

DAS LANGSELECT-PAKET

V1.1.3 2026-09-18

Generierung verschiedener Sprachversionen aus einer Quelle

Matthias WERNER¹

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

Dieses Paket unterstützt die Generierung von Dokumenten in zwei oder drei Sprachvarianten aus einem gemeinsamen Quelldokument. Es wird ein entsprechendes Nuttermakro generiert und es gibt mehrere Methoden, die Zielsprache zu bestimmen.

Das Paket gehört zum `osglecture`-Bundle, wird in die `osglecture`-Klasse integriert und vom Buildscript `ollm` unterstützt, kann aber auch eigenständig genutzt werden.

Inhaltsverzeichnis

1	Einleitung	2	3.3	Individuelle Sprachmakros . . .	8
1.1	TL;DR	2			
1.2	Motivation	2	4	Interaktion mit anderen Paketen	9
1.3	Alternativen	2	4.1	Babel/Polyglossia	9
1.4	Sprachen	3	4.2	Csquotes	9
2	Anwendung	4	5	FAQ	10
2.1	Auswahl der Zielsprache . . .	4	6	Implementation	11
2.2	Optionen	5			
3	Nuttermakros	6	6	Index	30
3.1	Automatisch Sprachmakros . .	6			
3.2	Sternvariante	7	7	Index	31

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

1 Einleitung

1.1 TL;DR

Angenommen Sie wollen von einem Text eine englische und eine deutsche Version erhalten, wobei Sie ein gemeinsames Quelldokument nutzen wollen..

1. Binden Sie `\usepackage[languages={en,de}]{langselect}` ein
2. Nutzen Sie im Quelldokument `\lende{<Texten>}{<Textde>}`. *<Text_{en}>* wird in der englischen Version erscheinen, während *<Text_{de}>* in der deutschen Version erscheinen wird.
 - Text, der in beiden Versionen vorkommt, schreiben Sie außerhalb des Makros oder nutzen die Sternversion, um gemeinsame Daten im Fließtext nur einmal schreiben zu müssen, siehe Abschnitt 3.2.

3. Übersetzen Sie das Dokument mit

```
> latex "\def\olsTargetLanguage{en}\include{<document>}"
```

, um die englische Version zu erhalten, und mit

```
> latex "\def\olsTargetLanguage{de}\include{<document>}"
```

für die deutsche.

- Es gibt noch einige andere Möglichkeiten, die Zielsprache zu bestimmen, u. a. über den Jobnamen und über `\DocumentMetadata`.

1.2 Motivation

Mitunter werden mehrere Sprachversionen des gleichen Dokuments benötigt. Im Anwendungsgebiet des `osglecture`-Bündel sind dies beispielsweise die Lehrskripte, Vorlesungsfolien und Handouts einer Lehrveranstaltung. Es fällt häufig schwer, mehrere parallel existierende Sprachversionen konsistent zu halten. Diese Aufgabe ist etwas einfacher, wenn alle Versionen in einem gemeinsamen Quelldokument enthalten sind.

Dieses Paket unterstützt die Generierung solcher Sprachversionen aus einem gemeinsamen L^AT_EX-Quelldokument indem es Makros für Sprachvarianten und Mechanismen zur Auswahl der Zielsprache bereitstellt.

1.3 Alternativen

Es gibt eine Reihe von Alternativen zu `langselect`.

Adhoc-Makros Man kann relativ einfach adhoc Lösungen schaffen, wie z. B.

```
1 \newif\ifenglish
2 \englishtrue
3 \ifenglish
4     Welcome!
```

```

5      \else
6          ¡Bienvenido!
7      \fi
8

```

Tatsächlich folgt **langselect** grundsätzlich diesem Ansatz, automatisiert aber die Erstellung der Sprachmakros, vereinfacht die gemeinsame Nutzung von nichtsprachabhängigen Elementen wie z. B. Formeln und ermöglicht die Auswahl der Sprachvariante über verschiedene Methoden.

multilang Das ausgeklügelte Paket **multilang**¹ von Richard GREWE erstellt Sprachversionen von gegebenen Makros, wobei die aktive Sprache direkt aus **Babel** oder **Polyglossia** übernommen wird. **langselect** hat ein ähnliches Ziel, aber ein anderes (aus Sicht des Paketautoren: praktischeres) Interface. Falls Sie aber den Ansatz von **langselect** nicht mögen, sollten Sie sich unbedingt **multilang** anschauen.

comment Victor EIJKHOUT hat das Paket **comment**² geschrieben, das auf einfache Weise ermöglicht, nur bestimmte Abschnitte im Dokument auszugeben. Dies kann auch sehr gut für eine Sprachauswahl genutzt werden.

bicaption/translations/translator/xtcpts Die Pakete **bicaption**³/**translations**⁴/**translator**⁵/**xtcpts**⁶ adressieren Programmierer von L^AT_EX-Paketen, um fixe Textelemente wie die Überschriften von Verzeichnissen zu internationalisieren. Für Dokumentautoren sind diese Pakete nur bedingt geeignet.

1.4 Sprachen

Das Konzept von **langselect** beinhaltet die Nutzung von Sprachen auf verschiedenen Ebenen mit unterschiedlicher Wirkung. Um Verwechslungen zu vermeiden, wollen wir hier die Begriffe eindeutig definieren.

1. Die Menge *möglicher* Zielsprachen des Dokuments. Ein Dokument, das **langselect** nutzt, enthält typischerweise Textteile in verschiedenen Zielsprachen. Die möglichen Zielsprachen werden über eine Paketoption gesetzt, siehe Abschnitt 2.2. Im weiteren Text dieser Dokumentation nennen wir die *möglichen* Zielsprachen *Auswahlsprachen*, da für die L^AT_EX-Übersetzung eine dieser Sprachen ausgewählt wird.

1. auf CTAN als multilang: <https://ctan.org/tex-archive/macros/latex/contrib/multilang>

2. auf CTAN als comment: <https://ctan.org/tex-archive/macros/latex/contrib/comment>

3. auf CTAN als bicaption: <https://ctan.org/tex-archive/macros/latex/contrib/bicaption>

4. auf CTAN als translations: <https://ctan.org/tex-archive/macros/latex/contrib/translations>

5. auf CTAN als translator: <https://ctan.org/tex-archive/macros/latex/contrib/translator>

6. auf CTAN als xtcpts: <https://ctan.org/tex-archive/macros/latex/contrib/xtcpts>

2. Die tatsächliche bei einer L^AT_EX-Übersetzung gebrauchte Zielsprache. Diese muss eine der Sprachen aus 1. sein. Typischerweise wird der Übersetzungsvorgang für jede der tatsächlichen Zielsprachen einzeln ausgeführt. Für die Auswahl der tatsächlichen Zielsprache gibt es mehrere Methoden, siehe Abschnitt 2.1.
3. Die Sprache, in der Textteile oder einzelne Wörter tatsächlich geschrieben sind und die beispielsweise bei der Worttrennung berücksichtigt wird. Dies ist das Konzept von Sprache, das Pakete wie Babel oder Polyglossia haben.⁷ Bitte beachten Sie, dass ein mehrsprachiges Dokument aus der Sicht von `langselect` ein monosprachiges Dokument aus der Sicht von Babel sein kann. Häufig ist es aber eher umgekehrt, so dann durchaus noch weitere Sprachen als die Auswahlssprachen hinzukommen: Beispielsweise kann ein Dokument mit den Auswahlssprachen Deutsch und Englisch im Text lateinische Wörter oder Sätze enthalten. Diese Auswahl wird über die üblichen Mechanismen wie in `\selectlanguage`, `\foreignlanguage` oder `\text{<sprache>}` vorgenommen. Wenn Babel oder Polyglossia geladen wird, was auf Wunsch auch direkt über `langselect` erfolgen kann, wird als aktive Sprache die Zielsprache ausgewählt.

Sprachen werden in `langselect` über ISO 639-1-Codes (z. B. „en“, „fr“, „de“, ...) bezeichnet. Wenn für die Sprachbehandlung auf der Ebene von Babel/Polyglossia ein bestimmter Dialekt oder eine bestimmte Variante gewünscht ist, kann ein Mapping erfolgen.

2 Anwendung

Das Paket wird auf die übliche Weise geladen:

```
\usepackage[<options>]{langselect}
```

Da die Auswahl der Zielsprache sowohl über Optionen als auch auf anderem Weg erfolgen kann, werden hier zunächst diese verschiedenen Wege beschrieben. Weitere Optionen wie das Setzen der Auswahlssprachen werden in Abschnitt 2.2 besprochen.

2.1 Auswahl der Zielsprache

Um eine Zielsprache für einen L^AT_EX-Übersetzungslauf zu bestimmen, gibt es verschiedene Möglichkeiten. Die Reihenfolge in der folgenden Liste spiegelt auch die Reihenfolge der Auswertung. Sobald damit eine Zielsprache ermittelt wird, wird eine weitere Auswertung abgebrochen.

1. Definition eines Makros `\olsTargetLanguage` vor Laden des Pakets. Damit ist es beispielsweise möglich, über


```
> latexmk -usepretex="\def\olsTargetLanguage{fr}" document.tex
```

 die Zielsprache Französisch für den Übersetzungslauf festzulegen.
2. Setzen der Paketoption `targetlang`. Verschiedene Werte sind hier möglich:

⁷ Wobei auch in diesen Paketen „Sprache“ ebenfalls für mehrere unterschiedliche und z. T. voneinander unabhängige Konzepte wie Trennungsmuster, zu nutzende Fonts etc. steht

`targetlang = <ISO-Code>`

Die Zielsprache wird direkt durch den Code angegeben.

`targetlang = {job=<n>}`

Die Zielsprache wird aus dem n . Element des Jobnamens bestimmt. Ein Element ist ein mit „-“ (Minus) abgetrennter Teil des Jobnamens, siehe die Dokumentation zum Paket `varsfromjobname`,⁸ das dafür geladen wird. Beispielsweise kann bei `targetlang = {job=2}` über

```
> latex -jobname doc-ru doc.tex
```

Russisch als Zielsprache eingestellt werden.

`targetlang = babel|polyglossia`

Für die Nutzung dieses Schlüsselwerts muss Babel oder Polyglossia vor `langselect` geladen sein. Die Zielsprache wird dann auf die durch das Sprachpaket eingestellte (Haupt-)Sprache gesetzt.⁹

`targetlang = meta`

Die in `\DocumentMetadata` angegebene Sprache als Zielsprache verwendet.

3. Wenn weder `\olsTargetLanguage` definiert noch die Option `targetlang` gesetzt wird, versucht `langselect` selbständig die Zielsprache zu bestimmen, indem es erst die Option `targetlang = babel` und dann (wenn kein Sprachpaket gefunden wurde) die Option `targetlang = meta` ausprobiert.
4. Wenn alles andere versagt, wird die erste Auswahlssprache als Zielsprache gesetzt.

2.2 Optionen

Folgende Optionen stehen zur Verfügung:

`languages = {<Liste von Auswahlssprachen>}` (erforderlich)

Gibt zwei oder drei Auswahlssprachen an. Die Reihenfolge ist relevant für die automatische Generierung der Sprachmakros (siehe Abschnitt 3).

`targetlang = <Sprache>|{job=<n>}|babel|polyglossia|meta`

Siehe Abschnitt 2.1

`prefix = {<prefix>}`

Voreinstellung: 1

Setzt einen Prefix für die Multisprachmakros. Es kann auch ein leerer Prefix gesetzt werden.

`auto = true|false`

Voreinstellung: true

Die eigentlichen Sprachmakros werden automatisch angelegt, siehe Abschnitt 3. Wird `auto = false` gesetzt, werden nur die Makros zur Erzeugung der Sprachmakros zur Verfügung gestellt, nicht die Sprachmakros selbst.

8. auf CTAN als `varsfromjobname`: <https://ctan.org/tex-archive/macros/latex/contrib/varsfromjobname>

9. Beide Werte bewirken exakt das Gleiche: Da Babel und Polyglossia die gleiche Schnittstelle anbieten, ist es egal, welches von beiden Paketen geladen ist. `targetlang = polyglossia` findet also auch Babel.

3 Nutzermakros

`trim = true|false`

Voreinstellung: true

Enfernt Whitespaces am Anfang und Ende der Argumente der Sprachmakros.

`load babel = {\langle Babel Optionen \rangle}`

`load polyglossia = {\langle Polyglossia Optionen \rangle}`

Lädt entweder Babel oder Polyglossia so, dass die Zielsprache von `langselect` als Hauptsprache gesetzt wird. Die hier übergebenen Optionen werden an das jeweilige Sprachpaket weitergeleitet. Wenn hier Optionen angegeben werden, darf kein Sprachpaket vor `langselect` geladen sein.

`map = {\langle Sprache \rangle = \langle Variante \rangle, ...}`

Legt ein Mapping von Sprachen zu einer Sprachvariante fest. Wenn `langselect` mit anderen Paketen interagiert, wird dann die spezifische Sprachvariante statt des ISO-Codes weitergegeben.

Falls Sie beispielsweise Babel von `langselect` aus laden und Ihr Dokument explizit *Britisches Englisch* und *Schweizerdeutsch* nutzt, sichern Sie mit folgendem Code ab, dass Babel korrekte Optionen erhält.

```
1 \documentclass{article}
2 \usepackage[
3   languages={en,de},
4   load=babel,
5   map={en=british,de=nswissgerman}
6 ]
7 \begin{document}
8   \title{\lende{Hallo}{Grüezi}}
9
```

`unified shorthands = true|false`

Voreinstellung: true

Definiert sprachenübergreifende Shorthands zur Unterstützung von `csquotes`.

`bcp47 = true|false`

Voreinstellung: true

Wenn Babel aktiv ist, sorgt `langselect` dafür, dass die von `langselect` genutzten ISO-Codes als Selektornamen für `\selectlanguage` funktionieren. Mit `bcp47 = false` muss die Babelzielsprache über `map` oder von Hand ausgewählt werden. Diese Option hat keine Wirkung, wenn Babel nicht geladen wird.

3 Nutzermakros

3.1 Automatisch Sprachmakros

Wenn Sie nicht die Option `auto = false` gesetzt haben, wird ein Multisprachmakro angelegt. Dessen Name wird durch die Liste der Auswahlssprachen festgelegt. Je nachdem, ob Sie zwei oder mehr Auswahlssprachen angegeben haben, ist der Name

- $\backslash\langle\text{Prefix}\rangle\langle\text{Sprache}_1\rangle\langle\text{Sprache}_2\rangle\{\langle\text{Text}_1\rangle\}\{\langle\text{Text}_2\rangle\}$
- oder
- $\backslash\langle\text{Prefix}\rangle\langle\text{Sprache}_1\rangle\langle\text{Sprache}_2\rangle\langle\text{Sprache}_3\rangle\{\langle\text{Text}_1\rangle\}\{\langle\text{Text}_2\rangle\}\{\langle\text{Text}_3\rangle\}$
- Beispielsweise wurde für diese Dokumentation das Paket mit folgenden Optionen geladen:

```

1 \usepackage[
2   languages={de,en},
3   targetlang={job=2}
4 ]{langselect}

```

Damit steht das Makro $\backslash\text{ldeen}\{\langle\text{Text}_1\rangle\}\{\langle\text{Text}_1\rangle\}$ zur Verfügung. Wenn jetzt die Zielsprache „de“ ausgewählt wird (hier über den Jobnamen, aber es gibt auch andere Möglichkeiten, vgl. Abschnitt 2.1), wird nur $\langle\text{Text}_1\rangle$ in das PDF-Dokument übernommen und $\langle\text{Text}_2\rangle$ verworfen. Bei der Zielsprache „en“ ist es umgekehrt.

Achtung!

Die automatische Generierung der Sprachmakros kann manchmal zu Problemen führen. Beispielsweise ergibt Abchasisch (Code: „ab“) als Erst- und Griechisch (Code: „el“) als Zweisprache den automatischen Namen $\backslash\text{label}$, was zu Fehlermeldungen führt. Nutzen Sie in solchen Fällen die im Abschnitt 3.3 beschriebenen Möglichkeiten.

3.2 Sternvariante

Ein häufiges Problem bei mehreren Sprachversionen ist die Konsistenz nichtsprachabhängiger Daten, insbesondere wenn diese Daten sich öfter ändern. (Denken Sie z. B. an Stundenpläne.)

Für solche Fälle stellt langselect eine Sternvariante der Sprachmakros bereit. In dieser Sternvariante können die Textargumente Platzhalter der Form „@<n>“ enthalten, wobei <n> einen Wert zwischen 1 und 9 darstellt. Hinter dem Makro müssen dann weitere Argumente angegeben werden, und zwar so viele, wie es dem höchsten verwendeten <n> entspricht. Die Sternvariante ersetzt die Platzhalter mit den entsprechenden Argumenten.

```

1 \documentclass{article}
2 \usepackage[languages={en,fr},targetlang=fr]{langselect}
3 \begin{document}
4 \lenfr*{
5   The formula @1 generally refers to the Pythagorean theorem.
6 }{
7   La formule @1 fait généralement référence au théorème de
8   Pythagore.

```

```

9      }{$a^{2}+b^{2}=c^{2}$}
10
11      \lenfr*{
12          Calculate the length of the third side if one side is @1 units long
13          and the hypotenuse is @2 units long.
14      }{
15          Calculez la longueur du troisième
16          côté si un côté mesure @1 unités et l'hypoténuse @2 unités.
17      }{$4$}{$5$}
18      \end{document}

```

La formule $a^2+b^2 = c^2$ fait généralement référence au théorème de Pythagore.

Calculez la longueur du troisième côté si un côté mesure 4 unités et l'hypoténuse 5 unités.

Die Sternvariante der Sprachmakros ist nicht vollständig expandierbar und sollte daher nicht in Kontexten verwendet werden, in denen vollständige Expandierbarkeit erforderlich ist. Dies gilt insbesondere für PDF-Strings, wie sie [hyperref](#) für die Metadaten eines PDF-Dokuments benötigt. Dort wählt [langselect](#) zwar die aktive Sprachfassung aus, Platzhalter wie „@<n>“ werden jedoch nicht ersetzt, und die zugehörigen Ersetzungsargumente bleiben im Eingabestrom erhalten. Verwenden Sie in PDF-Strings daher nur die sternlose Form.

3.3 Individuelle Sprachmakros

Wenn Ihnen die generierten Namen der Sprachmakros nicht gefallen oder diese sogar zu Kollisionen führen, können Sie sie ändern. Am einfachsten ist es, die Paketoption `prefix` zu ändern, so dass beispielsweise mit `prefix = mv` das Sprachmakro für die Deutsch/Englisch-Kombination `\mvdeen` heisst.

Wenn dies nicht ausreichend ist, geben Ihnen die folgenden zwei Makros die vollständige Kontrolle über die Namensbildung.

```
\olsMakeBilingualMacro[⟨Prefix⟩]{⟨Sprache1⟩}{⟨Sprache2⟩}[⟨Basisname⟩]
```

```
\olsMakeTrilingualMacro[⟨Prefix⟩]{⟨Sprache1⟩}{⟨Sprache2⟩}{⟨Sprache3⟩}[⟨Basisname⟩]
```

Diese Makros legen ein bilinguals bzw. trilinguales Sprachmakro für die angegebenen Sprachen an. Wenn `⟨Basisname⟩` angegeben wird, ist der Name des Sprachmakros `\⟨Prefix⟩⟨Basisname⟩`. Wenn das `⟨Prefix⟩`-Argument benutzt wird, bestimmt dieses den Prefix, sonst gelten die Paketoptionen.

4 Interaktion mit anderen Paketen

4.1 Babel/Polyglossia

Das Paket `langselect` ist zunächst einmal unabhängig von Sprachpaketen wie Babel oder Polyglossia. Es kann ohne oder neben diesen Paketen eingesetzt werden. Wenn man sich jedoch die Mühe eines mehrsprachigen Dokuments macht, will man in der Regel auch, dass für die jeweilige Sprache die richtigen Trennungsregeln, Fonteneinstellungen, Datumsformate etc. benutzt werden.¹⁰

Bitte beachten Sie, dass, solange Sie nicht *innerhalb* eines Textes eine andere Sprache (vielleicht ja die Sprache der anderen Sprachversion) nutzen, ihr Dokument aus der Sicht von Babel ein *einsprachiges* Dokument ist. Wenn Sie Babel auf die gewohnte Weise mit allen Sprachen ihrer Sprachversion laden, würde Babel als aktive Sprache aber die zuletzt genannte (oder die per `main` gesetzte) wählen, was nicht unbedingt die Zielsprache von `langselect` ist. Ohne spezielle Maßnahmen könnten Sie so falsche Trennungen und ähnliche Fehler erhalten.

Um dies zu vermeiden, bieten `langselect` verschiedene Möglichkeiten:

1. Sofern die Zielsprache nicht ohnehin aus Babel abgeleitet wird, wählt `langselect` sie zu Dokumentbeginn selbst als aktive Sprache aus. Bei aktivem Babel wird dafür dessen BCP-47-Zuordnung eingeschaltet (Option `bcp47`), so dass die von `langselect` genutzten ISO-Codes als Selektornamen genügen und die Auswahlssprachen einfach mit ihren Babel-Namen geladen werden können. Wenn sie spezielle Sprachvarianten möchten, können sie diese über die `map`-Option wählen, die aber für den Standardfall nicht notwendig ist.
2. Wie oben erwähnt, kann man die Paketoption `targetlang = babel` nutzen. Dann wird die Zielsprache durch die Hauptsprache von Babel bestimmt. Allerdings muss man dann auf die Möglichkeit verzichten, die Zielsprache „von außen“ (also über den Jobnamen oder mit `\olsTargetLanguage`) zu steuern.
3. Umgekehrt lädt `langselect` Babel oder Polyglossia mit Hilfe der Optionen `load babel/` `load polyglossia` und setzt die Zielsprache als Hauptsprache. Weitere Optionen an die Sprachpakete werden als Optionsargument übergeben.
4. Schließlich kann mit den beiden Hooks `langselect/”language/”before` und `langselect/` `language/after` beliebiger Code vor bzw. nach Ausgabe der Argumente des Sprachmakros eingefügt werden. Hier kann z. B. `\foreignlanguage` genutzt werden. Der before-Hook hat als Parameter die Zielsprachenvariante.

4.2 Csquotes

Unterschiedliche Sprachen haben unterschiedliche Konventionen zur Setzung von Satzzeichen. Besonders stört dies bei Anführungszeichen, da diese der Zielsprache folgen sollen, aber durchaus auch außerhalb der Sprachmakros oder in Argumenten der Sternvariante stehen können.

¹⁰ Im Folgenden werde ich mich ausschließlich auf Babel beziehen. Das Gesagte gilt jedoch entsprechend auch für Polyglossia.

Das Paket `csquotes`¹¹ bietet dafür eine Lösung, indem es `\enquote` und andere Makros anbietet, die der Zitatkonvention der aktuellen Sprache folgen.

Zur Erhöhung der Bequemlichkeit unterstützt `langselect` die Nutzung korrekter Anführungszeichen mit sprachunabhängigen Shorthands: Wenn die Option `unified shorthands` wahr ist, wird `csquotes` geladen und die Shorthands `"`` und `"'` auf `\openautoquote` bzw. `\closeautoquote` gesetzt.

Damit werden diese Shorthands in Sprachen wie Deutsch, wo sie bereits definiert sind, überschrieben.

5 FAQ

- **Ist `langselect` stabil?**

Obwohl das Paket mittlerweile die Versionsnummer 1 erreicht hat¹² und zumindest beim Autor intensiv im Einsatz ist, kann ich nicht versprechen, dass es soweit fehlerfrei ist, dass es das Attribut „stabil“ verdient. Ich bitte darum, gefundene Fehler über <https://github.com/werner-matthias/osglecture/issues> zu melden.

- **Wofür steht das „o“ im `\ols`-Prefix?**

Das Paket gehört zu Bundle `osglecture`, von dort ist das „o“ geerbt.

- **Gibt es Anwendungsbeispiele, die über ein Toy-Hello-World hinausgehen?**

Ja, einige. Gucken Sie doch mal zum Beispiel in den Quelltext dieser Dokumentation.

- **Ich habe ein `@1` in der Ausgabe, wo doch eine Argumentersetzung stattfinden sollte.**

Das hat in der Regel eine der beiden folgenden Ursache:

1. Sie nutzen alte Version ($\leq$ vo.6.0) von `langselect`. Dort konnten unterschiedliche Catcodes z. B. durch `\makeatletter` zu Problemen führen.
2. Sie haben den Stern im Sprachmakro vergessen.

- **Kann ich in der Sternversion einen `@n`-Parameter in einer Sprachversion auslassen?**

Nein. Außerdem wäre dann das Argument ja auch offensichtlich nicht mehr gemeinsam.

- **Kann ich `@n`-Parameter in gemischter Reihenfolge oder mehr als einmal verwenden?**

Ja.

- **Ich möchte `@` in einer Sternvariante des Sprachmacros verwenden, aber nicht als Parameter.**

Dann machen Sie es.

11. auf CTAN als `csquotes`: <https://ctan.org/tex-archive/macros/latex/contrib/csquotes>

12. Übrigens als einziges Paket im `osglecture`-Bundle

- **Aber ich brauche @42.**

Nutzen Sie @{}42.

- **Ich nutze Russisch, aber der Font ist nicht richtig.**

Das ist keine Problem von `langselect` (das kümmert sich gar nicht um den Font), sondern wahrscheinlich von Babel, `fontenc`, `fontspec`, etc. Aber vielleicht können die Hooks für Sie nützlich sein, siehe Abschnitt 4.1.

- **Meine Frage erscheint hier nicht.**

Sie können Sie unter <https://github.com/werner-matthias/osglecture/issues> stellen.

6 Implementation

```

1 \NeedsTeXFormat{LaTeX2e}[2022/06/01]
2 \def\packagename{langselect}
3 \ProvidesExplPackage{langselect}
4 {2026-09-18}
5 {v1.1.3}
6 {support for different languages versions from a common source}
7 \ExplSyntaxOff
8 \ExplSyntaxOn

```

`\olsIsoTags` enthält alle gültigen ISO 639-1-Codetags.

```

9 \seq_const_from_clist:cn{\olsIsoTags}{aa,ab,ae,af,ak,am,an,ar,as,av,ay,az,ba,be,
10   bg,bh,bi,bm,bn,bo,br,bs,ca,ce,ch,co,cr,cs,cu,cv,cy,da,de,dv,dz,ee,el,en,eo,es,
11   et,eu,fa,ff,fi,fj,fo,fr,fy,ga,gd,gl,gn,gu,gv,ha,he,hi,ho,hr,ht,hu,hy,hz,ia,id,
12   ie,ig,ii,ik,io,is,it,iu,ja,jv,ka,kg,ki,kj,kl,km,kn,ko,kr,ks,ku,kv,kw,ky,la,
13   lb,lg,li,ln,lo,lt,lu,lv,mg,mh,mi,mk,ml,mn,mr,ms,mt,my,na,nb,nd,ne,ng,nl,nn,no,
14   nr,nv,ny,oc,oj,om,or,os,pa,pi,pl,ps,pt,qu,rm,rn,ro,ru,rw,sa,sc,sd,se,sg,si,sk,
15   sl,sm,sn,so,sq,sr,ss,st,su,sv,sw,ta,te,tg,th,ti,tk,tl,tn,to,tr,ts,tt,tw,ty,ug,
16   uk,ur,uz,ve,vi,vo,wa,wo,xh,yi,yo,za,zh,zu}
17
18 \newcommand\IfTagIsValidF[2]{%
19   \seq_if_in:ceF{\olsIsoTags}{#1}{#2}
20 }
21 \RequirePackage{varsfromjobname}[2025/08/03]

```

Das Ergebnisse von `\varsfromjobname` und `\str_range:Nnn` haben Catcode 12, aber ISO-Tags und Optionen haben Catcode 11. Wir vereinheitlichen durch einen Wrapper auf Catcode 11.

```

22 \newcommand\olsChangeCatEleven[1]{
23   \def\tmpa{#1}
24   \edef\tmpb{

```

```

25     \noexpand\scantokens{
26         \noexpand\edef\noexpand\olsCatElven{\tmpa}
27     }
28 }
29 \tmpb
30 }

```

Bei Bedarf bilden wir Spachcode auf Sprachvarianten ab. Bei Interaktionen mit externen Code wird dann die Sprachvariante weitergegeben.

```

31 \cs_new:Npn \ols_use_map:n #1 {
32     \cs_if_exist_use:cF { olslangmap / #1 }{ #1 }
33 }

```

`\ols_babel_name_from_ini:nN` liest das Feld `name.babel` aus `babel-⟨Code⟩.ini`, das mehrere Babel-Namen für dieselbe Sprache enthalten kann. Häufig sind das nur alternative Bezeichnungen für das Gleiche. Für Deutsch gibt es aber zwei tatsächlich verschiedene Rechtschreibungen (`german` = alte, `ngerman` = reformierte Orthographie). Ist `ngerman` unter den Kandidaten, wird es deshalb bevorzugt (heutiger Standard für Deutsch). Ansonsten wird der letzte Eintrag verwendet. Falls jemand, der sich mit den Sprachen besser auskennt als ich, eine weitere notwendigen Sonderfall entdeckt: Bitte ein entsprechendes Issue aufmachen.

```

34 \ior_new:N \g_ols_ini_ior
35 \seq_new:N \ols_ini_seq
36 \seq_new:N \ols_ini_names_seq
37 \cs_new_protected:Npn \ols_babel_name_from_ini:nN #1 #2
38 {
39     \tl_clear:N #2
40     \ior_open:NnT \g_ols_ini_ior { babel- #1 .ini }
41     {
42         \ior_str_map_inline:Nn \g_ols_ini_ior
43         {
44             \str_if_eq:eeT { \str_range:nnn {##1} {1} {10} } { name.babel }
45             {
46                 \seq_set_split:Nnn \ols_ini_seq { = } { ##1 }
47                 \seq_set_split:Nne \ols_ini_names_seq { ~ }
48                 { \tl_trim_spaces:e { \seq_item:Nn \ols_ini_seq {2} } } }
49                 \seq_if_empty:NF \ols_ini_names_seq
50                 {
51                     \seq_if_in:NeTF \ols_ini_names_seq { \tl_to_str:n {ngerman}
52                     ↪ }
53                     { \tl_set:Nn #2 { ngerman } }
54                     { \tl_set:Ne #2 { \seq_item:Nn \ols_ini_names_seq {-1} } } }
55                 }
56                 \ior_map_break:
57             }
58         }
59     }
60 }

```

```

57     }
58     \ior_close:N \g_ols_ini_ior
59 }

```

Zeilen aus `\ior_str_map_inline` haben Catcode 12. Babel vergleicht Sprachoptionen mit `\ifx` (catcode-empfindlich), daher machen wir aus dem Namen wieder Buchstaben mit Standard-Catcodes.

```

60     \tl_if_empty:NF #2 { \tl_set_rescan:NnV #2 { } #2 }
61 }

```

Babel hat eine (in den verschiedenen Versionen unterschiedlich sortierte) Liste von Sprachennamen. Deshalb gehen wir zur „Konkurrenz“ und nutzen in `\ols_babel_name_from_polyglossia:nN polyglossias gloss-<Code>.ldf`-Dateien als Datenquelle: diese enthalten für die Sprachen mit mehreren Babel-Namen (wie Deutsch) ein explizites `babelname`-Feld mit einem einzigen, eindeutigen Wert. `polyglossia` selbst wird dabei nicht geladen, nur seine Textdateien werden wie schon bei `\ols_babel_name_from_ini:nN` zeilenweise eingelesen.

```

62 \tl_new:N \ols_ini_module_tl
63 \seq_new:N \ols_ini_regex_seq
64 \cs_new_protected:Npn \ols_babel_name_from_polyglossia:nN #1 #2
65 {
66     \tl_clear:N #2
67     \tl_clear:N \ols_ini_module_tl
68     \ior_open:NnT \g_ols_ini_ior { gloss- #1 .ldf }
69     {
70         \ior_str_map_inline:Nn \g_ols_ini_ior
71         {
72             \regex_extract_once:nnN { InheritGlossFile\{([a-zA-Z])+\} } {##1}
73             ↪ \ols_ini_regex_seq
74             \seq_if_empty:NF \ols_ini_regex_seq
75             {
76                 \tl_set:Nc \ols_ini_module_tl { \seq_item:Nn \ols_ini_regex_seq
77                 ↪ {2} }
78                 \ior_map_break:
79             }
80         }
81     }
82     \ior_close:N \g_ols_ini_ior
83 }

```

Der Modulname selbst ist der Babel-Name, sofern die Zielfeile kein eigenes `babelname` definiert – das betrifft die meisten Sprachen, bei denen Polyglossia- und Babel-Name ohnehin identisch sind.

```

81     \tl_if_empty:NF \ols_ini_module_tl
82     {

```

6 Implementation

```

83     \ior_open:NnT \g_ols_ini_ior { gloss- \ols_ini_module_tl .ldf }
84     {
85         \ior_str_map_inline:Nn \g_ols_ini_ior
86         {
87             \regex_extract_once:nnN { babelname\s*=\s*([a-zA-Z]+) } {##1}
88             ↪ \ols_ini_regex_seq
89             \seq_if_empty:NF \ols_ini_regex_seq
90             {
91                 \tl_set:Ne #2 { \seq_item:Nn \ols_ini_regex_seq {2} }
92                 \ior_map_break:
93             }
94             \ior_close:N \g_ols_ini_ior
95         }
96         \tl_if_empty:NT #2 { \tl_set_eq:NN #2 \ols_ini_module_tl }
97     }
98     \tl_if_empty:NF #2 { \tl_set_rescan:NnV #2 { } #2 }
99 }

```

`\ols_babel_name:nN` versucht zuerst `polyglossias` eindeutige Angabe und fällt nur, wenn diese fehlt (etwa weil `polyglossia` in der genutzten Installation gar nicht vorhanden ist, oder für eine Sprache ohne eigene gloss-Datei), auf Babels eigene, mitunter mehrdeutige .ini-Liste zurück.

```

100 \cs_new_protected:Npn \ols_babel_name:nN #1 #2
101 {
102     \ols_babel_name_from_polyglossia:nN {#1} #2
103     \tl_if_empty:NT #2 { \ols_babel_name_from_ini:nN {#1} #2 }
104 }

```

Wir definieren einige Optionen. Alle außer `languages` sind optional. Einige Ergebnisse werden wir in Variablen speichern, die wir zuerst anlegen.

```

105 \bool_new:N\ols_load_babel
106 \bool_new:N\ols_load_polyglossia

```

Wahr, wenn die Zielsprache aus dem aktiven Sprachpaket abgeleitet wurde. Dann ist dessen Hauptsprache bereits die Zielsprache und darf nicht erneut (womöglich mit einem Code statt eines Sprachnamens) ausgewählt werden.

```

107 \bool_new:N\ols_target_from_langpkg
108 \DeclareKeys{
109     languages.clist_gset:N = \ols_langs,
110     languages.usage=load,
111     targetlang.code = {

```

Die Option `targetlang` wird nur ausgewertet, wenn die Zielsprache nicht bereits durch Definition von `\olsTargetLanguage` gesetzt wurde..

```
112 \cs_if_exist:NF\olsTargetLanguage{
```

Wird ein Element des Jobnames angegeben?

```
113 \bool_if:nT{
114   \str_if_eq_p:ee{\str_range:nnn{#1}{1}{3}}{job}
115 }{%
116   \olsChangeCatEleven{\getfromjobname{\str_range:Nnn{#1}{-1}{-1}}}
117   \xdef\olsTargetLanguage{\olsCatElven}
118 }
```

Soll die Zielsprache von `babel` oder `polyglossia` abgeleitet werden?

```
119 \bool_if:nT{
120   \str_if_eq_p:nn{#1}{babel} || \str_if_eq_p:nn{#1}{polyglossia}
121 }{
122   \str_if_eq:eeF{\BCPdata{main.language}}{ }{
123     \edef\olsTargetLanguage{\BCPdata{main.language}}
124     \bool_set_true:N\ols_target_from_langpkg
125   }
126 }
```

Soll die Sprache aus dem Metadaten genutzt werden?

```
127 \str_if_eq:nnTF{#1}{meta}{
128   \IfDocumentMetadataTF{
129     \edef\tmpa{\GetDocumentProperty{document/lang}}
130     \edef\tmpb{\str_range:Nnn\tmpa{1}{2}}
131     \olsChangeCatEleven{\tmpb}
132     \edef\olsTargetLanguage{\olsCatElven}
133   }{
134     \PackageWarningNoLine{\packagename}{No~meta~data~provided}
135   }
136 }
137 }
```

Wenn bisher noch keine Sprache zugewiesen wurde, wird das Argument als Sprache angenommen. Es wird später überprüft, ob es gültig ist.

```
138 \cs_if_exist:NF \olsTargetLanguage {
139   \edef\olsTargetLanguage{#1}
140 }
141 },
```

Prefix für die Nutzer-Sprachbefehle.

```
142 targetlang.usage=load,
143 prefix.store=\ols_sprefix,
144 prefix.initial:n=l,
145 prefix.usage=load,
```

Automatische Erzeugung der Sprachbefehle?

```
146 auto.bool_set:N=\ols_generate,
147 auto.usage=load,
148 auto.initial:n=true,
```

Automatische Entfernung von Randleerzeichen?

```
149 trim.bool_set:N=\ols_trim,
150 trim.usage=load,
151 trim.initial:n=true,
```

Abbildung auf Sprachvarianten.

```
152 map.code:n = {
153   \prop_new:N \ols_lang_pl
154   \prop_clear:N \ols_lang_pl
155   \prop_set_from_keyval:Nn \ols_lang_pl {#1}
```

Da Property-Listen nicht expandierbar sind, setzen wir die Einträge auf gewöhnlich Makros um.

```
156   \prop_map_inline:Nn \ols_lang_pl
157   { \cs_gset:cpn { olslangmap / \tl_to_str:n {##1} } {##2} }
158   },
```

Laden von `babel` oder `polyglossia` mit Optionen. Ein leerer Wert (`load babel ohne =`) **darf nicht** als leere Option weitergereicht werden.

```
159 load~babel .code:n = {
160   \bool_set_true:N \ols_load_babel
161   \tl_if_empty:nF {#1} { \PassOptionsToPackage{#1}{babel} }
162 },
163 load~polyglossia .code:n = {
164   \bool_set_true:N \ols_load_polyglossia
165   \tl_if_empty:nF {#1} { \PassOptionsToPackage{#1}{polyglossia} }
166 },
167 load.usage = load,
168 unified~shorthands.bool_set:N=\ols_shorthands,
169 unified~shorthands.initial:n=true,
170 unified~shorthands.usage=load,
```

Sollen die von `langselect` genutzten ISO-Codes bei einem aktiven `babel` als BCP-47-Selektornamen zugelassen werden?

```
171 bcp47.bool_set:N=\ols_bcptags,
172 bcp47.initial:n=true,
173 bcp47.usage=load
174 }%
175 \ProcessKeyOptions\relax
```

6 Implementation

Wenn über `\olsTargetLanguage` oder die `targetlang` keine Zielsprache zugewiesen wurde, untersuchen wir noch einmal, ob über Sprachpakete oder `\DocumentMetadata` eine Sprache identifiziert werden kann.

```
176 \cs_if_exist:NF \olsTargetLanguage {
177   \str_if_eq:eeF{\BCPdata{main.language}}{}{
178     \edef\olsTargetLanguage{\BCPdata{main.language}}
179     \bool_set_true:N\ols_target_from_langpkg
180   }
181 }
182 \cs_if_exist:NF \olsTargetLanguage {
183   \IfDocumentMetadataT{
184     \edef\tmpa{\GetDocumentProperty{document/lang}}
185     \edef\tmpb{\str_range:Nnn\tmpa{1}{2}}
186     \olsChangeCatEleven{\tmpb}
187     \edef\olsTargetLanguage{\olsCatElven}
188   }
189 }
```

Wir führen verschiedene Checks durch. Sind mindestens zwei Auswahlssprachen angegeben?

```
190 \int_compare:nNtT{\clist_count:N\ols_langs} < {2}{
191   \PackageError{\packagename}{No-sufficient-number-of-selectable
192     \MessageBreak~languages~provided}{
193     You~have~to~provide~a~list~of~two~or~three~\MessageBreak
194     languages~via~'language'~option.}
195   \aftergroup\endinput
196 }
```

Im Moment kann `langselect` nicht mehr als drei Auswahlssprachen verarbeiten. Wir warnen daher, wenn mehr als drei Sprachen angegeben werden.

```
197 \int_compare:nNtT{\clist_count:N\ols_langs} > {3}{
198   \PackageWarningNoLine{\packagename}{Too-many-selectable-languages-provided.
199     \MessageBreak I~will~ignore~the~superfluous~languages}
200 }
```

Als nächstes überprüfen wir, ob alle Auswahlssprachen gültige ISO 639-1 Codes sind.

```
201 \clist_map_inline:Nn\ols_langs {
202   \IfTagIsValidF{#1}{
203     \PackageError{\packagename}{Couldn't-resolve-selectable-language-'#1'}{
204       Use-valid-ISO-639-1-code-in-option-'languages'.}
205     \aftergroup\endinput
206   }
207 }
```

Nun prüfen wir, ob eine valide Zielsprache ermittelt werden konnte.

6 Implementation

```

208 \cs_if_exist:NF \olsTargetLanguage {
209   \PackageWarningNoLine{\packagename}{Can't~identify~any~target~language.
210     \MessageBreak Falling-back-to~'\clist_item:Nn \ols_langs {1}'}
211   \edef\olsTargetLanguage{\clist_item:Nn \ols_langs {1}}
212   \IfTagIsValidF{\olsTargetLanguage}{
213     \PackageWarningNoLine{\packagename}{Do~not~recognize~language~}
214     ↪ '\olsTargetLanguage'.
215     \MessageBreak Take~'\clist_item:Nn \ols_langs {1}'~as~replacement}
216   \edef\olsTargetLanguage{\clist_item:Nn \ols_langs {1}}
217 }

```

Die Zielsprache sollte in der Liste von Auswahl Sprachen enthalten sein.

```

217 \clist_if_in:NVF\ols_langs{\olsTargetLanguage}{
218   \PackageError{\packagename}{Target~language~'\olsTargetLanguage'~is~}
219   ↪ not~\MessageBreak
220   in~the~list~of~selectable~languages}{
221   Check~option~'language'~and~'targetlang'.}
222 }

```

Babel und Polyglossia können nicht beiden geladen werden.

```

222 \bool_if:nT{
223   \ols_load_babel && \ols_load_polyglossia
224 }{
225   \bool_set_false:N \ols_load_babel
226   \bool_set_false:N \ols_load_polyglossia
227   \PackageError{\packagename}{The~options~'load~babel'~and~'load~polyglossia'
228     \MessageBreak
229     ↪ are~mutual~exclusive,~you~can't~use~both}{Use~one~load~option,~only.}
230 }

```

Wenn bereits ein Sprachpaket aktiv ist, wird babel/polyglossia nicht selbst geladen. Die Zielsprache wird später aktiviert.

```

230 \bool_if:nT{
231   ( \cs_if_exist_p:c{ver@babel.sty} || \cs_if_exist_p:c{ver@polyglossia.sty} )
232   &&
233   ( \ols_load_babel || \ols_load_polyglossia )
234 }{
235   \bool_set_false:N \ols_load_babel
236   \bool_set_false:N \ols_load_polyglossia
237   \PackageWarningNoLine{\packagename}{Can't~load~babel/polyglossia~for
238     ~you,since~\MessageBreak
239     a~language~package~is~already~loaded.\MessageBreak
240     I'll~ignore~that~option}
241 }

```

6 Implementation

`\ols_mapped_target_lang` ist die Bezeichnung der Zielsprache, wie sie ggf. an externen Code weitergegeben wird, d.h. `\DocumentMetadata`, Sprachpakete und Hooks.

```
242 \edef\ols_mapped_target_lang{\ols_use_map:n{\olsTargetLanguage}}
```

Jetzt sollte die Zielsprache feststehen. Wir geben die Informationen ins Log.

```
243 \PackageNoteNoLine{\packagename}{Target~language~is~'\olsTargetLanguage'~}
    ↳ (\ols_mapped_target_lang)}
```

Wir korrigieren noch ggf. die über `\DocumentMetadata` gesetzten Metadaten auf die der Zielsprache, so dass beispielsweise Babel keine falsche Hauptsprache ableitet.

```
244 \IfDocumentMetadataT{
245   \edef\olsDocLang{\exp_args:N\str_range:nnn{\GetDocumentProperty{
    ↳ document/lang}}{1}{2}}
246   \bool_if:nF{
247     \str_if_eq_p:ee{\olsDocLang}{\olsTargetLanguage}
248   }{
249     \PackageWarningNoLine{\packagename}{Target~language~'\olsTargetLanguage'~
250       doesn't~comply~with~\MessageBreak \c_backslash_str
251       DocumentMetadata~'\GetDocumentProperty{document/lang}'.~\MessageBreak
252       I'll~try~to~overwrite~the~meta~data}
253     \AddToDocumentProperties[document]{lang}{\olsTargetLanguage}
254   }
255 }
```

Da der Name des Sprachmakros nicht bekannt ist, sind das Patchen oder generische Hooks komplizierter. Wir stellen deshalb eigene Hooks zur Verfügung.

```
256 \NewHookWithArguments{langselect/language/before}{1}
257 \NewReversedHook{langselect/language/after}
258 \NewHookWithArguments{langselect/argument}{2}
```

Ein Boolean zur Markierung, ob mehr als zwei Auswahl Sprachen angegeben wurden.

```
259 \bool_new:c{ols_trilang}
260 \int_compare:nNnTF{\clist_count:N{\ols_langs}} > {2}{
261   \bool_set_true:c{ols_trilang}
262 }{
263   \bool_set_false:c{ols_trilang}
264 }
265 \def\olsFrstLanguage{\clist_item:Nn \ols_langs{1}}
266 \def\olsScndLanguage{\clist_item:Nn \ols_langs{2}}
267 \bool_if:cT{ols_trilang}{
268   \def\olsThrdLanguage{\clist_item:Nn \ols_langs{3}}
269 }
270 \bool_if:NTF\ols_trim{
```

```

271 \let\ols_trim:n=\tl_trim_spaces:n
272 }{
273 \let\ols_trim:n=\relax
274 }

```

Wenn `\DeclarePresInitialBranchingCommand` existiert (also die `presitemize`-Umgebung definiert ist), wird das Sprachmakro dafür registriert.

```

275 \cs_new_protected:Npn \ols_register_pres_initial_branching:cn #1#2
276 {
277   \cs_if_exist:NTF \DeclarePresInitialBranchingCommand
278   {
279     \exp_args:Nc \DeclarePresInitialBranchingCommand {#1} {#2}
280   }
281   {
282     \AddToHook { package/osglecture-presitemize/after }
283     {
284       \exp_args:Nc \DeclarePresInitialBranchingCommand {#1} {#2}
285     }
286   }
287 }
288

```

Für PDF-Strings registrieren wir eine vereinfachte Fassung jedes Sprachmakros bei `hyperref`. Sie konsumiert auch einen Stern, wählt aber lediglich den aktiven Sprachzweig und führt weder Hooks noch Ersetzungen aus.

```

289 \cs_new_protected:Npn \ols_register_pdfstring_bilingual:cn #1#2
290 {
291   \AddToHook { package/hyperref/after }
292   {
293     \pdfstringdefDisableCommands
294     {
295       \int_compare:nNnTF {#2} = {1}
296       {
297         \ExpandArgs{c}\RenewExpandableDocumentCommand {#1}
298         {s +m +m} {##2}
299       }
300       {
301         \ExpandArgs{c}\RenewExpandableDocumentCommand {#1}
302         {s +m +m} {##3}
303       }
304     }
305   }
306 }
307 \cs_new_protected:Npn \ols_register_pdfstring_trilingual:cn #1#2

```

```

308 {
309   \AddToHook { package/hyperref/after }
310   {
311     \pdfstringdefDisableCommands
312     {
313       \int_case:nn {#2}
314       {
315         {1}
316         {
317           \ExpandArgs{c}\RenewExpandableDocumentCommand {#1}
318             {s +m +m +m} {##2}
319         }
320         {2}
321         {
322           \ExpandArgs{c}\RenewExpandableDocumentCommand {#1}
323             {s +m +m +m} {##3}
324         }
325         {3}
326         {
327           \ExpandArgs{c}\RenewExpandableDocumentCommand {#1}
328             {s +m +m +m} {##4}
329         }
330       }
331     }
332   }
333 }

```

Die Makros `\olsMakeBilingualMacro` bzw. `\olsMakeTrilingualMacro` dienen zur Generierung der eigentlichen Sprachmakros.

#1: prefix, #2,#3: selectable languages, #4: base name

```

334 \NewDocumentCommand{\olsMakeBilingualMacro}{0{\ols_sprefix} m m o}{
335   \str_if_eq:eeT{\olsTargetLanguage}{#2} {
336     \IfValueTF{#4}{
337       \ExpandArgs{c}\NewExpandableDocumentCommand{#1#4}{s +m +m}{
338         \UseHook{langselect/language/before}{\ols_mapped_target_lang}
339         \IfBooleanTF{##1}{
340           \ols_trim:n{\olsProcessArgs{##2}}
341         }{
342           \ols_trim:n{##2}
343         }
344       }
345     }
346   }
347   }{

```

6 Implementation

```
348     \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3}{s +m +m}{
349         \UseHook{langselect/language/before}{\ols_mapped_target_lang}
350         \IfBooleanTF{##1}{
351             \ols_trim:n{\olsProcessArgs{##2}}
352         }{
353             \ols_trim:n{##2}
354             \UseHook{langselect/language/after}
355         }
356     }
357 }
358 }
359 \str_if_eq:eeT{\olsTargetLanguage}{#3}{
360     \IfValueTF{#4}{
361         \ExpandArgs{c}\NewExpandableDocumentCommand{#1#4}{s +m +m}{
362             \UseHook{langselect/language/before}{\ols_mapped_target_lang}
363             \IfBooleanTF{##1}{
364                 \ols_trim:n{\olsProcessArgs{##3}}
365             }{
366                 \ols_trim:n{##3}
367                 \UseHook{langselect/language/after}
368             }
369         }
370     }{
371         \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3}{s +m +m}{
372             \UseHook{langselect/language/before}{\ols_mapped_target_lang}
373             \IfBooleanTF{##1}{
374                 \ols_trim:n{\olsProcessArgs{##3}}
375             }{
376                 \ols_trim:n{##3}\UseHook{langselect/language/after}
377             }
378         }
379     }
380 }
381 \IfValueTF{#4}
382 {
383     \ols_register_pres_initial_branching:cn {#1#4} {2}
384     \str_if_eq:eeTF {\olsTargetLanguage} {#2}
385     { \ols_register_pdfstring_bilingual:cn {#1#4} {1} }
386     { \ols_register_pdfstring_bilingual:cn {#1#4} {2} }
387 }
388 {
389     \ols_register_pres_initial_branching:cn {#1#2#3} {2}
390     \str_if_eq:eeTF {\olsTargetLanguage} {#2}
391     { \ols_register_pdfstring_bilingual:cn {#1#2#3} {1} }
```

6 Implementation

```
392     { \ols_register_pdfstring_bilingual:cn {#1#2#3} {2} }
393   }
394 }
```

Nun das Gleiche nochmal für *drei* Sprachen.

#1: prefix, #2...#4: selectable languages, #5: base name

```
395 \NewDocumentCommand{\olsMakeTrilingualMacro}{0{\ols_sprefix} m m m o}{
396   \str_if_eq:eeT{\olsTargetLanguage}{#2} {
397     \IfValueTF{#5}{
398       \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{
399         \UseHook{langselect/language/before}{\ols_mapped_target_lang}
400         \IfBooleanTF{##1}{
401           \ols_trim:n{\olsProcessArgs{##2}}
402         }{
403           \ols_trim:n{##2}
404           \UseHook{langselect/language/after}
405         }
406       }
407     }{
408       \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
409         \UseHook{langselect/language/before}{\ols_mapped_target_lang}
410         \IfBooleanTF{##1}{
411           \ols_trim:n{\olsProcessArgs{##2}}
412         }{
413           \ols_trim:n{##2}
414           \UseHook{langselect/language/after}
415         }
416       }
417     }
418   }
419   \str_if_eq:eeT{\olsTargetLanguage}{#3} {
420     \IfValueTF{#5}{
421       \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{
422         \UseHook{langselect/language/before}{\ols_mapped_target_lang}
423         \IfBooleanTF{##1}{
424           \ols_trim:n{\olsProcessArgs{##3}}
425         }{
426           \ols_trim:n{##3}
427           \UseHook{langselect/language/after}
428         }
429       }
430     }{
431       \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
432         \UseHook{langselect/language/before}{\ols_mapped_target_lang}
```

6 Implementation

```

433     \IfBooleanTF{##1}{
434         \ols_trim:n{\olsProcessArgs{##3}}
435     }{
436         \ols_trim:n{##3}
437         \UseHook{langselect/language/after}
438     }
439 }
440 }
441 }
442 \str_if_eq:eeT{\olsTargetLanguage}{#4} {
443     \IfValueTF{#5}{
444         \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{
445             \UseHook{langselect/language/before}{\ols_mapped_target_lang}
446             \IfBooleanTF{##1}{
447                 \ols_trim:n{\olsProcessArgs{##4}}
448             }{
449                 \ols_trim:n{##4}
450                 \UseHook{langselect/language/after}
451             }
452         }
453     }{
454         \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
455             \UseHook{langselect/language/before}{\ols_mapped_target_lang}
456             \IfBooleanTF{##1}{
457                 \ols_trim:n{\olsProcessArgs{##4}}
458             }{
459                 \ols_trim:n{##4}
460                 \UseHook{langselect/language/after}
461             }
462         }
463     }
464 }
465 \IfValueTF{#5}
466 {
467     \ols_register_pres_initial_branching:cn {#1#5} {3}
468     \str_if_eq:eeTF {\olsTargetLanguage} {#2}
469     { \ols_register_pdfstring_trilingual:cn {#1#5} {1} }
470     {
471         \str_if_eq:eeTF {\olsTargetLanguage} {#3}
472         { \ols_register_pdfstring_trilingual:cn {#1#5} {2} }
473         { \ols_register_pdfstring_trilingual:cn {#1#5} {3} }
474     }
475 }
476 {

```

6 Implementation

```

477 \ols_register_pres_initial_branching:cn {#1#2#3#4} {3}
478 \str_if_eq:eeTF {\olsTargetLanguage} {#2}
479 { \ols_register_pdfstring_trilingual:cn {#1#2#3#4} {1} }
480 {
481   \str_if_eq:eeTF {\olsTargetLanguage} {#3}
482   { \ols_register_pdfstring_trilingual:cn {#1#2#3#4} {2} }
483   { \ols_register_pdfstring_trilingual:cn {#1#2#3#4} {3} }
484 }
485 }
486 }

```

Falls die automatische Erzeugung nicht deaktiviert ist, werden die konkreten Sprachmakros angelegt. Die Namen sind $\langle \text{Prefix} \rangle \langle \text{Sprache}_1 \rangle \langle \text{Sprache}_2 \rangle$ bzw. $\langle \text{Prefix} \rangle \langle \text{Sprache}_1 \rangle \langle \text{Sprache}_2 \rangle \langle \text{Sprache}_3 \rangle$.

```

487 \bool_if:cTF{\ols_generate}{
488   \bool_if:cTF{\ols_trilang}{
489     \olsMakeTrilingualMacro{\olsFrstLanguage}{\olsScndLanguage}{
490       ↪ \olsThrdLanguage}
491   }{
492     \olsMakeBilingualMacro{\olsFrstLanguage}{\olsScndLanguage}
493   }{
494     \PackageInfoNoLine{\packagename}{No~language~macros~generated.\MessageBreak
495     Use~\string\olsMakeBilingualMacro~or~\string\olsMakeTrilingualMacro~to~
496     ↪ generate.}
497   }

```

Für die Sternvarianten der Sprachmakros legen wir uns ein Makro an, das die folgenden Argumente verarbeitet.

```

497 \seq_new:N \ols_args_seq
498 \int_new:N \ols_num_of_args
499 \tl_new:N \ols_format_str_tl
500 \NewDocumentCommand \olsProcessArgs { +m } {
501   \tl_gset:Nn \ols_format_str_tl{#1\UseHook{langselect/language/after}}
502   \int_zero:N \ols_num_of_args

```

Suche den höchsten Wert n , für den $@n$ im Formatstring vorkommt.

```

503 \int_step_inline:nnnn { 9 } { -1 } { 1 }
504 {
505   \regex_if_match:nnT {@##1}{ #1 }
506   {
507     \int_set:Nn \ols_num_of_args { ##1 } \prg_break:
508   }
509 }

```

Wenn kein Platzhalterzeichen gefunden wurde, gib den Text in Originalform aus. Ansonsten wird das Ersetzungsmakro gerufen.

```

510 \int_compare:nNnTF {\ols_num_of_args} = { 0 }
511 { #1 }{
512   \seq_clear:N \ols_args_seq
513   \ols_collect_replace:n \ols_num_of_args
514 }
515 }

```

`\ols_collect_replace` erhält *ein* Argument übergeben, nämlich die Anzahl der verbleibenden Ersetzungstexte. Es konsumiert aber *zwei* Argumente. Dieses zweite holt es sich aus dem nach dem Makro liegenden Tokenstrom, wo es den jeweilig nächsten Ersetzungstext findet. Dieser wird in einer Tokensequenz gespeichert und `\ols_collect_replace` erneut aufgerufen.

```

516 \cs_new_protected:Npn \ols_collect_replace:n #1 #2{
517   \IfHookEmptyTF{langselect/argument}{
518     \seq_put_right:Nn \ols_args_seq { #2 }
519   }{
520     \seq_put_right:Nn \ols_args_seq {
521       \UseHook{langselect/argument}{\ols_mapped_target_lang}{#2}
522     }
523   }
524   \int_compare:nNnTF { #1 } = { 1 }
525   {

```

Wenn alle Ersetzungstexte eingelesen wurden, werden sie nacheinander auf die Platzhalterausdrücke angewendet.

```

526   \seq_map_indexed_inline:Nn \ols_args_seq
527   {
528     \tl_set:Nn \ols_replacement_tl { ##2 }
529
530     \regex_replace_all:nnN
531     { @##1 }
532     { \u{\ols_replacement_tl} }
533     \ols_format_str_tl
534   }
535   \ols_format_str_tl
536 }{
537   \ols_collect_replace:n { \int_eval:n { #1 - 1 } }
538 }
539 }

```

Wenn gewünscht, lädt `langselect` Babel und setzt die Zielsprache als Hauptsprache. Babel erwartet dafür Sprachnamen, keine ISO-Codes. Für jede Auswahlssprache ohne `map`-Eintrag wird der Name deshalb ermittelt und als impliziter Map-Eintrag hinterlegt – auch wenn `load babel`

selbst nicht gesetzt ist, aber `bcp47` oder `load polyglossia` aktiv sind, denn dann brauchen die BCP-47-Anbindung bzw. `\setdefaultlanguage` weiter unten dieselben Einträge. Nur wenn `load babel` oder `load polyglossia` gesetzt ist und kein Name ermittelt werden kann, gibt es einen Fehler; ist nur `bcp47` gesetzt, bleibt eine unauflösbare Sprache einfach unangetastet, Babel entscheidet dann wie ohne `langselect`.

```

540 \tl_new:N \ols_babel_name_tl
541 \tl_new:N \ols_babel_opts_tl
542 \bool_if:nT { \ols_load_babel || \ols_load_polyglossia || \ols_bcptags } {
543   \clist_map_inline:Nn \ols_langs {
544     \cs_if_exist:cF { olslangmap / \tl_to_str:n {#1} }{
545       \ols_babel_name:nN {#1} \ols_babel_name_tl
546       \tl_if_empty:NTF \ols_babel_name_tl {
547         \bool_if:nT { \ols_load_babel || \ols_load_polyglossia } {
548           \PackageError{\packagename}{Can't find a language name for '#1'}{
549             Add a '~map' entry such as ~map={#1=...} to the package options,
550             \MessageBreak
551             → or load babel/polyglossia yourself before ~\packagename.}
552         }{
553           \cs_gset:cpe { olslangmap / \tl_to_str:n {#1} } { \ols_babel_name_tl }
554         }
555       }
556     }

```

`\ols_mapped_target_lang` nach den neuen Map-Einträgen neu berechnen – @1 und @2 unten brauchen beide den aktualisierten Wert. `\selectlanguage\setdefaultlanguage`

```

557   \edef\ols_mapped_target_lang{\ols_use_map:n{\olsTargetLanguage}}
558 }
559 \bool_if:NT \ols_load_babel{

```

Alle Auswahl Sprachen unter ihren Babel-Namen laden, die Zielsprache als Hauptsprache.

```

560   \tl_clear:N \ols_babel_opts_tl
561   \clist_map_inline:Nn \ols_langs
562     { \tl_put_right:Ne \ols_babel_opts_tl { \ols_use_map:n {#1} , } }
563   \tl_put_right:Ne \ols_babel_opts_tl { main = \ols_mapped_target_lang }
564   \exp_args:NV \PassOptionsToPackage \ols_babel_opts_tl { babel }
565   \RequirePackage { babel }
566   \bool_if:nT{
567     \ols_shorthands && !\str_if_eq_p:ee{\olsTargetLanguage}{de}}
568   {
569     \languageshorthands{ngerman}
570   }
571 }

```

```

572 \bool_if:NT\ols_load_polyglossia{
573   \RequirePackage{polyglossia}
574 }

```

Für die Shorthand-Logik brauchen wir csquotes.

```

575 \bool_if:NT\ols_shorthands {
576   \RequirePackage[autostyle=true]{csquotes}
577 }

```

Die folgende Initialisierung kann erst geschehen, wenn feststeht, welches Sprachpaket (falls überhaupt eines) letztlich aktiv ist. Da andere Pakete oder Klassen Sprachentscheidungen typischerweise nach hinten verschieben, wird alles mit `\AddToHook[begindocument/before]` bzw. `\AddToHook[begindocument]` verschoben.

```

578 \AddToHook{begindocument/before}{%

```

Aktiviere ISO-Codes vor der Sprachauswahl, damit `\selectlanguage` den Code auflösen kann.

```

579   \bool_lazy_and:nnT{\ols_bcptags}{\cs_if_exist_p:N\babeladjust}{

```

`\bbl@bcp@map@⟨Code⟩` wird für jede Auswahlssprache mit ermitteltem Namen (oben) vorbelegt, bevor `autoload.bcp47` aktiviert wird. Dessen eigene Tag-Auflösung prüft `\bbl@bcp@map@⟨Code⟩` zuerst und nutzt es unverändert, wenn vorhanden (so hängt die Sprachauswahl nicht von Babels eigener, je nach Release unterschiedlich vollständiger BCP-47-Datenbank ab). Für Sprachen, die nicht ermittelt werden konnten, bleibt `autoload.bcp47s` eigenes Verhalten unverändert.

```

580     \clist_map_inline:Nn \ols_langs {
581       \cs_if_exist:cT{ olslangmap / \tl_to_str:n {#1} }{
582         \cs_gset:cpe { bbl@bcp@map@ #1 } { \ols_use_map:n {#1} }
583       }
584     }
585     \babeladjust{ bcp47.toname = on , autoload.bcp47 = on }
586   }
587   \bool_if:NT\ols_shorthands {

```

Die Shorthand-Befehle benötigen ein aktives @.

```

588     \makeatletter

```

Die Babel-Shorthand-Infrastruktur ist bei Babel bereits vorhanden, bei Polyglossia muss sie explizit initialisiert und ohne Sprachpaket direkt geladen werden.

```

589     \@ifpackageloaded{babel}{\}%
590     \@ifpackageloaded{polyglossia}
591       {\InitializeBabelShorthands}
592       {\input{babelsh.def}}{\}%
593     }
594     \@ifpackageloaded{babel}

```

```

595     {\usesorthands*{"}}
596     {\usesorthands{"}}
597     \defineshorthand{"`"}{\openautoquote}
598     \defineshorthand{"'"}{\closeautoquote}
599     \defineshorthand{"{/"}{\textormath{\bbl@allowhyphens
600         \discretionary{/}{/}{/}\bbl@allowhyphens}{}}
601     \makeatother
602 }
603 }

```

Die Sprachauswahl erfolgt im `begindocument`-Hook, also *nach* der Aktivierung der Hauptsprache durch das Sprachpaket.

```

604 \AddToHook{begindocument}{%
605     \bool_if:NF\ols_target_from_langpkg{
606         \cs_if_exist:cT{ver@babel.sty}{
607             \selectlanguage{\ols_mapped_target_lang}
608         }
609         \cs_if_exist:cT{ver@polyglossia.sty}{
610             \setdefaultlanguage{\ols_mapped_target_lang}
611         }
612     }
613 }

```

Index

A	
<code>auto</code>	5 f.
B	
<code>babel</code> (Paket)	15 f., 18
<code>bcp47</code>	6, 9, 27
<code>bicaption</code> (Paket)	3
C	
<code>\closeautoquote</code>	10
<code>comment</code> (Paket)	3
<code>csquotes</code> (Paket)	6, 10, 28
CTAN	3, 5, 10
D	
<code>\DeclarePresInitialBranchingCommand</code>	20
<code>\DocumentMetadata</code>	2, 5, 17, 19
E	
EIJKHOUT, Victor	3
<code>\enquote</code>	10
F	
<code>fontenc</code> (Paket)	11
<code>fontspec</code> (Paket)	11
<code>\foreignlanguage</code>	4, 9
G	
GREWE, Richard	3
H	
<code>hyperref</code> (Paket)	8, 20
I	
<code>\include</code>	2
L	
<code>langselect</code> (Paket)	8, 10
<code>languages</code>	5, 14
<code>load babel</code>	6, 9, 26 f.
<code>load polyglossia</code>	6, 9, 27
M	
<code>main</code>	9
<code>\makeatletter</code>	10
<code>map</code>	6, 9, 26
<code>multilang</code> (Paket)	3
O	
<code>\ols_babel_name:nN</code>	14
<code>\ols_babel_name_from_ini:nN</code>	12 f.
<code>\ols_babel_name_from_polyglossia:nN</code>	13
<code>\ols_collect_replace</code>	26
<code>\ols_mapped_target_lang</code>	19, 27
<code>\olsIsoTags</code>	11
<code>\olsMakeBilingualMacro</code>	8, 21
<code>\olsMakeTrilingualMacro</code>	8, 21
<code>\olsTargetLanguage</code>	2, 4 f., 9, 14, 17
<code>\openautoquote</code>	10
<code>osglecture</code> (Bundle)	10
P	
<code>polyglossia</code> (Paket)	15 f., 18
<code>prefix</code>	5, 8
<code>presitemize</code> (Umgebung)	20
S	
<code>\selectlanguage</code>	4, 6, 27 f.
<code>\setdefaultlanguage</code>	27
T	
<code>targetlang</code>	4 f., 9, 14, 17
<code>translations</code> (Paket)	3
<code>translator</code> (Paket)	3
<code>trim</code>	6
U	
<code>unified shorthands</code>	6, 10
<code>\usepackage</code>	2, 4
V	
<code>\varsfromjobname</code>	11
<code>varsfromjobname</code> (Paket)	5
X	
<code>xtcpts</code> (Paket)	3

Index

A	
<code>auto</code>	5 f.
B	
<code>babel</code> (Paket)	15 f., 18
<code>bcp47</code>	6, 9, 27
<code>bicaption</code> (Paket)	3
C	
<code>\closeautoquote</code>	10
<code>comment</code> (Paket)	3
<code>csquotes</code> (Paket)	6, 10, 28
CTAN	3, 5, 10
D	
<code>\DeclarePresInitialBranchingCommand</code>	20
<code>\DocumentMetadata</code>	2, 5, 17, 19
E	
EIJKHOUT, Victor	3
<code>\enquote</code>	10
F	
<code>fontenc</code> (Paket)	11
<code>fontspec</code> (Paket)	11
<code>\foreignlanguage</code>	4, 9
G	
GREWE, Richard	3
H	
<code>hyperref</code> (Paket)	8, 20
I	
<code>\include</code>	2
L	
<code>langselect</code> (Paket)	8, 10
<code>languages</code>	5, 14
<code>load babel</code>	6, 9, 26 f.
<code>load polyglossia</code>	6, 9, 27
M	
<code>main</code>	9
<code>\makeatletter</code>	10
<code>map</code>	6, 9, 26
<code>multilang</code> (Paket)	3
O	
<code>\ols_babel_name:nN</code>	14
<code>\ols_babel_name_from_ini:nN</code>	12 f.
<code>\ols_babel_name_from_polyglossia:nN</code>	13
<code>\ols_collect_replace</code>	26
<code>\ols_mapped_target_lang</code>	19, 27
<code>\olsIsoTags</code>	11
<code>\olsMakeBilingualMacro</code>	8, 21
<code>\olsMakeTrilingualMacro</code>	8, 21
<code>\olsTargetLanguage</code>	2, 4 f., 9, 14, 17
<code>\openautoquote</code>	10
<code>osglecture</code> (Bundle)	10
P	
<code>polyglossia</code> (Paket)	15 f., 18
<code>prefix</code>	5, 8
<code>presitemize</code> (Umgebung)	20
S	
<code>\selectlanguage</code>	4, 6, 27 f.
<code>\setdefaultlanguage</code>	27
T	
<code>targetlang</code>	4 f., 9, 14, 17
<code>translations</code> (Paket)	3
<code>translator</code> (Paket)	3
<code>trim</code>	6
U	
<code>unified shorthands</code>	6, 10
<code>\usepackage</code>	2, 4
V	
<code>\varsfromjobname</code>	11
<code>varsfromjobname</code> (Paket)	5
X	
<code>xtcpts</code> (Paket)	3