

PORTABLE DOKUMENTMODI

VO.5.1 2026-09-27

Erweiterbare Modusgraphen für L^AT_EX-Dokumente

Matthias WERNER¹

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

L^AT_EX-Pakete wie [beamer](#)² und [ltx-talk](#)³ bieten das Konzept von *Modi* an, bei dem, abhängig vom Dokumentmodus, nur bestimmte Dokumentteile gezielt übersetzt werden.

Dieses Paket erweitert das Konzept und verwaltet Dokumentmodi als gerichteten azyklischen Graphen. Es ergänzt die Modusschnittstellen von [beamer](#) und [ltx-talk](#), funktioniert aber ebenso mit gewöhnlichen Dokumentklassen. Es gehört zum [osglecture](#)-Bundle, ist jedoch auch unabhängig davon nutzbar.

Inhaltsverzeichnis

1. Schnellstart	2	6. Filtermodi	7
		6.1. Ausnahmen registrieren	8
2. Modi und Vererbung	2	6.2. Zusatzpaket für ltx-talk	9
2.1. Eigene Modi und mehrere aktive Modi	3	7. Modusabhängige Befehle definieren	10
2.2. Zeitpunkt der Deklarationen .	3		
3. Inhalte und Werte auswählen	3	8. Einschränkungen	11
3.1. Modusabhängige Werte	4		
4. Konfigurationsreferenz	5	A. Bundle-Integration: ein eigener Dokumenttyp	12
4.1. Paketoptionen	5	A.1. Target registrieren	12
4.2. Deklaration und Auswahl . . .	5	A.2. Modus und Verhalten festlegen	12
5. Zusammenspiel mit Backends	6	A.3. Dokumentprofil einrichten . .	13
5.1. Eigene Befehle overlayfähig machen	7	Index	14

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

1. Schnellstart

Wer Beamer oder [ltx-talk](#) verwendet, kennt Modusspezifikationen wie `<presentation>` oder `<handout>`. [oslecture-modes](#) macht solche Dokumentmodi auch in anderen Klassen nutzbar und erlaubt eigene Modi mit mehreren Elternmodi. Ein gemeinsamer Quelltext kann so beispielsweise Folien, ein Handout und ein Skript enthalten.

```

1 \documentclass{article}
2 \usepackage[mode=script]{oslecture-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}

```

Das Beispiel gibt den gemeinsamen Text, den Text für Druckfassungen und die ausführliche Erklärung aus. Der Präsentationstext entfällt, weil `script` den Modus `print`, aber nicht `presentation` aktiviert. `mode=handout` würde stattdessen beide Textvarianten und die kurze Erklärung auswählen. Die Paketoption ändert dabei nicht die Dokumentklasse.

Das Kernpaket benötigt einen aktuellen L^AT_EX-Kernel. Es kann unabhängig von [oslecture](#), OLLM und Projektverzeichnissen verwendet werden. Laden Sie es gewöhnlich nach der Dokumentklasse. Ohne explizite Moduswahl erkennt es Beamer und [ltx-talk](#); bei anderen Klassen wählt es `article`. Nur die portablen Textfilter aus Abschnitt 6 benötigen LuaL^AT_EX.

2. Modi und Vererbung

Ein ausgewählter Modus aktiviert auch alle seine Eltern, deren Eltern usw. Mehrere Eltern sind erlaubt; Kreise sind es nicht. Anders als bei einer reinen Baumstruktur kann deshalb `handout` sowohl zu `presentation` als auch zu `print` gehören. Die folgenden Modi sind bereits deklariert; die Tabelle nennt jeweils die unmittelbaren Eltern.

Modus	Eltern
<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>	—

3. Inhalte und Werte auswählen

`slides` ist ein Alias für `projector`, kein zusätzlicher Modus. `all` ist eine stets wahre Abfrage. Der Modus `*` dient der Textfilterung (Abschnitt 6).

2.1. Eigene Modi und mehrere aktive Modi

```
1 \documentclass{article}
2 \usepackage{oslecture-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` ersetzt die bisherige Auswahl; `\AddLectureModes` ergänzt sie. Beide nehmen kommaseparierte Listen entgegen. Die Auswahl `slides`, `web` aktiviert beispielsweise gleichzeitig `projector`, `presentation`, `web` und `longform`. Die Dokumentklasse und ihre nativen Ausgabeoptionen bleiben davon unberührt.

2.2. Zeitpunkt der Deklarationen

Deklarieren Sie Modi, Eltern und Aliase und wählen Sie danach die aktiven Modi, bevor die erste Abfrage erfolgt. `\FinalizeLectureModes` prüft den gesamten Graphen, auch inaktive Modi, und schließt diese Konfiguration ab. Die erste Abfrage mit `\IfLectureModeTF`, `\IfLectureModeT`, `\IfLectureModeF` oder einer spezifizierten `\lecturemode`-Anweisung erledigt dies bei Bedarf automatisch; spätestens vor `\begin{document}` wird abgeschlossen. Danach sind Deklarationen, `\SetLectureModes` und `\AddLectureModes` nicht mehr zulässig. Die Aktivierung bereits deklarierter Filtermodi im Dokument ist ein eigener Vorgang, siehe Abschnitt 6.

3. Inhalte und Werte auswählen

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

Gibt den Inhalt aus, wenn der Ausdruck wahr ist. Ohne Spezifikation wird der Inhalt immer ausgegeben.

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

Führt je nach Ergebnis den zweiten oder dritten Parameter aus.

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

Führt den zweiten Parameter nur bei wahrer Abfrage aus.

3. Inhalte und Werte auswählen

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

Führt den zweiten Parameter nur bei falscher Abfrage aus.

Diese Abfragen akzeptieren einen Modusnamen oder mehrere Namen, getrennt durch `|` oder Kommas (Oder-Verknüpfung). Aliase sind erlaubt; unbekannte Namen ergeben einen Fehler. `all` trifft immer zu. Ein einzelnes `|`, Overlaynummern, Negation und Und-Verknüpfung gehören nicht zu dieser Abfragesyntax. Für verschachtelte Bedingungen können Sie mehrere Abfragen ineinandersetzen. Bei nativen Overlaybefehlen gelten andere Regeln für Kommas und `|`, siehe Abschnitt 5.

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

3.1. Modusabhängige Werte

`\ModeValue{<specification>}`

Expandiert zu einem ausgewählten Wert. Die Spezifikation besteht aus mit `|` getrennten Einträgen der Form `modus=wert`. `default=wert` oder ein Eintrag ohne `=` gibt den Ersatzwert an. Ohne Treffer und ohne Ersatzwert ist das Ergebnis leer.

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

Die Reihenfolge der Einträge in der Spezifikation entscheidet nicht über die Auswahl. Maßgeblich ist `\ActiveLectureModes`: Eltern stehen vor ihren Kindern, die ausgewählten Modi werden in Auswahlreihenfolge durchlaufen; jeder Modus kommt nur einmal vor. Der erste Modus mit einem Eintrag gewinnt, `default` kommt zuletzt. Verwenden Sie hier die kanonischen Namen, also beispielsweise `projector` statt des 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` ist expandierbar und auch für Längen- und Registerzuweisungen geeignet. Es finalisiert den Graphen nicht selbst: Soll es bereits in der Präambel verwendet werden, rufen Sie zuvor `\FinalizeLectureModes` auf. Werte dürfen weitere Gleichheitszeichen und Doppelpunkte enthalten; `|` dient als Trenner. Verwenden Sie höchstens einen Ersatzwert.

4. Konfigurationsreferenz

4.1. Paketoptionen

`mode = {⟨name⟩}` Voreinstellung: auto
Wählt beim Laden einen bereits bekannten Modus. Eigene Modi erst nach dem Laden deklarieren und dann mit `\SetLectureModes` wählen. Bei auto wählt Beamer im Handoutbetrieb `handout, beamer`, sonst `slides, beamer`; `ltx-talk` liefert entsprechend `handout, article` oder `slides`. Andere Klassen erhalten `article`.

`full = true|false` Voreinstellung: true
Aktiviert die Integration in vorhandene Modus- und Overlaybefehle. `full=false` lässt diese nativen Befehle unverändert. Portable Abfragen bleiben nutzbar. Fehlende Befehle können weiterhin ergänzt werden.

`patch-overlays = true|false` Voreinstellung: true
Dieselbe Einstellung wie `full`; bei mehrfacher Angabe gilt die zuletzt gesetzte Einstellung.

`ignorenonframetext = true|false` Voreinstellung: false
Aktiviert zu Dokumentbeginn die Filterung mit *. Bloße Angabe der Option bedeutet `true`.

`raw-skip-envs = {⟨list⟩}`
Ersetzt die Liste der beim Filtern wörtlich zu überspringenden Umgebungen. Vorgabe: `verbatim, Verbatim, lstlisting, minted, comment`. Siehe Abschnitt 6.

`deferred = true|false` Voreinstellung: false
Für Klassenautoren: unterdrückt die automatische Moduswahl und Overlayintegration beim Laden. Die Klasse wählt den Modus selbst und ruft nach dem Laden ihrer Basisklasse den unten beschriebenen Aktivierungsbefehl auf.

4.2. Deklaration und Auswahl

`\DeclareLectureMode[⟨parents=⟨list⟩⟩]{⟨mode⟩}`
Deklariert einen Modus, optional mit kommaseparierten Eltern. `parent` ist ein Synonym für `parents`. Eltern müssen spätestens beim Abschluss deklariert sein; verwenden Sie ihre kanonischen Namen.

`\DeclareLectureModeParents{⟨mode⟩}{⟨list⟩}`
Setzt die vollständige Elternliste neu; ergänzt nicht die alte Liste. Deklariert den Modus bei Bedarf mit.

`\DeclareLectureModeAlias{⟨alias⟩}{⟨mode⟩}`
Deklariert einen alternativen Namen. Aliasketten sind erlaubt, Aliaskreise und unbekannte Endziele nicht.

`\SetLectureModes{⟨list⟩}`
Ersetzt die bisherige Auswahl durch die angegebene Liste.

`\AddLectureModes{<list>}`

Ergänzt die Auswahl. Die Modi müssen bereits deklariert sein; Aliase werden aufgelöst, doppelte Einträge entfernt.

`\FinalizeLectureModes`

Prüft und schließt den Graphen ab. Wiederholter Aufruf ist erlaubt.

`\LectureModes`

Expandiert zur kommaseparierten Auswahl ohne geerbte Eltern und mit aufgelösten Aliasnamen.

`\ActiveLectureModes`

Expandiert zur aktiven Liste einschließlich Eltern. Vor Abschluss ist diese Liste noch nicht berechnet. Beide Listenbefehle finalisieren nicht selbst. Die Reihenfolge bestimmt auch `\ModeValue`.

`\LectureModesActivateOverlays`

Bindet nach dem Laden der Basisklasse die Overlaybefehle ein. Nur bei `deferred=true` selbst einmal aufrufen. Wählt keinen Modus.

5. Zusammenspiel mit Backends

Besitzt die geladene Klasse keinen Befehl `\mode`, stellt das Paket dessen portable Inlineform bereit. Existiert bereits eine native Implementierung, behandelt das Paket ausschließlich Inlineausdrücke aus registrierten Modi. Ebenso sortieren die patchbaren Overlaybefehle und -Decoder von Beamer und `ltx-talk` eine Winkelspezifikation in folgende Arten ein:

- *portabel* – eine mit `||` verknüpfte Oder-Liste registrierter Modi (oder `all`). Sie wird vollständig vom Paket ausgewertet und funktioniert auch ohne Overlaybackend.
- *modusqualifiziert* – jedes mit `|` getrennte Segment ist ein registrierter Modus, eine Form `modus:rest` oder ein rein numerisches Overlayatom. Die portablen Anteile wertet das Paket aus; die aktiven nativen Reste werden zu einer einzigen Backendspezifikation zusammengesetzt. So wird `<slides:2-|handout:0>` portabel entschieden und dem Backend nur der zutreffende Overlaybereich gereicht.
- *nativ* – alles Übrige: Backend-Modusnamen, `|`-Modusskopierung über unbekannte Modi, Actionspezifikationen und native Modusschalter mit unbekannten Namen. Wird unverändert an das Backend delegiert.

Bei Backendbefehlen sind `|` und `,` innerhalb von `<...>` stets Backendsyntax; `||` ist der einzige portable Oder-Operator. Ein Ausdruck, der portable Modi mit nativen Overlayatomen mischt, muss dies also über `||` oder die Form `modus:rest` kenntlich machen. Fehlt der geladenen Klasse ein Overlaybackend, wird eine native oder unerkannte Spezifikation zu *anzeigen* abgeschwächt. Alle diese Eingriffe lassen sich mit `patch-overlays=false` abschalten.

5.1. Eigene Befehle overlayfähig machen

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

Erweitert einen vorhandenen Befehl um eine unmittelbar folgende Winkelspezifikation. Registrierte Modi werden portabel ausgewertet; echte Overlays werden vom vorhandenen Backend ausgewertet.

`arguments` = {*signature*}

Voreinstellung: leer

Argumentsignatur des Originalbefehls in der Syntax von `\NewDocumentCommand`, beispielsweise `m` oder `s o`. Sie muss alle nach der Winkelspezifikation folgenden Argumente beschreiben, damit diese auch im ausgelassenen Zweig korrekt gelesen werden.

`default-mode` = {*mode*}

Voreinstellung: leer

Modusspezifikation für Aufrufe ohne Winkelargument. Ohne diese Option wird ein Aufruf ohne Winkelargument unverändert ausgeführt.

`fallback` = `flatten|omit|error`

Voreinstellung: `flatten`

Verhalten bei echten Overlays ohne natives Backend: `flatten` führt den Befehl aus, `omit` lässt ihn einschließlich Argumenten aus, `error` meldet einen Fehler.

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

Die Winkelspezifikation steht vor Stern und optionalen Argumenten des Originalbefehls, beispielsweise:

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

6. Filtermodi

`oslecture-modes` verallgemeinert Beamers eigenes `\mode*` zu einem portablen Mechanismus: `*` ist selbst nur ein vorinstallierter Modus (siehe Abschnitt 2), und das Paket unterscheidet nicht zwischen ihm und jedem anderen, selbst deklarierten *Filtermodus*. Ist ein Filtermodus aktiv, wird alles außerhalb registrierter *Inhaltsumgebungen* (per Vorgabe `frame`, falls vorhanden) verworfen, mit registrierten Ausnahmen. Das funktioniert unter Beamer, `ltx-talk` und mit

gewöhnlichen Klassen wie `article` gleichermaßen – eine Prüfung mit Aufgaben-, Lösungs- und Bewertungsumgebungen benötigt weder Beamer noch `ltx-talk`.

Die tatsächliche Filterung benötigt Lua^AT_EX und läuft, wo möglich, über das eigene Backend: Ist ein natives `\mode*` vorhanden (Beamer) und sind keine Ausnahmen über dessen feste Positivliste hinaus registriert, delegiert * unverändert dorthin – ohne Lua^AT_EX und ohne Verhaltensänderung an bestehenden Beamer-Dokumenten. Sobald eigene Ausnahmen registriert sind oder ein anderer Filtermodus als * verwendet wird, übernimmt zwingend der portable Scanner dieses Pakets, da kein Backend eine beliebige, selbst deklarierte Positivliste kennen kann.

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

Die Paketoption `ignorenonframetext` (Vorgabe bei bloßer Nennung: wahr) aktiviert den eingebauten Modus * – entspricht also einem `\mode*` zu Dokumentbeginn. Für jeden anderen Filtermodus ist keine eigene Paketoption nötig: Er wird wie jeder andere Modus deklariert (`\DeclareLectureMode`) und über die Schalterform `\mode<name>` – ohne Inhaltsargument – direkt im Dokument aktiviert; `\mode*` ist dafür nur die aus Beamer-Kompatibilität beibehaltene Syntax für `\mode<*>`. `raw-skip-envs=<Liste>` ersetzt die Liste wörtlich zu überspringender Umgebungen. Vorgabe sind `verbatim`, `Verbatim`, `lstlisting`, `minted` und `comment`. Ergänzen Sie eigene `Verbatim`-Umgebungen in dieser Liste, wenn sie in ausgefilterten Dokumentteilen vorkommen.

Alles außerhalb einer registrierten Inhaltsumgebung wird verworfen, mit eingebauten Ausnahmen: `frame` und `frame*` selbst (falls vorhanden), `\againframe` (falls vorhanden) als eigenständiger Befehl zum Wiederholen einer Inhaltsumgebung, die Gliederungsbefehle einschließlich `\appendix`, `\lecturemode`, `\mode` und `\maketitle` passieren den Scanner unverändert; die Gliederungsbefehle erhalten außerdem ein unmittelbar folgendes `\label`; ein `\label` an anderer Stelle verschwindet dagegen mit dem verworfenen Text. Der optionale, weiterhin `ltx-talk`-spezifische Adapter `osglecture-modes-ltxtalk` ergänzt `\lecture` als weitere allgemeine Ausnahme und setzt die Filterung nach jedem `\lecture` fort.

Beginn der portablen Filterung

Im aktuellen Paket aktiviert der Modusschalter die Filterung, das tatsächliche Überspringen beginnt aber erst nach `\maketitle`. Beim Zusatzpaket für `ltx-talk` startet es auch nach `\lecture`. Text davor wird normal verarbeitet. Ein Dokument ohne einen solchen Aufruf wird durch den Schalter allein noch nicht gefiltert. Dies betrifft den portablen Filter; bei Delegation an Beamer gilt dessen natives Verhalten.

6.1. Ausnahmen registrieren

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

6. Filtermodi

Erhält einen bereits definierten Befehl mit genau einem Pflichtargument, etwa einen eigenen Eingabebefehl. Befehl und Argument werden normal ausgeführt.

`\DeclareLectureKeepLabelCommand[ $\langle mode = \langle name \rangle \rangle$ ]{ $\langle command \rangle$ }`

Wie oben, aber nur unmittelbar nach einem Gliederungsbefehl. Damit kann ein eigener Label-Befehl das eingebaute `\label` ersetzen, ohne Labels in verworfenem Text zu erhalten.

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

Erhält die Umgebungen einer kommaseparierten Liste. Sie müssen bereits definiert sein. Ihr Inhalt wird normal verarbeitet; nach dem Ende einer solchen Umgebung wird wieder gefiltert.

Ohne `mode=` gilt die Ausnahme in jedem Filtermodus. Mit `mode= $\langle name \rangle$` gilt sie nur, wenn dieser Modus aktiv ist. Dabei zählt auch die Vererbung: Ein ausgewählter Modus mit mehreren Filtermodi als Eltern vereinigt deren Ausnahmen. Registrieren Sie die Ausnahmen in der Präambel, bevor Sie die Filterung aktivieren.

```
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}
```

Das Beispiel gibt den Titel, die Aufgabe und die Lösung aus. Ersetzen Sie `\mode<solutions>` durch `\mode*`, bleiben Titel und Aufgabe. Ohne Aktivierung eines Filters wird auch der Text außerhalb der Umgebungen normal verarbeitet; die Ausnahmeregistrierung allein blendet nichts aus. Ein Schalter aktiviert den genannten Modus zusätzlich, er ersetzt nicht die bisher aktive Modusmenge. Mehrere aktivierte Filtermodi vereinigen daher ihre Ausnahmen. Eine portable Anweisung zum Deaktivieren bietet das Paket derzeit nicht.

6.2. Zusatzpaket für ltx-talk

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

Das Zusatzpaket lädt das Kernpaket und ergänzt `\lecture` als Ausnahme. Es akzeptiert `mode`, `ignorenonframetext` und `raw-skip-envs`. Weitere Kernoptionen geben Sie bei Bedarf beim vorherigen Laden von `osglecture-modes` an. Für eigene Umgebungen und Befehle gelten dieselben Registrierungsbeefhle wie oben.

7. Modusabhängige Befehle definieren

Für einfache Fallunterscheidungen genügt `\IfLectureModeTF`. Soll ein öffentlicher Befehl je nach Dokumentmodus eine eigene Definition erhalten, können Sie dagegen Verhaltensklassen verwenden. Diese bündeln Modi mit derselben Befehlsdefinition, etwa `dynamic` für Folien und `longform` für Artikel und Skripte. Verhaltensklassen und Dokumentmodi sind getrennte Namenräume; gleiche Namen stellen keine automatische Verbindung her. Das eigenständige Paket deklariert keine Verhaltensklassen.

```
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>}`

Deklariert eine Verhaltensklasse.

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

Ordnet einer kommaseparierten Modusliste eine bereits deklarierte Verhaltensklasse zu. Aliase sind erlaubt. Die Zuordnung wird nicht von Elternmodi geerbt: Ordnen Sie die tatsächlich ausgewählten Modi zu.

`\DeclareModeAwareCommand{⟨command⟩}{⟨classes⟩}`

Deklariert einen neuen öffentlichen Befehl und die kommaseparierte Liste der Verhaltensklassen, für die Definitionen vorliegen müssen. Ein vorhandener Befehl wird nicht überschrieben.

`\DeclareModeAwareCommandImplementation{⟨command⟩}{⟨class⟩}{⟨implementation command⟩}`

Registriert den Definitionsbefehl für eine Verhaltensklasse. Dieser besitzt selbst die vollständige Argumentsignatur. Verwenden Sie für alle Varianten eine zum gemeinsamen Quelltext passende Signatur.

`\FinalizeModeAwareCommands`

Prüft alle geforderten Definitionen, auch für im aktuellen Lauf inaktive Verhaltensklassen, und bindet die öffentlichen Befehle an ihre gewählten Definitionen. Schließt bei Bedarf auch den Modusgraphen ab.

Die Auswahl berücksichtigt die mit `\SetLectureModes` oder `\AddLectureModes` gewählten Modi. Mindestens einer muss eine Verhaltensklasse liefern; mehrere zugeordnete Modi dürfen keine widersprüchlichen Klassen auswählen. Beim ersten Gebrauch eines solchen Befehls, spätestens vor `\begin{document}`, erfolgt der Abschluss automatisch. Danach sind keine weiteren Deklarationen oder Zuordnungen für modusabhängige Befehle möglich. Später aktivierte Filtermodi wählen die Befehlsdefinitionen nicht erneut aus.

8. Einschränkungen

- Ein registrierter Modus darf einem nativen Overlayrest vorangestellt (`<slides:2->`), mit `|` modusweise gestaffelt (`<slides:2-|handout:0>`) oder per `||` oder-verknüpft werden. Ein Segment mit einem nicht registrierten Modusnamen macht jedoch die gesamte Spezifikation nativ, und eine Disjunktion mit nacktem `|` oder `,` über die Grenze zwischen portablem Modus und nativem Atom hinweg (`<presentation|2->`) ist nicht als portable Oder-Verknüpfung verfügbar; dafür ist `||` zu verwenden.
- Laden Sie die Basisklasse vor `oslecture-modes`. Für Klassen, die erst später ihre Basis-klassse laden, gibt es die Option `deferred=true`.
- Fehlt einer Basisklasse `\only`, `\uncover`, `\visible`, `\onslide`, `\invisible`, `\alt`, `\pause` oder `\temporal`, stellt `oslecture-modes` eine portable Fassung des *Befehls* bereit. Die entsprechenden Environments (`onlyenv`, `uncoverenv`, `visibleenv`, `invisibleenv`) bleiben dort undefiniert; gemeinsamer Quelltext sollte in diesem Fall die Befehlsform verwenden.

A. Bundle-Integration: ein eigener Dokumenttyp

Dieser Abschnitt richtet sich an Nutzer von `osglecture` und OLLM. Für die eigenständige Verwendung von `osglecture-modes` sind diese Schritte nicht nötig. Als Beispiel ergänzen wir `studyguide`, eine druckbare Langform auf Basis von `scrbook`. Das OLLM-Target bestimmt den Dokumenttyp; dieser wird als Modus ausgewählt. Das Dokumentprofil bestimmt dagegen Basisklasse und Einrichtung des Dokuments.

A.1. Target registrieren

Legen Sie in einem eingebundenen OLLM-Definitionspfad die Datei `targets/studyguide.toml` an:

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

Geben Sie das Target anschließend in `ollmconfig.toml` frei. Der Definitionspfad muss über die lokale OLLM-Konfiguration oder ein installiertes Bundle-Preset eingebunden sein.

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

A.2. Modus und Verhalten festlegen

Erstellen Sie die für TeX auffindbare Datei `osglecture-mode-studyguide.def`. Die Klasse `osglecture` stellt die Verhaltensklasse `longform` bereits bereit.

```
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}
```

Damit sind Abfragen auf `studyguide`, `longform` und `print` wahr. Soll ein Befehl anders arbeiten als in bisherigen Langformen, deklarieren Sie eine eigene Verhaltensklasse und ergänzen Sie die erforderlichen Befehlsdefinitionen nach Abschnitt 7.

A.3. Dokumentprofil einrichten

Legen Sie `osglecture-profile-studyguide-scrbook.def` an:

```
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` enthält die Modusdeklarationen und Zuordnungen. Die ebenfalls bereitzustellende Datei `osglecture-studyguide.code.tex` enthält die Einrichtung und Formatierung nach dem Laden der Basisklasse. Das Profil umfasst auch `script` und `article`, damit diese Targets beim Ersetzen des projektweiten Langformprofils verfügbar bleiben. Wählen Sie das Profil in der Projekt- oder Nutzerkonfiguration:

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

Die Klasse lädt die Modusdatei vor Auswahl des neuen Modus. Eine Änderung am Klassenkern ist dafür nicht nötig. Prüfen Sie abschließend den Build für jede freigegebene Sprache: Target, Dokumenttyp und Profil müssen zusammenpassen, die Modusabfragen das erwartete Ergebnis liefern und die verwendeten modusabhängigen Befehle definiert sein. Prüfen Sie auch ein Dokument mit explizitem `doctype` und `profile` ohne OLLM-Projekt. Die vollständige Profil- und Targetkonfiguration beschreiben die Dokumentationen von [osglecture](#) und OLLM.

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> (Paket)	1
<code>\begin</code>	3, 11
C	
<code>comment</code> (Umgebung)	5, 8
D	
<code>\DeclareLectureKeepCommand</code>	8
<code>\DeclareLectureKeepEnvironment</code>	9
<code>\DeclareLectureKeepLabelCommand</code>	9
<code>\DeclareLectureMode</code>	5, 8
<code>\DeclareLectureModeAlias</code>	5
<code>\DeclareLectureModeParents</code>	5
<code>\DeclareModeAwareCommand</code>	11
<code>\DeclareModeAwareCommandImplementation</code>	11
<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> (Umgebung)	7 f.
<code>frame*</code> (Umgebung)	8
<code>full</code>	5
I	
<code>\IfLectureModeF</code>	3 f.
<code>\IfLectureModeT</code>	3
<code>\IfLectureModeTF</code>	3, 10
<code>ignorenonframetext</code>	5, 10
<code>\invisible</code>	11
<code>invisibleenv</code> (Umgebung)	11
L	
<code>\label</code>	8 f.
<code>\lecture</code>	8, 10
<code>\lecturemode</code>	3, 8
<code>\LectureModes</code>	6
<code>\LectureModesActivateOverlays</code>	6
<code>lstlisting</code> (Umgebung)	5, 8
<code>ltx-talk</code> (Paket)	1
M	
<code>\MakeOverlayAwareCommand</code>	7
<code>\maketitle</code>	8
<code>minted</code> (Umgebung)	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> (Umgebung)	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> (Umgebung)	11
V	
<code>Verbatim</code> (Umgebung)	5, 8
<code>verbatim</code> (Umgebung)	5, 8
<code>\visible</code>	11
<code>visibleenv</code> (Umgebung)	11