 8
<code>\LectureModes</code>	6
<code>\LectureModesActivateOverlays</code>	6
<code>lstlisting</code> (environment)	5, 8
<code>ltx-talk</code> (package)	1
M	
<code>\MakeOverlayAwareCommand</code>	7
<code>\maketitle</code>	8
<code>minted</code> (environment)	5, 8
<code>\mode</code>	6, 8
<code>mode</code>	5, 10
<code>\mode*</code>	7 ff.
<code>\mode<*></code>	8
<code>\mode<name></code>	8
<code>\ModeValue</code>	4, 6
N	
<code>\NewDocumentCommand</code>	7
O	
<code>\only</code>	11
<code>onlyenv</code> (environment)	11
<code>\onslide</code>	11
P	
<code>patch-overlays</code>	5
<code>\pause</code>	11
R	
<code>raw-skip-envs</code>	5, 10
S	
<code>\SetLectureModes</code>	3, 5, 11
T	
<code>\temporal</code>	11
U	
<code>\uncover</code>	11
<code>uncoverenv</code> (environment)	11
V	
<code>Verbatim</code> (environment)	5, 8
<code>verbatim</code> (environment)	5, 8
<code>\visible</code>	11
<code>visibleenv</code> (environment)	11