# The sTeX4 Package Collection*

Michael Kohlhase, Dennis Müller
FAU Erlangen-Nürnberg
http://kwarc.info/

2026-06-25

## Contents



*Version 4.0.1 (last revised 2026-06-25)

sTEX is – by now – relatively stable and ready to use for the general public. However, it is also actively being developed further and subject to ongoing research. Some of the features described in here might not fully work as expected, some are still experimental, there might occasionally be cryptic error messages, and in general bugs are expected.
We welcome all kinds of issues you might encounter at `https://github.com/slatex/sTeX`.

*If you have questions or problems with sTEX, you can talk to us directly at* `https://matrix.to/#/#stex:fau.de`.

# Chapter 1

# Introduction & Setup

## 1.1 What is sTeX?

sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

At its core is the sTeX package for LaTeX, that allows for semantically marking up document fragments; in particular concepts, formulae and mathematical statements (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated documents formats them into normal-looking PDF.

But sTeX also comes with a conversion pipeline into semantically annotated HTML (FTML), which can host semantic added-value services that make the documents *active* (i.e. interactive and user-adaptive) – essentially turning LaTeX into a document format for (mathematical) knowledge management (MKM).

The sTeX system consists of the following components:

- The sTeX package,
- the RuSTeX system for converting LaTeX documents to (semantically enriched) FTML,
- the FLAMS system; a semantically informed back-end for managing and hosting the FTML with integrated added-value services,
- a collection of math archives of sTeX document fragments and symbols for reuse, available of `mathhub.info`, and
- the FLAMS IDE: A VS Code extension that, besides the usual IDE functionalities like syntax highlighting, integrates FLAMS and allows for accessing the public math archives on `mathhub.info` to make the entire toolchain easily accessible to authors.

If PDF is all you are interested in, the sTeX package is all you *need*. Either way, however, we recommend using the sTeX IDE – it very much helps with semantically annotating your LaTeX documents.

## 1.2 The sTeX package

The sTeX package extends LaTeX with:

- A mechanism to declare symbols – concepts, functions, relations, variables, etc., which can be used and referenced in text or via semantic macros in mathematical formulae,
- a module system based on logical identifiers – modules bundle declarations, definitions, theorems, document snippets, and symbols for reuse, and
- an analogous organizational structure for developing documents modularly from individual fragments and sections.

The sTeX package has been designed to have minimal impact on other packages and document classes, or the actual document layout – formatting of semantic environments, symbol references and semantic macros can be fully customized.

## 1.3 Math archives and the MathHub Directory

To make the most of sTeX, it is strongly encouraged to follow a workflow of small document fragments and modules to maximize reuse.

One considerable weakness of LaTeX is the way source files are referenced: they need to be either in a `texmf` directory, or else be referenced via file paths relative to the main `.tex`-file being compiled. This is highly inconvenient if we want to collaboratively develop many highly interrelated document fragments.

sTeX therefore adds an organizational layer on top of LaTeX's: **math archives** stored in a fixed `MathHub` directory anywhere on your hard drive. Referencing source files and modules is then done relative to the containing math archive, and is thus *independent* of user's individual setups or the current `.tex`-file.

The drawback of this approach is that sTeX needs to know the location of your `MathHub` directory. There are multiple ways to achieve that, but the simplest and recommended approach is to set an environment variable: Simply create a new directory `<path>/MathHub` somewhere on your hard drive and set the environment variable `MATHHUB` as the path to this new directory.

Alternatively, you can let the FLAMS IDE do the work for you (see section 1.4).

For more on math archives, see Section 1 (Math Archives and the `MathHub` Directory) in the sTeX Manual.

## 1.4 Setting Up the FLAMS IDE

sTeX is based on LaTeX, and adds additional layers of presentational and functional markup to it. As a consequence the source files of sTeX documents look quite different from the resulting FTML and PDF documents. Thus the best way of interacting the sTeX document collections is via an integrated development environment (IDE). In this tutorial we will use the FLAMS plugin for the VS Code, which you should set up as a first step (this also sets up the necessary auxiliary software).

Setting up sTeX with the dedicated IDE is easy:

1. Download and install VS Code here: `https://code.visualstudio.com/download`

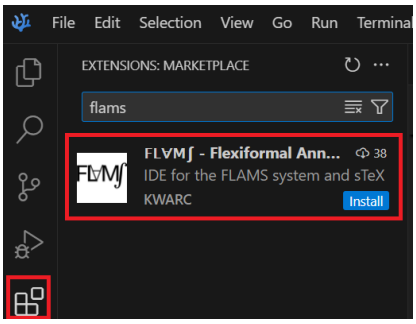



Figure 1: Installing the FLAMS IDE

2. Start VS Code and navigate to the *Extensions*-tab on the left. Here you can search for Extensions in the VS Code marketplace. Look for the FLAMS extension by *KWARC*, as in Figure 1.

3. Having done so, upon opening any folder in VS Code containing a `.tex`-file, you will be prompted to either download a new FLAMS or point the extension to an existing FLAMS executable. In the vast majority of cases, you'll want to download and choose a directory to install FLAMS to.

# Part I
# Tutorial

*The dynamic HTML version of this part can be found at*
`https://mathhub.info/?a=sTeX/Documentation&d=tutorial&l=en`

sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

In this part, we will give a broad but shallow introduction to sTeX, and what you can get out of it. Additionally, this serves as an introduction to the FLAMS IDE, and we consequently assume that you have that one set up, as described in section 1.4.

Note that in PDFs, the specific highlighting of semantically annotated text is fully customizable (see chapter 9). In this document, we use this highlighting for notation components, this highlighting for symbol references, this highlighting for (local) variables and **this highlighting** for definienda; i.e. new concepts being introduced.

# Chapter 2

# The Basics

This document itself uses sTeX and serves as a direct example for the following. You can download its source files, the generated PDF files, and the generated HTML documents directly from within the IDE, by navigating to the FLAMS tab in the menu on the left and finding `sTeX/Documentation` in the list of math archives and clicking the small "Install"-button next to it, see the screenshot on the left of Figure 2.

Once downloading is finished (this may take a while since dependencies are also downloaded), you can then browse the `.tex`-files in `sTeX/Documentation` directly from the math archives panel in the sTeX tab, as you can see in the right screenshot in Figure 2.

For example, you can now navigate to the file `tutorial/intro.en` to see the sources of this very chapter.

As a first example, consider the following document fragment from section 1.1:

> sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

If you were to look at the generated HTML from this fragment, you could hover over the highlighted words (sTeX, PDF, HTML, OMDoc) and get a little popup with their definitions (Figure 3). Neat, huh?

Here, in the PDF, hovering will only show you a unique identifier (MMT-URI) for the word, and link to a definition on the web. Still useful, but not quite as neat, of course.

A plain LaTeX-version of the above document fragment, without any sTeX markup, could look like this:

**Example 1**
Input:

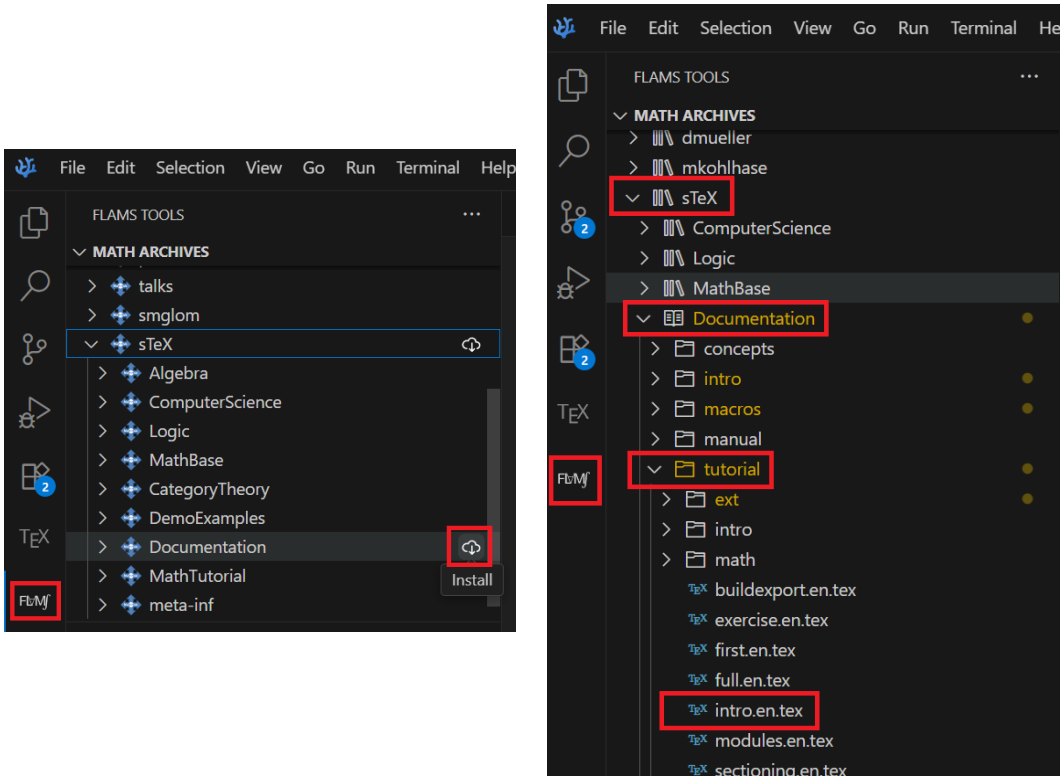



Figure 2: Installing Math Archives

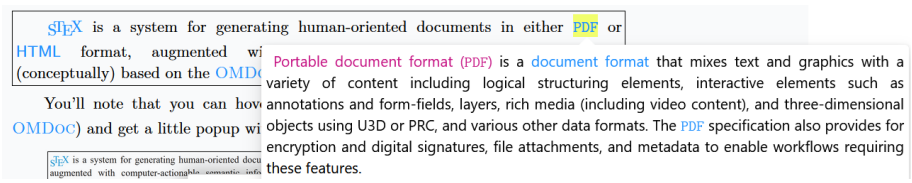



Figure 3: Definition on Hover

File `tutorial/intro/intro1plain.en.tex`

```
\documentclass{article}
\usepackage{stex-logo}
\begin{document}

  \sTeX{} is a system for generating human-oriented documents
  in either \textsf{PDF} or \textsf{HTML} format, augmented
  with computer-actionable semantic information (conceptually)
  based on the \textsc{OMDoc} format and ontology.

\end{document}
```

Output:

sTEX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

(Examples like the one above always show the file the source code is in, so if you have downloaded the `sTeX/Documentation` math archive you can toy around with it yourself)

If you save a file in the IDE (regardless of whether it has unsaved changes), a preview window will pop up, showing you the HTML generated from the `.tex`-file; see (Figure 4).

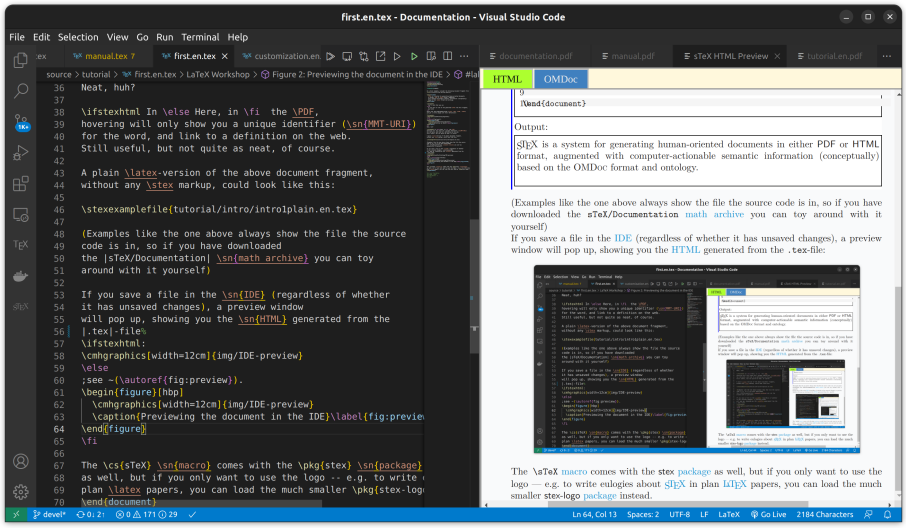



Figure 4: Previewing the document in the IDE

The `\sTeX` macro comes with the stex package as well, but if you only want to use the logo – e.g. to write eulogies about sTEX in plain LATEX papers, you can load the much smaller stex-logo package instead.

## 2.1 Text symbols

The most central concept behind sTeX is that of a *symbol*:

> A **symbol** is a *named* concept that can be defined, documented and referenced. Examples for symbols are mathematical constants, functions, theorems, statements, principles – anything that has a (somewhat) precise meaning and can be referenced by name can be a symbol.

Before we explain how we can declare new symbols and associate them with definitions, notations and all that, let's assume an ideal world in which others have done that job already for us – after all, sTeX is all about *reuse*, and naturally, there are sTeX symbols for all of the above already. Let's start with the one for sTeX itself:

### 2.1.1 Using Modules & Search in the IDE

In the VS Code IDE, navigate to the FLAMS-tab on the left. In the search panel, select the "`Symbols`" radio button and search for "`sTeX`". The third search result should be what we're looking for (Figure 5).

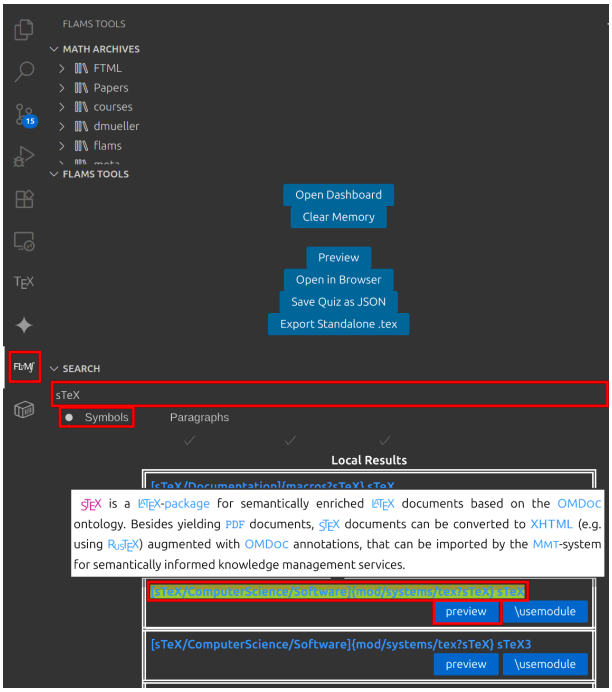



Figure 5: Search in the sTeX IDE

Search results are grouped into *local* and *remote* results. Local ones are the ones you already have in your local `MathHub` directory; remote ones you can download directly from within the IDE.

You can click the preview button to see the generated FTML for the document – the resulting window that pops up also has an OMDoc button you can click, which (among other things) shows you the semantic macros provided by the respective module: In this case, it tells us that there is a *text symbol* named "sTeX" with semantic macro `\stex` in the module `sTeX` in the same document. It produces the presentation "sTEX" as we want (Figure 6).

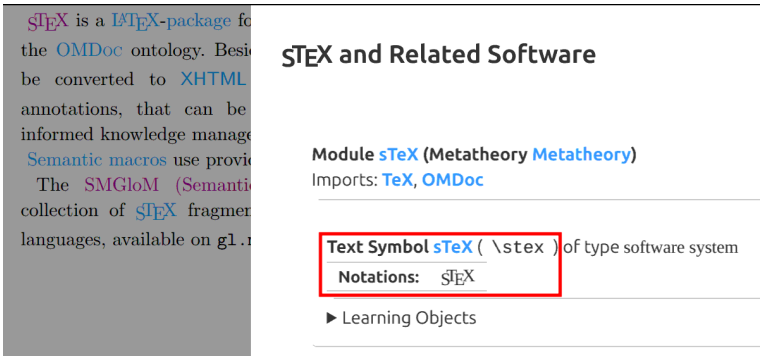



Figure 6: OMDoc Preview

> sTEX A **text symbol** is a symbol foo with an associated semantic macro `\foo`. The macro `\foo` is allowed in text or math mode and produces a predefined piece of text output annotated with foo.
> The variant `\fooname` produces the same output without annotation.

If we want to use the sTEX symbol in a document – which we have open in the IDE – we simply click on the `\usemodule` button, and the IDE will automatically insert the line
`\usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}`,
making all symbols in that module available to use – in particular, we can now use the `\stex` semantic macro instead of the plain, non-semantic `\sTeX` macro – that is, of course, after we include the `stex` package first.

> sTEX The `\usemodule` macro takes as *optional* argument the name of a math archive, and as a regular argument the path to an sTEX module (see section 7.5).

Analogously, we can also search for the PDF, HTML and OMDoc symbols, all of which are also text symbols and have the associated semantic macros `\PDF`, `\HTML` and `\omdoc`; the document should thus look like this:

**Example 2**
Input:

File `tutorial/intro/intro1stex.en.tex`

```
\documentclass{article}
\usepackage{stex}
\begin{document}
  \usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}
  \usemodule[sTeX/ComputerScience/Software]{mod/formats?PDF}
  \usemodule[sTeX/ComputerScience/Software]{mod/formats?HTML}
  \usemodule[sTeX/ComputerScience/Software]{mod/formats?OMDoc}

  \stex is a system for generating human-oriented documents
  in either \PDF or \HTML format, augmented
  with computer-actionable semantic information (conceptually)
  based on the \omdoc format and ontology.
\end{document}
```

Output:

sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

Now, our generated HTML looks a lot more interesting, with highlighting, pop-ups on hover and all that. Notably however, if we compile the file with `pdflatex`, it looks pretty much exactly as before.

That's because we haven't told sTeX what to do with semantic annotations yet – and by default, it does not do anything fancy, except for wrapping them in an `\emph`. We can customize how we want sTeX to highlight various semantic text fragments (see chapter 9). A default highlighting schema is provided in the stex-highlighting package – including that will

- highlight semantically annotated text in this color,
- show the MMT-URI of the corresponding symbol in a tooltip on hovering over the text,
- make the text link to the place the symbol is being defined in the current document (if it is), or, alternatively,
- make it link to an external resource, if one is known. In our case, they link to `stexmmt.mathhub.info/:sTeX`, where the HTML for all the symbols we use in this document are hosted.

## 2.2 Symbol References

Let's continue with the next paragraph of section 1.1; for now unannotated:

**Example 3**
Input:

File `tutorial/intro/intro2plain.en.tex`

```
\documentclass{article}
\usepackage{stex}
\begin{document}

  At its core is the \sTeX{} package for \LaTeX, that allows for
  semantically marking up document fragments; in particular
  concepts, formulae and mathematical statements (such as
  definitions, theorems and proofs). Running \texttt{pdflatex}
  over \sTeX-annotated documents formats them into normal-looking
  \textsf{PDF}.

\end{document}
```

Output:

At its core is the sTeX package for LaTeX, that allows for semantically marking up document fragments; in particular concepts, formulae and mathematical statements (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated documents formats them into normal-looking PDF.

We already know how to annotate "sTeX" and "PDF"; and if we use the search field in the IDE again, we can also find a text symbol for "LaTeX". But if we look at the documentation, we will note that *more* is highlighted:

At its core is the sTeX package for LaTeX, that allows for semantically marking up document fragments; in particular concepts, formulae and mathematical statements (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated documents formats them into normal-looking PDF.

The "`package`"-symbol can be found in the LaTeX module too, and searching for the keywords "formula" and "mathematics" will yield the symbols "`well-formed formula`" and "`mathematics`", but they are not *text symbols* and "`mathematics`" and "`package`" do not even have a semantic macro – and the one for "`well-formed formula`" would not work outside of math mode.

Text symbols are special in that way – they are intended for symbols that have a specific formatting associated (such as LaTeX, OMDoc, or HTML, which we prefer to typeset as sans serif). For those settings, it makes sense to associate that formatting with a semantic macro that does the typesetting for us.

Symbols *without* a text macro can be referenced with the `\symname` macro: `\symname{package}` prints the *name* of the "`package`"-symbol and annotates it accordingly, without any special formatting – in particular it is compatible with being in `\emph`, `\textbf` and similar macros. That takes care of *one* of the missing annotations.

More generally, the `\symref` macro can be used to annotate arbitrary text with a symbol: `\symref{mathematics}{mathematical}` associates the text `mathematical` with the symbol "`mathematics`"; thus, we get "mathematical" and similarly "formulae".

In general, any macro that expects a symbol name can be given either

1. the *name* of the symbol,
2. the name of its semantic macro,
3. or any suffix of its MMT-URI containing at least the module name.

The second option is often short – and therefore convenient to write; for example, to achieve "formulae", we can also write `\symref{wff}{formulae}`, since `\wff` is the semantic macro for "well-formed formula".
The third option allows for distinguishing between multiple symbols with the same name – the IDE can help in the latter case, by underlining ambiguous symbol references in yellow, and offering the `Quick Fix` functionality to let you select and autocomplete the specific symbol you want to reference.

Since `\symname` and `\symref` are a lot to type for something that should ideally be used as often as possible, the macros `\sn` and `\sr` exist as well and behave exactly the same way. We also provide some convenience abbreviations for `\sn`; namely `\Sn` (capitalizes the first letter of the symbol name), `\sns` (adds an "s" at the end, for the most common pluralization of a name), and `\Sns` (both).

Using all of the above, our annotated fragment now looks like this:

**Example 4**

Input:

File `tutorial/intro/intro2stex.en.tex`

```
\usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}
\usemodule[sTeX/Logic/General]{mod/syntax?Formula}
\usemodule[sTeX/MathBase/General]{mod?Mathematics}
\usemodule[sTeX/ComputerScience/Software]{mod/formats?PDF}

 At its core is the \stex \sn{package} for \latex, that allows for
 semantically marking up document fragments; in particular
 concepts, \sr{wff}{formulae} and \sr{mathematics}{mathematical}
 statements (such as definitions, theorems and proofs). Running
 \texttt{pdflatex} over \stex-annotated documents formats them
 into normal-looking \PDF.
```

Output:

At its core is the sTeX package for LaTeX, that allows for semantically marking up document fragments; in particular concepts, formulae and mathematical statements (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated documents formats them into normal-looking PDF.

There's only one problem: *the document does not compile*, with an error `Undefined control sequence`. The reason being that *some* macro in the module `Formula` uses the `\text` macro. We can fix that by using the amsfonts package of course, but this points to a more general problem; namely that modules can make use of various LaTeX packages for typesetting symbols.

Good practice suggests putting those packages into a *prelude* per math archive, which we can then import from anywhere, using the `\libinput` macro. For more on that, see

section 5.3.

For now, suffice it to say that we can import all packages required for the module `Formula` from the math archive `sTeX/Logic/General` by adding the line

```
\libinput[sTeX/Logic/General]{preamble}
```

before the `\begin{document}`.

## 2.3 Modules and Simple Symbol Declarations

Consider again the first two paragraphs of section 1.1:

> sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.
>
> At its core is the sTeX package for LaTeX, that allows for semantically marking up document fragments; in particular concepts, formulae and mathematical statements (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated documents formats them into normal-looking PDF.

Firstly, note that the first paragraph would be perfectly suitable to serve as a pop-up definition on hover for the sTeX symbol. Secondly, what if all the symbols used in the above *didn't* already exist?

In this section, we will describe how to make your own symbols and collect them as reusable fragments in modules and math archives from scratch.

We start by creating a new math archive. In the IDE, switch to the sTeX-tab on the left and click the "`New sTeX Archive`" button (Figure 7). You will then be asked for

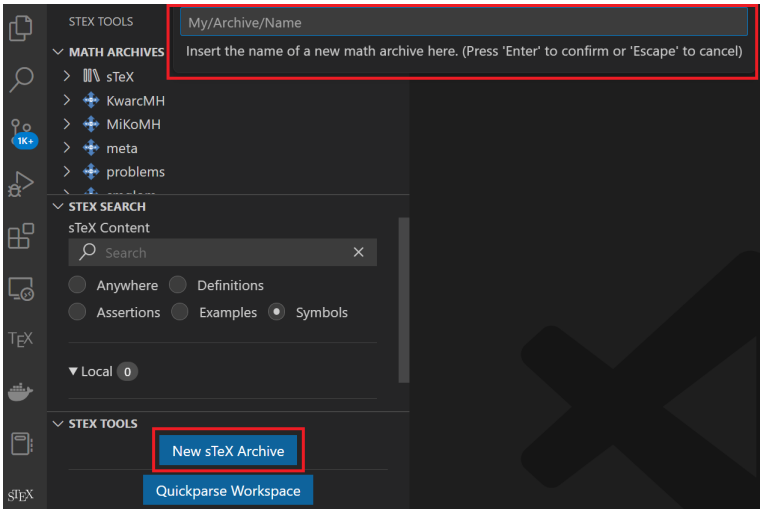



Figure 7: New Math Archive in the IDE

the name of the archive, and a url-base, where the content is supposedly going to end up

online. You can safely keep the defaults for the latter. In the following, we assume that your archive is named `my/archive`.

The IDE will then create the following files and directories in your MathHub directory:

```
- my
  - archive
    - lib
      - preamble.tex
    - META-INF
      - MANIFEST.MF
    - source
      - helloworld.tex
```

. . . and open the file `helloworld.tex` with the content

```
1 \documentclass{stex}
2 \libinput{preamble}
3 \begin{document}
4 % A first sTeX document
5 \end{document}
```

You can now reference any newly created content in you new archive using for example `\usemodule[my/archive]{...}`.

Let's start with the "LaTeX" symbol. Rename the file `helloworld.tex` to something more meaningful, for example `latex.en.tex` – the `.en` will be picked up on by sTeX to signify that the fragment will be in *english* (see subsection 7.1.1).

What we want to achieve in this file is the following:

> **TeX** is a document typesetting software developed by Donald Knuth, with a focus on mathematical formulae. It is based on a powerful and extensible **macro** expansion engine.
>
> **LaTeX** is a (nowadays) default collection of TeX macros developed by Leslie Lamport. Among other things, LaTeX introduces **environments**, a distinction between preamble and document content, **packages** to bundle and distribute macro definitions, and **document classes**: special packages that govern the global layout of a document.

In particular, in the HTML the two paragraphs above should be shown when hovering over the symbols they define (as indicated by the magenta definiendum highlighting). So we need symbols and semantic macros, for: TeX, macro, LaTeX, environment, package and document class.

Symbol declarations are only allowed within modules:

> sTeX A **module** is a *named* block that bundles symbol declarations for subsequent reuse. A module is introduced with the `smodule`-environment.

Let's name our module `LaTeX`. We then wrap the contents of our document in a `smodule` environment:

```
\begin{document}
  \begin{smodule}{LaTeX}
    ...
  \end{smodule}
\end{document}
```

FLAMS Note, that the IDE immediately picks up on this and displays the full FTML URI of our new module over the `\begin{smodule}{LaTeX}` (Figure 8) –


Figure 8: VS Code URI on Hover

From this, we can glimpse that the namespace of the module is `http://mathhub.info?a=my/archive&p=latex&m=LaTeX`. This implies, that to use the module somewhere else, we will have to type `\usemodule[my/archive]{latex?LaTeX}` – the `latex`-part pointing to the *file* and `LaTeX` referring to the actual module.

If we rename the file to `LaTeX.en.tex`, we notice that the namespace changes to `http://mathhub.info?a=my/archive&m=LaTeX`, allowing us to now use it with `\usemodule[my/archive]{LaTeX}` directly. That's because the module name `LaTeX` and the file name `LaTeX` match now (see section 7.5, Figure 9).


Figure 9: VS Code URI on Hover

Note that "`LaTeX`" and "`latex`" only differ in capitalization – if your file system is case-insensitive (as e.g. MacOS's was until quite recently), this distinction gets murky, but remains very important especially if you want to share your math archive with others!
It is therefore *highly recommended* to treat file names as case-sensitive either way.

Within the module, we can now declare new symbols using the `\symdecl`-macro. We start with those that are not text symbols:

```
\symdecl*{macro}
\symdecl*{environment}
\symdecl*{package}
\symdecl*{document class}
```

The `*` after the `\symdecl` indicates, that we do not want a semantic macro for the symbol – otherwise, it would generate one with the same name as the symbol itself and "pollute the macro space", so to speak.

The symbols TEX and LATEX, however, have a definite way of being typeset associated with them, which can be produced using the standard `\TeX` and `\LaTeX` macros. So let's make them text symbols, using the `\textsymdecl` macro:

```
\textsymdecl{tex}{\TeX}
\textsymdecl{latex}{\LaTeX}
```

The first argument being the name of the generated macro (i.e. `\tex` and `\latex`) and the second one specifying the output to produce.

As we cannot rely on the underlying TEX engine treating UniCode charaters natively, symbol names can only consist of printable ASCII characters.

## 2.4 Documenting Symbols

We can now use the two new macros, `\symname`/`\sn`, `\symref`/`\sr` etc. to mark up the above two paragraphs. However, the system does not yet know what to show a reader when hovering over the symbol in the HTML. We can fix that by using the `sdefinition` or `sparagraph` environments.

Ignoring the former for now, which is more useful for mathematical concepts, we can use the following to mark up the first paragraph:

```
\begin{sparagraph}[style=symdoc,for={tex,macro}]
  \tex is a document typesetting
  software developed by Donald Knuth, with a focus on
  mathematical formulae. It is based on a powerful
  and extensible \sn{macro} expansion engine.
\end{sparagraph}
```

In general, the `sparagraph` environment can be used to mark up arbitrary paragraphs semantically, but the `style=symdoc` option tells STEX to use this paragraph as a documentation for the symbols provided in the `for=` option.

We just used the semantic macro `\stex` and the `\sn` macro to mark up the fragment – but we can do better. Both concepts are being *introduced* in the above paragraph, and we can let STEX know that that is the case:

Within an `sparagraph` environment with `style=symdoc` (or an `sdefinition` environment), we can mark up *definienda*, meaning the terms *being defined*, explicitly. Analogously to `\symname` and `\symref`, we have the macros `\definame` and `\definiendum` for that purpose.

Note that the `\tex` macro induced by the text symbol above already marks up the "TEX" it produces, so wrapping it in another `\definiendum` would be redundant. However, every text symbol also generates a *second* macro with the suffix `name` that generates a non-marked-up version of the same presentation. In other words, we get the macro `\texname` for free, that produces "TEX" (of course, we could just as well use the `\TeX` macro, but that one you probably know already).

Furthermore, every `\definiendum` or `\definame` automatically adds the symbol being referenced to the internal `for=`-list of the `sparagraph` environment, obviating the need to list it explicitly.

As such, we can produce a better markup like this:

```
\begin{sparagraph}[style=symdoc]
  \definiendum{tex}{\texname} is a document typesetting
  software developed by Donald Knuth, with a focus on
  mathematical formulae. It is based on a powerful
  and extensible \definame{macro} expansion engine.
\end{sparagraph}
```

**Exercise**
In your archive `my/archive`, create additional files that produce the following outputs:

Mathematics.en.tex

To do **mathematics** is to be, at once, touched by fire and bound by reason. This is no contradiction. Logic forms a narrow channel through which intuition flows with vastly augmented force.

– Jordan Ellenberg

PDF.en.tex

**Portable Document Format (PDF)** is a document format that mixes text and graphics with a variety of content.

HTML.en.tex

The **HyperText Markup Language (HTML)** is a representation format for web-pages.

OMDoc.en.tex

**OMDoc** is a document format for representing mathematical documents with their flexiformal semantics.

such that the following file compiles and shows the above snippets on hover:

sTeX.en.tex

```
\documentclass{stex}
\libinput{preamble}
\begin{document}
\begin{smodule}{sTeX}
  \usemodule{OMDoc}
  \usemodule{PDF}
  \usemodule{HTML}
  \textsymdecl{stex}{\sTeX}
  \begin{sparagraph}[style=symdoc]
    \definiendum{stex}{\stexname} is a system for generating
    documents in either \PDF or \HTML format, augmented with
    computer-actionable semantic information (conceptually)
    based on the \OMDoc format and ontology.
  \end{sparagraph}
\end{smodule}
\end{document}
```

**sTeX** is a system for generating documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

The preamble of every file should only be

```
\documentclass{stex}
\libinput{preamble}
```

and the macros \OMDoc, \PDF, \HTML should produce \textsc{OMDoc}, \textsf{PDF} and \textsf{HTML}, respectively (but with semantic annotations of course).

---

*Lösung:* Can be found in [sTeX/Documentation]source/tutorial/solution

---

## 2.5 Sectioning and Reusing Document Fragments

We know now how to import and reuse the symbols of some module (using \usemodule). What about the actual document *content*?

Assume we want to write a new article that includes all of the fragments in my/archive we made so far, in a file all.en.tex in the same math archive:

```
1  \documentclass{article}
2  \usepackage{stex}
3  \libinput{preamble}
4  \begin{document}
5    \author{Me}
6    \title{The \texttt{my/archive} Archive}
7    \maketitle
8    \tableofcontents
9    ...
10 \end{document}
```

In there, we want sections as follows:

```
- 1 Preliminaries
  (Mathematics)
  - 1.1 Document Formats
    (PDF)
    (HTML)
    (OMDoc)
- 2 \TeX and Friends
  (LaTeX)
  (sTeX)
```

We could of course do the following:

```
\section{Preliminaries}
  \input{Mathematics.en}
  \subsection{Document Formats}
    \input{PDF.en}
    \input{HTML.en}
    \input{OMDoc.en}
\section{\TeX and Friends}
  \input{LaTeX.en}
  \input{sTeX.en}
```

...but this approach has two drawbacks:

Firstly, we need to manually keep track of the section levels, by explicitly writing `\section`, `\subsection` etc. This is fine as long as we are just interested in this particular article. But what if we want to *reuse* the article's content in another document at some point? The section levels might be entirely different then – e.g. we might want the "Preliminaries" section to be a subsection instead.

Secondly, the `\input` macro considers the file name/path provided to be either *absolute* or relative to the *current* `tex` *file being compiled* – which means that the `\input{Mathematics.en}` only works for files in the same directory as `Mathematics.en.tex`.

In short: using `\section`, `\chapter` etc. explicitly, and `\input` to reuse fragments, breaks reusability.

Instead of using `\section` and `\subsection`, sTeX therefore provides the `sfragment` environment.

`\begin{sfragment}{Foo}...\end{sfragment}` inserts a sectioning header depending on the current section level and availability. These are: `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph` and `\subparagraph`. This allows us to do the following instead:

```
\begin{sfragment}{Preliminaries}
  \input{Mathematics.en}
  \begin{sfragment}{Document Formats}
    \input{PDF.en}
    \input{HTML.en}
    \input{OMDoc.en}
  \end{sfragment}
\end{sfragment}
\begin{sfragment}{\TeX and Friends}
  \input{LaTeX.en}
  \input{sTeX.en}
\end{sfragment}
```

The only problem remaining now is that if we do this, sTeX will insert a `\part` for the first `sfragment`. If we want the "top-level" sectioning level to be `\section` instead, we can insert a `\setsectionlevel{section}` in the preamble.

As a more reuse-friendly replacement of `\input`, sTeX provides the `\inputref` macro. Using that has two advantages: Firstly, its argument is relative to some (optionally provided, or the current) math archive and is thus independent of the specific location of the file relative to the currently being compiled `.tex`-file. Secondly, when converting to HTML, it will *not* "copy" the referenced file's content in its entirety (as `\input` would), but instead dynamically insert the already existent (if so) HTML of the referenced file, avoiding content duplication and having to process the file all over again.

In general `\inputref[some/archive]{file/path}` inputs the file `file/path.tex` in the archive `some/archive`. As the `\input`-ed files in the example above are in the same archive anyway, we can simply substitute the `\input`s by `\inputref`s and call it a day.

Finally, we can make two more minor changes:

1. The *title* of our document is only supposed to be there, if we compile the document directly – if we were to `\inputref` our file into a "driver file" `all.en.tex`, the title and the table of contents should be omitted.

We can achieve this using the \ifinputref conditional: by wrapping the header in an \ifinputref \else...\fi, it will only be processed if the file is *not* being loaded using \inputref. \ifinputref is a "classic" TEX conditional and is treated as such in both PDF and HTML compilation. A smarter macro to use is \IfInputref, which takes two arguments for the *true* and *false* cases, respectively. Additionally, when compiling to HTML, *both* arguments to \IfInputref will be processed, and the back-end will decide which of the two to present when serving a document.

2. The table of contents should also be omitted in HTML mode. To achieve that, we can use the \ifstexhtml conditional, which is *true* if the document is being compiled to HTML, and *false* if compiled to PDF.

Note, that since *both* arguments of \IfInputref are processed, they should *not* open TEX groups or environments!

In summary, we can modify our document to do the following:

```
\IfInputref{}{
  \author{Me}
  \title{The \texttt{my/archive} Archive}
  \maketitle
  \ifstexhtml \else \tableofcontents \fi
}
```

The final `all.en.tex` can be found in `[sTeX/Documentation]tutorial/solution/all.en.tex`.

## 2.6 Building and Exporting HTML

So far we know how to write STEX documents, (we assume) how to build PDF files from them (via `pdflatex` of course), and on saving documents the IDE will preview the generated HTML. But if we do that with our new `all.en.tex`, we get presented with Figure 10 Where did all of our fragments go?


Figure 10: Missing Fragments in the HTML Preview

Well, they don't exist yet as HTML. The HTML Preview window in the IDE is really just that: A *preview*. But when using \inputref, it has to find the HTML of the \inputrefed fragment *somewhere*. Meaning: we have to compile all of the fragments we used to HTML first. Individually, we can compile the currently open file and all its dependencies in VS Code using the button in Figure 11.

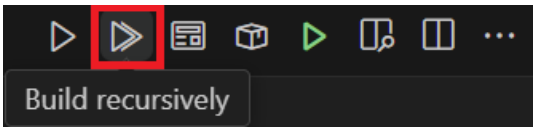



Figure 11: The `Build PDF/XHTML/OMDoc` Button

This will queue all dependencies of the file in a new build queue and open the FLAMS dashboard for us (see Figure 12). Clicking on the `run` button in the dashboard will then

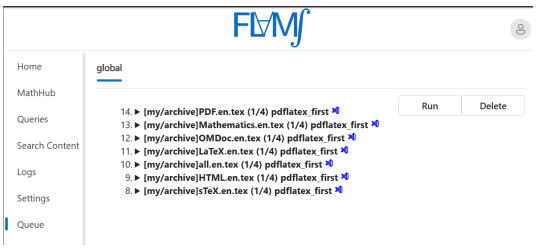



Figure 12: The FLAMS build queue

for every queued file:

1. Run `pdflatex` twice,
2. Convert the file to HTML,
3. Extract all the semantics, and
4. Construct a search index.

Once that's done, saving `all.en.tex` again yields the correct HTML in the preview window.

At this point, FLAMS knows about the HTML and the semantics, and we can look at the HTML in VS Code or the FLAMS dashboard – of course, features like the fancy pop-up windows require a semantically informed back-end infrastructure, in the form of the FLAMS system. However, FLAMS *can* dump a standalone and self-contained HTML version for you. Let's do that now:

With our `all.en.tex` file open and everything built as above, click the `Export Standalone HTML`-button in the IDE (see Figure 13).

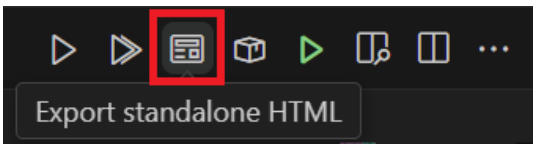



Figure 13: Exporting HTML in the IDE

In the dialog box that opens now, select an **empty** directory and FLAM will dump a standalone version of our `all.en.tex` document there. You will still not be able to open it in the browser directly without issues, because most browsers forbid javascript modules on the `file://` protocol, but opening the file via `http` will yield the desired result, and you can now upload the directory's content to wherever you might want to use it.

If you want to test this, a quick and easy way to do so is to use VS Code: You can install the `Live Server` extension, open the directory and click the `Go Live` button on the lower right of the window, which will start a small web-server in the selected directory and open its `index.html` in the browser for you.

# Chapter 3

# Mathematical Concepts

So far, we have seen how to declare and reference symbols generate semantic macros for text symbols, collect them in modules and document them properly.

But where sTeX really shines is when it comes to mathematics and related subject areas: semantic macros are significantly more useful when used for generating symbolic notations in math mode, and by associating symbols with (flexi-)formal semantics, sTeX can even *check* that your content is (to some degree) formally correct, or at least well-formed.

Alos sTeX provides specialized functionality for mathematical statements: the text fragments marked as Definition, Theorem, Proof that are iconic to mathematical documents.

The example snippets in this chapter can be found in the math archive `sTeX/MathTutorial`. If you downloaded the `sTeX/Documentation` archive in the sTeX IDE, you already have that archive. If not, you can download it from within the IDE, as described in Part 1 (The Basics) in the sTeX tutorial.

## 3.1 Simple Symbol Declarations

We will start with symbols and semantic macros for mathematical concepts and objects and their contribution to mathematical formulae.

### 3.1.1 Semantic Macros and Notations

Let us start with a very fundamental concept; namely equality. As you should by now know, declaring a new symbol requires a module, so let's open a new one and use `\symdecl`:

```
\begin{smodule}{Equality}
  \symdecl{equal}
\end{smodule}
```

As mentioned in Section 1 (Modules and Simple Symbol Declarations), the starred variant `\symdecl*` does not create a semantic macro, so presumably, the variant without a `*` *does*. And indeed, we now have a macro `\equal`, which however will produce errors if we try to use it. That's because we haven't told sTeX what to do with it yet.

A **semantic macro** is a LaTeX-macro that allows for referencing a symbol itself, or – in the case of e.g. a function – the *application* of a symbol to (one or multiple) *arguments*; primarily by invoking a symbol's notation in *math mode*.
The command `\symdecl{macroname}` declares a new symbol with name `macroname` and a semantic macro `\macroname`. In the case where we want the name and the semantic macro to be distinct, the command `\symdecl{macroname}[name=some name]` declares the name of the symbol to be `some name` instead.
The starred variant `\symdecl*{name}` declares the concept with the given name, but does not generate a semantic macro.

So let's provide equality with a notation. As a first step, we should let STEX know that "equal" takes two arguments. We might also want to shorten the semantic macro to e.g. `\eq`, without changing the name. Hence:

```
\symdecl{eq}[name=equal,args=2]
```

Next, we add an infix notation with the `\notation` macro:

```
\notation{eq}{#1 = #2}
```

That seems like a lot to write, so for the very common case where we want to declare a symbol with a semantic macro and a notation all at once, the `\symdef` macro does all three by combining the optional and mandatory argument of `\symdecl` and `\notation`:

```
\symdef{eq}[name=equal,args=2]{#1 = #2}
```

and indeed, we can now use the `\eq` macro in math mode to invoke our new notation: `$\eq{a}{b}$` now yields $a = b$ – notably without any highlighting (and hover interaction in the HTML) though. Since our semantic macro takes *arguments*, which should be differently highlighted, we need to let our notation know which parts of the notation are highlightable components.

We can do so with the `\comp` and `\maincomp` macros:

The `\comp`-macro marks components to be highlighted in a notation for a symbol taking (one or more) arguments.
This is necessary because it is (nearly) impossible for LaTeX to figure out, which parts of a notation to highlight and which not on its own – in particular, the highlighting should stop for the *arguments* of a semantic macro.
Additionally, the `\maincomp` macro can be used to mark (at most) one notation component to represent the *primary* component of the notation.
Notations that do not take arguments, as well as operator notations, are automatically wrapped in `\maincomp`.

In our case, this applies only to the "=", symbol, so:

```
\symdef{eq}[name=equal,args=2]{#1 \mathrel{\maincomp{=}} #2}
```

You may be wondering about the role of the `\mathrel` macro in the example above: TeX determines spacing/kerning in math mode by assigning a *class* to every

character. Both individual characters and whole subexpressions can be assigned one of these classes using dedicated macros. These are:

| class | TEX macro | examples |
|---|---|---|
| ordinary (default class) | `\mathord` | $\alpha\ i\ \Diamond$ |
| large operator | `\mathop` | $\sum\ \prod\ \int$ |
| opening | `\mathopen` | $(\ [\ \langle$ |
| closing | `\mathclose` | $)\ ]\ \rangle$ |
| binary relation | `\mathrel` | $\leq\ >\ =$ |
| binary operator | `\mathbin` | $+\ \cdot\ \circ$ |
| punctuation | `\mathpunct` | $,\ ;$ |

TEX "forgets" the class of an expression if it is wrapped in a `\comp` macro. It is therefore a good idea to wrap any occurence of a `\comp` in the corresponding TEX macro for the desired class (e.g. `\mathrel{\comp{\leq}}`).

Having done so, we can now type `$\eq{a}{b}$` to get $a = b$. Thanks to using `\maincomp`, we now also have an operator notation, which we can invoke using `$\eq!$`, yielding $=$.

What if we want to add more notations? Say we want to be able to invoke equality to get the variant notation $a \equiv b$ ()without changing the intended meaning). If we want to be able to choose one of several notations, we should give the notation an *identifier*.

Let's again modify our earlier notation by adding the identifier `eq` to the optional arguments of `\symdef`, like so:

```
\symdef{eq}[name=equal,args=2,eq]{#1 \mathrel{\maincomp{=}} #2}
```

We can now invoke the specific notation provided here by writing `$\eq[eq]{a}{b}$` to the same effect. But we can also add more notations using the `\notation` macro:

```
\notation{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
```

which we can now invoke with `$\eq[equiv]{a}{b}$`, yielding $a \equiv b$.

By default, the *first* notation provided for a given symbol is considered the *default* notation, which is invoked if the semantic macro is used without an optional argument – hence, `$\eq{a}{b}$` still yields $a = b$.

If we use the starred variant of the `\notation` macro, the notation is set as the new default. Hence, had we done

```
\notation*{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
```

then `$\eq{a}{b}$` would now yield $a \equiv b$.

Any already existing notation can be set as default using the `\setnotation` macro; e.g. instead of using `\notation*`, we could also do

```
\notation{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
\setnotation{eq}{equiv}
```

**Exercise**

Implement the symbol "equal" as above in a new module "`Equality`" and add a documentation such that hovering over the symbol in the HTML yields the following snippet:

> Two objects $a, b$ are considered **equal** (written $a = b$ or $a \equiv b$), if there is no property that distinguishes them.

*Lösung:* Can be found in `[sTeX/MathTutorial]/mod/Equality1.en.tex`

### 3.1.2 Types and Variables

sTEX Every symbol or variable can be assigned a **type**, signifying what "kind of object" the symbol represents, or what (primary) set it is contained in.

Types are particularly useful for *variables*:

sTEX A **variable** represents a *generic* or *unspecified* object.
Variables can be declared using the `\vardef`-macro, whose syntax is analogous to `\symdef`.
Note that variables are local to the current TEX-group (e.g. environment).

Let's leave our equality-module aside for now and turn our attention to something simpler: natural numbers. Consider the following module:

**Example 5**
Input:

```
\begin{smodule}{Nat}
  \symdef{Nat}[name=natural numbers]{\mathbb N}
  \begin{sparagraph}[style=symdoc]
    The \definame{Nat} $\defnotation{\Nat}$ are the numbers
    $0,1,2,...$
  \end{sparagraph}
  \symdef{plus}[name=addition,args=2]{#1 \mathbin{\maincomp{+}} #2}
  \begin{sparagraph}[style=symdoc]
    \Definame{addition} $\defnotation{\plus{a}{b}}$
    refers to the process of adding two \sn{Nat}.
  \end{sparagraph}
\end{smodule}
```

Output:

The **natural numbers** $\mathbb{N}$ are the numbers 0,1,2,...
**Addition** $a+b$ refers to the process of adding two natural numbers.

(like `\definame` and `\definiendum`, the `\defnotation` macro is only allowed in documenting environments like `sparagraph[style=symdoc]` or `sdefinition`, and highlights the notation components marked with `\comp` or `\maincomp` the same way as `\definame` and `\definiendum` do.)

Note, that as the `\Nat` semantic macro does not take any arguments, we do not need to wrap the notation in a `\comp` or `\maincomp`.

The above fragment uses two variables $a$ and $b$. In fact, FLAM will consider them variables even though they are not marked up as such – but since they are not marked up, we are missing out on useful functionality.

Let's change that by adding two variable definitions[1]:

**Example 6**

Input:

```
\begin{sparagraph}[style=symdoc]
  \vardef{va}[name=a]{a}\vardef{vb}[name=b]{b}
  \Definame{addition} $\defnotation{\plus{\va}{\vb}}$
  refers to the process of adding two \sn{Nat}.
\end{sparagraph}
```

Output:

**Addition** $a+b$ refers to the process of adding two natural numbers.

Okay, so now $a$ and $b$ are gray, but besides that, we haven't achieved much yet.

Let's change that by giving the variables the type $\mathbb{N}$:

**Example 7**

Input:

```
\begin{sparagraph}[style=symdoc]
  \vardef{va}[name=a,type=\Nat]{a}\vardef{vb}[name=b,type=\Nat]{b}
  \Definame{addition} $\defnotation{\plus{\va}{\vb}}$
  refers to the process of adding two \sn{Nat}.
\end{sparagraph}
```

Output:

**Addition** $a+b$ refers to the process of adding two natural numbers.

*Now* if we hover over the $a$ and $b$ (in the HTML), it will show us that their type is $\mathbb{N}$!

We can of course also assign types to symbols. In the IDE, find the symbol *"function space"* with semantic macro `\funspace` (in `[sTeX/MathBase/Functions]{mod?Function}`). The OMDoc preview window shows you how to use this symbol (Figure 14). This tells us that if we write `\funspace{a_1,...,a_n}{b}` (depending on which notation we use), we will get $a_1 \times ... \times a_n \to b$.

We want addition to have type $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, hence we do:

```
\symdef{plus}[name=addition,args=2,
  type=\funspace{\Nat,\Nat}{\Nat}
]{#1 \mathbin{\maincomp{+}} #2}
```

[1] Technically, this is called a *variable reservation*, for those in the know.

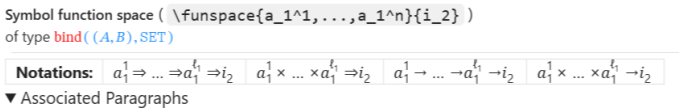



Figure 14: Syntax Preview

So far (and when using the **use** button in the IDE), we have been using the `\usemodule` macro to import content. `\usemodule` is allowed anywhere and imports the referenced module content local to the current TEX group.
Now that we use imported symbols in types (and since we are *in* a module), we need to make sure that the imported modules are also (transitively) *exported*, since our new symbols now *depend* on the imported module.
For that we use the `\importmodule` macro within the module; i.e. the file should now look something like this:

```
\begin{smodule}{Nat}
  \importmodule[sTeX/MathBase/Functions]{mod?Function}
  ...
```

Note that the HTML is aware of this now (after you save): *Clicking* on any occurence of addition now yields Figure 15.

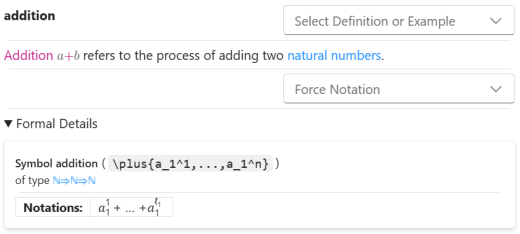



Figure 15: On-Click Popup in the HTML

By virtue of using `[sTeX/MathBase/Functions]{mod?Function}`, we also imported `[sTeX/MathBase/Sets]{mod?Set}`, which gives us the *"collection"* symbol. Let's use this as a type for the natural numbers:

```
\symdef{Nat}[name=natural numbers,type=\collection]{\mathbb N}
```

### 3.1.3 Flexary Macros and Argument Modes

Here is one thing you might wonder: Writing `$\plus{a}{b}$` is one thing, but what if we want to produce $a + b + c + d + e$? Do we really need to write `$\plus{a}{\plus{b}{\plus{c}{...}}}$`?

Of course not. We can declare the symbol such that the semantic macro `\plus` expects a (comma-separated) *sequence* of arguments instead of two "normal" arguments.

The optional `args`-argument of `\symdecl` expects a string of characters indicating the semantic macro's **argument modes**. There are four such modes:

- `i` a **simple argument**,
- `a` a – *(left or right) associative* – **sequence argument**, represented as a single TEX-argument `{a,b,...}`,
- `b` A **binding argument** that expects a variable that is bound by the symbol in its application, and
- `B` A **binding sequence argument** of arbitrarily many bound variables by the symbol (`{x,y,z,...}`).

If `args` is given as a number $n$ instead, the semantic macro takes $n$ arguments of mode `i`.

**Example 8**

- For `\plus{a,b,c}` yielding $a+b+c$, we do `\symdecl{plus}[args=a]`,
- for `\inset{a,b,c}{A}` yielding $a,b,c \in A$, we do `\symdecl{inset}[args=ai]`,
- in `\add{i}{1}{n}{f(i)}` yielding $\sum_{i=1}^{n} f(i)$, the variable $i$ is bound in the expression, we hence do `\symdecl{add}[args=biii]`,
- in `\foral{x,y,z}{P(x,y,z)}` yielding $\forall x,y,z.\ P(x,y,z)$, the variables $x,y,z$ are all bound by the $\forall$, we hence do `\symdecl{foral}[args=Bii]`.

So when we wrote `\symdecl{plus}[args=2]`, this was actually shorthand for `\symdecl{plus}[args=ii]`.

Let's revise our previous declaration and the syntax of the `\plus` macro:

```
\symdef{plus}[name=addition,args=a,
  type=\funspace{\Nat,\Nat}{\Nat}
]{#1 \mathbin{\maincomp{+}} #2}
\begin{sparagraph}[style=symdoc]
  \vardef{va}[name=a]{a}\vardef{vb}[name=b]{b}
  \Definame{addition} $\defnotation{\plus{\va,\vb}}$
  refers to the process of adding two \sn{Nat}.
\end{sparagraph}
```

Now we get new errors, that are easy to explain: Our notation `{#1 \mathbin{\maincomp{+}} #2}` refers to *two* arguments, but our semantic macro only takes *one* (albeit a sequence argument). We now need to let STEX know what to do with the sequence argument in our notation. Using the `\argsep` macro, we can tell STEX to insert the *separator* "+" between the individual elements of the argument sequence `#1`:

```
\symdef{plus}[name=addition,args=a,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{+}}}}
```

TODO[2] TODO[3]

Now we can finally write `$\plus{a,b,c,d,e}$` and get $a+b+c+d+e$ – hooray!

[2] TODO: argmap
[3] TODO: argarraymap

**Exercise**
Analogously to the above, implement a symbol "`multiplication`" with semantic macro `\mult`, that takes a single sequence argument and has a default notation such that `\mult{a,b,c}` produces $a \cdot b \cdot c$.

---

*Lösung:* Can be found in `[sTeX/MathTutorial]mod/Nat.en.tex`

---

### 3.1.4 Precedences

If you have done the previous exercise, you now have semantic macros `\plus` and `\mult` at your disposal. We can of course nest them to produce e.g. $a + b \cdot c$ (with `$\plus{a,\mult{b,c}}$`). If we do `$\mult{a,\plus{b,c}}$` however, we get $a \cdot b + c$. Annoying – we now have to insert parentheses: `$\mult{a,(\plus{b,c})}$`... or do we?

We do *not*. Instead, we can assign *precedences* to notations to have sTeX insert parentheses automatically.

sTeX

`\notation` (and hence `\symdef`) take an optional argument `prec=<opprec>;<argprec1>x...x<argprec n>` consisting of an **operator precedence** `<opprec>` and for each argument `k` an **argument precedence** `<argprec k>`.
All precedences are integers, e.g. 10 or -500. It is good practice to use precedences that leave enough room to smuggle values inbetween, so that we can fine-tune them later as more symbols may intervene.
The precise numbers used for precedences are arbitrary – what matters is which precedence is higher than which other precedence when used together.
By default, all precedences are 0, unless the symbol takes no arguments, in which case the operator precedence is `\neginfprec` (negative infinity).
If we only provide a single number in `prec=`, this is taken as both the operator precedence and all argument precedences.

The *lower* a precedence, the *stronger* a notation binds its arguments. In our particular case, we want multiplication to bind stronger than addition, so we can (arbitrarily) assign them precedences e.g. 10 and 20:

```
\symdef{plus}[name=addition,args=a,assoc=bin,prec=20,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{+}}}}
\symdef{mult}[name=multiplication,args=a,assoc=bin,prec=10,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{\cdot}}}}
```

And now if we type `$\mult{a,\plus{b,c}}$`, sTeX will automatically insert parentheses and yield $a \cdot (b + c)$ – and conversely, if we do `$\plus{a,\mult{b,c}}$`, sTeX will *not* insert parentheses and yield $a + b \cdot c$.

### 3.1.5 Finishing Equality

You might wonder if – as with addition – we can make "equal" take a sequence argument as well. Naturally, we can:

```
\symdef{eq}[name=equal,args=a,eq,
  type=\funspace{\vA,\vA}{\prop}
]{\argsep{#1}{\mathrel{\maincomp=}}}
\notation{eq}[equiv]{\argsep{#1}{\mathrel{\maincomp\equiv}}}
```

### 3.1.6 Variable Sequences

There is a special kind of variable in sTEX for when we want to use *sequences* of variables.

We can use the \varseq macro to declare a new sequence variable; in the simplest case that looks something like the following:

```
\varseq{seqn}[name=n,type=\Nat]{1,\ellipses,k}{\maincomp{n}_{#1}}
```

We have just declared a new variable sequence of type $\mathbb{N}$, that ranges over indices $1, \ldots, k$, with notation $n_i$ for some specific index $i$.

If we now do \seqn{i}, we get $n_i$, and if we do \seqn!, we get $n_1, \ldots, n_k$.

We can also do multi-dimensional sequences, e.g.

```
\varseq{seqm}[name=m,type=\Nat,args=2]
  {{1}{1},\ellipses,{\ell}{k}}
  {\maincomp{m}_{#1}^{#2}}
```

Now \seqm{i}{j} produces $m_i^j$, and \seqm! produces $m_1^1, \ldots, m_\ell^k$.

Of course, we can manually change the way \seqn! is typeset by providing an explicit operator notation using op=; e.g. if we do

```
\varseq{seqn}[name=n,type=\Nat,op={(n_i)_{i=1}^k}]
  {1,\ellipses,k}{\maincomp{n}_{#1}}
```

then \seqn! produces $(n_i)_{i=1}^k$.

So far so nice, but sequence variables get especially useful in combination with sequence arguments: Consider for example the \plus semantic macro for addition. This expects one sequence argument, or alternatively, a *sequence variable*: \plus{\seqn} now produces $n_1 + \ldots + n_k$, and \eq{\seqm} now produces $m_1^1 = \ldots = m_\ell^k$.

TODO[4]

## 3.2 Statements

Now that we have equality, natural numbers, addition and multiplication at our disposal, let's implement some *statements*. Both addition and multiplication are, for example, *associative* and *commutative*.

We could state these properties directly for the two operations, but we can also first define *associativity* and *commutativity* in general, and then assert them specifically for addition and multiplication.

[4] TODO: seqmap

### 3.2.1 Definitions

Let's define what it means to be *associative*. This means, of course, declaring a new symbol. Note that we don't need a semantic macro for associativity, since there is no notation to attach to it. We will also for now ignore its type. Note however, that associativity is still a property of (binary) operations, so it still makes sense to have the symbol take an *argument*; namely the operation it applies to.

We will also finally provide an actual (more or less) formal *definition* for the symbol, so where we used the `sparagraph` environment with `style=symdoc` before, we will now use the `sdefinition` environment, which also gives us `\definame`, `\definiendum`, `\defnotation` and all that.

A first variant of a corresponding module could look like this:

**Example 9**

Input:

File [sTeX/MathTutorial]`props/Associative1.en.tex`

```
\begin{smodule}{Associative}
  \importmodule{mod?Equality}

  \symdecl*{associative}[args=1]
  \begin{sdefinition}[for=associative]
    \vardef{vA}[name=A,type=\collection]{A}
    \vardef{vop}[name=op,type=\funspace{\vA,\vA}\vA,args=a,assoc=bin]
      {\argsep{#1}{\mathbin{\maincomp{\circ}}}}
    %
    A binary operation $\fun{\vop!}{\vA,\vA}\vA$ is called
    \definame{associative}, if
    $\eq{
      \vop{(\vop{a,b}),c},
      \vop{a,(\vop{b,c})}
    }$ for all $\inset{a,b,c}\vA$.
  \end{sdefinition}
\end{smodule}
```

Output:

**Definition 3.2.1.** A binary operation $\circ : A \times A \to A$ is called **associative**, if $(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c \in A$.

Note, that the semantic macros `\fun` and `\inset` come from `[sTeX/MathBase/Functions]mod?Function` and `[sTeX/MathBase/Sets]mod?Set`, respectively. Also note, that the variable declaration for `\vop` makes use of all the fun features we already discussed for addition.

Note that the above is more than good enough, if you merely want to produce nice-looking, "wikified" HTML and PDF documents. The rest of this subsection will cover how to add more flexiformal semantics to the above.
If this seems laborious and/or difficult, keep in mind that this is to some degree experimental still, and you are not forced to go overboard with semantic annota-

tions!
But if you aim to create a "library of symbols" for mathematical concepts, then all of the possibilities that we discuss here will add value for the community. Generally, the higher the ratio of readers to authors the more any investment in semantization will pay off.

### Semantic Macros in Text Mode

The first thing we can do to further improve this document is marking up the "for all" in the definition – after all, there naturally is a symbol for the universal quantifier, which can be found in `[sTeX/Logic/General]mod/syntax?UniversalQuantifier` and has the semantic macro `\foral` (as to not conflict with the TEX primitive macro `\forall`).

The naive approach would be to replace the "for all" by e.g. `\sr{foral}{for all}`. That would (correctly) associate and highlight the text fragment with the symbol "universal quantifier", *but* we are not just referencing the symbol here – we are actually using it, by *applying* it to the variables $a, b, c$ and the expression $(a \circ b) \circ c = a \circ (b \circ c)$.

In *math mode*, we can just use the semantic macro `\foral` – that will take two arguments (of modes `B` and `i`) and produce the corresponding notation, so that

```
$\foral{\inset{a,b,c}{\vA}}{
  \eq{ \vop{(\vop{a,b}),c} , \vop{a,(\vop{b,c})} }
}$
```

will produce $\forall a, b, c \in A.(a \circ b) \circ c = a \circ (b \circ c)$.

In *text mode*, however, we don't have a specific notation – instead, the specific "notation" is whatever sentence we want to mark up semantically. In text mode, semantic macros therefore behave differently:

1. They take *precisely* one argument, regardless of how many arguments the macro would take in math mode or (equivalently) the `args` property of the symbol.
2. *Within* that argument, we can use `\comp` to highlight arbitrary text fragments, and
3. we can use the `\arg` macro to mark up the *actual* arguments that the symbol is supposed to be applied to.

   `\arg` takes as optional argument the index of the argument that is being marked up; if not they are used consecutively. The starred variant `\arg*` produces no output.

So we could now do

```
\foral{\comp{For all} $\arg{\inset{a,b,c}{\vA}}$, we have
  $\arg{
    \eq{ \vop{(\vop{a,b}),c} , \vop{a,(\vop{b,c})}}
  }$
}
```

which produces "For all $a, b, c \in A$, we have $(a \circ b) \circ c = a \circ (b \circ c)$".

In our case though, we want to "switch the arguments around" – first comes the equation, then the variables to be bound. Hence:

```
\foral{
  $\arg[2]{
    \eq{ \vop{(\vop{a,b}),c}, \vop{a,(\vop{b,c})} }
  }$
  \comp{for all}
  $\arg[1]{ \inset{a,b,c}{\vA} }$
}
```

which produces "$(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c \in A$".

### Definientia

Now we have a fully semantically annotated expression in the definition for "associative". Can we let MMT know, that this expression really is *the* definition of the symbol?

Yes, we can. All we need to do is wrap the sentence in a `\definiens` macro (plural: *definientia*; like the word *"definiendum"* refers to "the term being defined", *"definiens"* refers to "the thing the term is being defined *as*").

The `\definiens` macro is only allowed within the `sdefinition` environment, and requires that the environment lists the symbol that gets the definiens attached explicitly in its `for=` argument. It is possible to attach definientia to multiple symbols within an `sdefinition` environment, in which case the symbol needs to be provided as an optional argument, e.g. we could do `\definiens[associative]{...}`. Since "associative" is the only symbol being defined in our definition, we can omit that argument.

Alternatively, as with types we can attach definientia to a `\symdecl` directly using the optional argument `def=...`.

At this point, you might justifiably wonder, why we even still need to declare associative with `\symdecl*` before we define it. And indeed, we don't – the `sdefinition` environment takes the same optional arguments as the `\symdecl` macro, and if we explicitly provide a `name=` (or a `macro=`), it will generate a symbol for us. We can hence get rid of the `\symdecl*` and instead do:

```
1 \begin{sdefinition}[name=associative,args=1]
2   ...
3 \end{sdefinition}
```

One more problem remains: We stated that associative is to take one argument – but we haven't told STEX what it is yet. In our case, the argument is represented by the variable `\vop`. In fact, chances are that arguments to symbols in types or definientia are almost always represented by some variable.

We can use one of two ways to a variable as being an argument:

1. If the variable (e.g. `\vop` with name `op`) was already declared prior to the `sdefinition` environment, we can use the `\varbind` macro in the environment; e.g. by adding `\varbind{op}`.
2. We can move (or copy) the `\vardef` for the variable into the environment and add `bind` to its optional arguments.

In total, our fully annotated definition now looks like this:

**Example 10**
Input:

File [sTeX/MathTutorial]props/Associative.en.tex

```
\begin{sdefinition}[name=associative,args=1]
  \vardef{vA}[name=A,type=\collection]{A}
  \vardef{vop}[name=op,type=\funspace{\vA,\vA}\vA,
    args=a,assoc=bin,bind % <- argument for the symbol
  ]{\argsep{#1}{\mathbin{\maincomp{\circ}}}}
  \vardef{va}[name=a,type=\vA]{a}
  \vardef{vb}[name=b,type=\vA]{b}
  \vardef{vc}[name=c,type=\vA]{c}
  %
  A binary operation $\fun{\vop!}{\vA,\vA}\vA$ is called
  \definame{associative}, if
  \definiens{\foral{$\arg[2]{\eq{
    \vop{(\vop{\va,\vb}),\vc},
    \vop{\va,(\vop{\vb,\vc})}
  }}$ \comp{for all} $\arg[1]{\inset{\va,\vb,\vc}\vA}$}}.
\end{sdefinition}
%
```

Output:

**Definition 3.2.2.** A binary operation $\circ : A \times A \to A$ is called **associative**, if $(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c \in A$.

And indeed, if we look at the OMDoc tab of the HTML preview, we can see that not only does MMT attach the definiens to the symbol, it has also inferred the type of "associative" from the definiens (Figure 16).

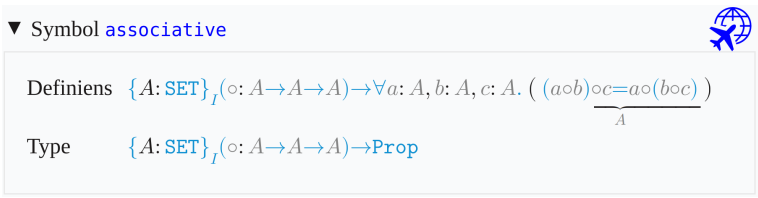



Figure 16: Type Inferred from Definiens

### Using Symbols Without Semantic Macros and Exporting Code in Modules

So now we don't have a semantic macro for "associative", but it *does* take an argument. How can we ever actually *use* the symbol now?

The answer is: with the `\symuse` macro. Like `\symref` and friends, `\symuse` takes a symbol name or the name of its semantic macro as argument, but behaves otherwise like using a semantic macro directly. So for, say, addition, `\symuse{addition}` and `\symuse{plus}` behave exactly like `\plus`.

In our case, this means we can do `\symuse{associative}`. "associative" does not have a notation, but in practice, we say something like *"+ is associative"* rather than using some specific mathematical notation for the same thing.

Combining this with what we just learned, we can now say that addition is associative by doing:

```
\symuse{associative}{$\arg{\plus!}$ \comp{is associative}}
```

In fact, we would do the exact same thing every time we want to say that *some* operator is associative, so it makes sense to introduce a macro for this. In fact, such a macro is easy to define using standard LaTeX methods. This is where `\STEXexport` becomes very handy:

In a module, we can put arbitrary LaTeX code in an `\STEXexport`, and this code will be executed every time the module is imported via `\usemodule` or `\importmodule`. This is especially useful for macro definitions, and this way modules can almost act like LaTeX packages!

So we can define a new macro `\isassociative` that applies "associative" to an arbitrary operation and produces the semantically marked-up text "`#1` is associative", and wrap that macro definition in an `\STEXexport`, and whenever we use the `Associative` module, we also get the `\isassociative` macro:

```
\STEXexport{
  \def\isassociative#1{
    \symuse{associative}{\arg{#1} ~is ~\comp{associative}}
  }
}
```

And now, we can do e.g. `\isassociative{$\plus!$}` to produce "$+$ is associative".

For technical reasons, `\STEXexport` processes its content in the `expl3` category code scheme – what this means is that all spaces are ignored entirely, and the characters `_` and `:` are valid characters in macro names.
In practice, this means you will have to use the `~` character for spaces, and if you want to use a subscript `_`, you should use the macro `\c_math_subscript_token` instead.

**Exercise**
Analogously to all the above, implement a module for *commutativity*; i.e the property of a binary operation that $a \circ b = b \circ a$ for all $a, b$. Make the module export a macro `\iscommutative` analogously to `\isassociative`.

---

*Lösung:* Can be found in `[sTeX/MathTutorial]props/Commutative.en.tex`

---

TODO[5]

### 3.2.2 Assertions

Having defined associativity and commutativity, we can now assert that both properties hold for addition and multiplication.

For *assertions* (i.e. theorems, lemmata, axioms, claims,...), sTeX provides the `sassertion` environment.

In the simplest case, that can look like the following:

[5] TODO: intent?

```
\begin{sassertion}
  \isassociative{\Sn{plus}}
\end{sassertion}
```

which yields

> Addition is associative

Do we want this to be typeset as a **Theorem**? For that we just add a `[style=theorem]` to the `sassertion` environment, provided we have a customization for that – (see chapter 9). We can also load the stexthm package, which uses the amsthm package to provide common typesettings for the types: `theorem`, `observation`, `corollary`, `lemma`, `axiom` and `remark`.

So far, this is not too useful – after all, we could have just as well used e.g. the `amsthm` package and gone straight for the non-STEX variant

```
\begin{theorem}
  \isassociative{\Sn{plus}}
\end{theorem}
```

But as with `sdefinition`, we can immediately add a corresponding symbol in the `sassertion` environment, and have it be documented directly by the environment:

```
\begin{sassertion}[style=theorem,name=addition is associative]
  \isassociative{\Sn{plus}}
\end{sassertion}
```

And now, if we do `\sn{addition is associative}`, we get addition is associative with a corresponding hover pop-up (in the HTML).

Of course, the usefulness of this grows with more elaborate assertions. For very short assertions such as the above, we might not even want to typeset them in such a space hungry manner.

For that purpose, we provide the `\inlineass` macro (and analogously: `\inlinedef` for `sdefinition`), which takes the same optional arguments as the environment. So we could also do:

```
\inlineass[name=addition is associative]{\isassociative{\Sn{plus}}}
```

So far, MMT is blissfully unaware of the semantic contents of our assertions. We can easily remedy that by wrapping the expression representing the assertion in a `\conclusion` macro, analogously to the definiens macro in `sdefinition`s:

```
\inlineass[name=addition is associative]{
  \conclusion{\isassociative{\Sn{plus}}}
}
```

We can now see the statement in the OMDoc tab of the HTML preview (Figure 17).

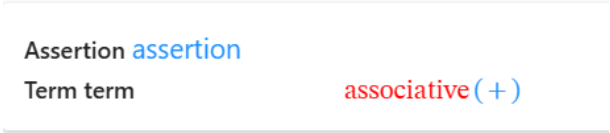



Figure 17: Assertion Statement in OMDoc

Not exactly pretty – the OMDoc tab uses notations to render content, and we did not provide any for associative.

### 3.2.3 Proofs

sTeX provides the `sproof` environment for marking up *proofs*.
The markup mechanism for `sproof` is still highly experimental and likely subject to change in the near future. As such, we omit a closer explanation of its usage until the syntax and functionality have sufficiently stabilized.

## 3.3 Mathematical Structures

A common concept in mathematics is that of a mathematical structure – a *tuple* of interdependent components. For example: A *monoid* is a structure $\langle M, \circ, e\rangle$ such that certain axioms hold; where $M$ is a set, $\circ$ is a binary operation, and $e \in M$.

From a representational perspective, this is particularly interesting: $M$, $\circ$ and $e$ in the above are not symbols in the same way that the previous symbols we considered were – they don't represent definite objects. Instead, they are *components* of some other object, namely a monoid; where a *particular* monoid could either be a fixed object (such as $\langle \mathbb{Z}, +, 0\rangle$) or an *indefinite* monoid; i.e. a variable. We call the components of a mathematical structure **fields**.

In this section, we will discuss how to declare and use mathematical structures in sTeX, build them up modularly, and connect them among each other to avoid duplication.

We will do so by considering *lattices* both algebraically and order-theoretically, and identify the two perspectives.

### 3.3.1 Declaring and Using Structures

The simplest kinds of structures are *magmas* and *(directed) graphs*, so we might as well start there:

**Definition 3.3.1.** A **magma** is a structure $\langle U, \circ\rangle$, where $U$ is a collection and $\circ$ a binary operation $U \times U \to U$.

The obvious start is to create a new module `Magma`. Within this module, we import the `Functions` module so we can later assign a type to the operation. We can then use the `mathstructure` environment, that creates a new symbol "magma":

```
\begin{smodule}{Magma}
  \importmodule[sTeX/MathBase/Functions]{mod?Function}
  \begin{mathstructure}{magma}
    ...
  \end{mathstructure}
\end{smodule}
```

`mathstructure` behaves very similarly as `smodule` – within the environment, we can declare new symbols, notations and all that.

So within the `mathstructure`, we can add symbols for the two fields $U$ and $\circ$:

```
\symdef{univ}[name=universe,type=\collection]{U}
\symdef{op}[name=operation,args=a,assoc=bin,
  type=\funspace{\univ,\univ}\univ
]{\argsep{#1}{\mathbin{\maincomp{\circ}}}}
```

Once we close the environment (with `\end{mathstructure}`), the symbols are "gone". However, we now have a new symbol "magma" with semantic macro `\magma`. It's usage is somewhat more complicated than "normal" semantic macros, but one thing we *can* do with it now is `$\magma!$`, which will produce $\langle U, \circ\rangle$.

Notably however, the `\magma` macro is already available *within* the `mathstructure` environment as well.

This allows us to provide an `sdefinition` using the semantic macros declared in the structure:

**Example 11**

Input:

File [sTeX/MathTutorial]`algebra/Magma.en.tex`

```
\begin{mathstructure}{magma}
  \symdef{univ}[name=universe,type=\collection]{U}
  \symdef{op}[name=operation,args=a,assoc=bin,
    type=\funspace{\univ,\univ}\univ]
    {\argsep{#1}{\mathbin{\maincomp\circ}}}

  \begin{sdefinition}[for={magma,univ,op}]
    A \definame{magma} is a \sr{mathstruct}{structure} $\magma!$,
    where $\univ$ is a \sn{collection} and $\op!$
    a binary operation $\funspace{\univ,\univ}\univ$.
  \end{sdefinition}
\end{mathstructure}
```

Output:

**Definition 3.3.2.** A **magma** is a structure $\langle U, \circ\rangle$, where $U$ is a collection and $\circ$ a binary operation $U \times U \to U$.

### Instantiating Structures

More importantly however, we can now declare a variable magma, using the optional `return=` argument. For example, we can now do

```
\vardef{vM}[name=M,return=\magma]{M}
```

and we get the semantic macro `\vM` with which we can do the following:

| Syntax | Result |
|---|---|
| `$\vM!$` | $M$ |
| `$\vM{}$` | $\langle U_M, \circ_M\rangle$ |
| `$\vM{univ}$` | $U_M$ |
| `$\vM{op}!$` | $\circ_M$ |
| `$\vM{op}{a,b,c}$` | $a \circ_M b \circ_M c$ |

In other words: Given a symbol or variable with semantic macro `\foo` and `return=\struct`, then `\foo{<fn>}` behaves like the semantic macro `\fn` *within* the `mathstructure` environment for `struct` – but instantiated for the specific instance `foo`.

By default, STEX attaches the symbol's (or variable's) operator notation as a subscript suffix to the notation component marked with `\maincomp` – e.g., since the "`\circ`" in the notation for `op` is marked with `\maincomp`, doing `$\vM{op}{a,b}$` ultimately outputs `a \circ_{\vM!} b`. Hence, we get $a \circ_M b$.

We can change the way the `\maincomp` notation component is modified, by using the optional argument `copm=` in the semantic macro for the mathematical structure. For example, to not change it at all, we can do:

```
\vardef{vM}[name=M,return={\magma[comp=##1]}]{M}
```

...or to suffix it with a `'`, we can do

```
\vardef{vMp}[name=Mp,return={\magma[comp=##1']}]{M'}
```

This allows us to do things like:

```
Let $\vM!:=\vM{}$ and $\vMp!:=\vMp{}$ \sns{magma}. Then...
```

yielding

> Let $M := \langle U, \circ\rangle$ and $M' := \langle U', \circ'\rangle$ magmas. Then...

We can also *assign* fields to (arbitrary) expressions, by doing `name=<tex>` in square brackets. For example we can do the following:

```
\vardef{vA}[type=\collection]{A}
\vardef{vM}[name=M,return={\magma[comp=##1][univ=\vA]}]{M}
\vardef{vMp}[name=Mp,return={\magma[comp={##1}'}][univ=\vA]}]{M'}

Let $\vM!:=\vM{}$ and $\vMp!:=\vMp{}$ \sns{magma} on $\vA$....
```

> Let $M := \langle A, \circ\rangle$ and $M' := \langle A, \circ'\rangle$ magmas.

Of course, we can also use `return=` with variable sequences – for example:

```
\varseq{vMs}[name=M,return={\magma[comp={##1}_{#1}]},op=(M_i)_1^n]
  {1,\ellipses,n}{\maincomp{M}_{#1}}
Let $\vMs! := \vMs{i}{}_1^n$ a sequence of \sns{magma}...
```

> Let $(M_i)_1^n := \langle U_i, \circ_i\rangle_1^n$ a sequence of magmas...

Note that in the above, it seems that using `#1` in the `return` argument is allowed. Indeed, it is - the `return` statement takes the same arguments as the semantic macro itself does and is appropriately instantiated. Since the first (and only) argument to the

sequence `\vMs` is the index, when doing `\vMs{i}...` the `#1` in the `return`-statement will be replaced by `i`.

Also, note that if we want to produce $M_i$ – i.e. the magma at index $i$ in the sequence, we can do `\vMs{i}!`.

Think of the `!` as a "stop sign" - if the expression up to the `!` has an associated presentation, the `!` tells sTeX to "stop eating arguments" and present whatever it has until now.

### 3.3.2 Extending Structures and Axioms

It is extremely common to "build up" structures in a hierarchical manner by adding new fields or axioms: A *semigroup* is an associative magma. A *band* is an idempotent semigroup. A *monoid* is a semigroup with a unit. A *partial order* is an antisymmetric preorder.

We alluded to the fact earlier, that the `mathstructure` environment behaves like an `smodule` – that is literally true: Every `mathstructure foo` in a module `FooMod` is in fact also a module `?FooMod/foo-module`. We can therefore easily extend structures using `\importmodule{...?FooMod/foo-module}` – but extending structures is so common, and using `\importmodule` tiring, that there is a shortcut: the `extstructure` environment. It takes as second argument a comma-separated list of structure names. That allows us to easily define semigroups:

**Example 12**

Input:

File [sTeX/MathTutorial]`algebra/Semigroup.en.tex`

```
8  \begin{extstructure}{semigroup}{magma}
9    \begin{sdefinition}
10     A \definame{semigroup} is a \sn{magma} $\semigroup!$,
11     where \inlineass[name=associative axiom]{
12       \conclusion{\isassociative{$\op!$}}.
13     }
14   \end{sdefinition}
15 \end{extstructure}
```

Output:

**Definition 3.3.3.** A **semigroup** is a magma $\langle U, \circ\rangle$, where $\circ$ is associative.

Note our usage of `\inlineass` to generate a new symbol for the associative axiom.

If we look at the OMDoc tab in the HTML preview window, we can see the output in Figure 18.

So MMT has decided that our statement is an *axiom*.

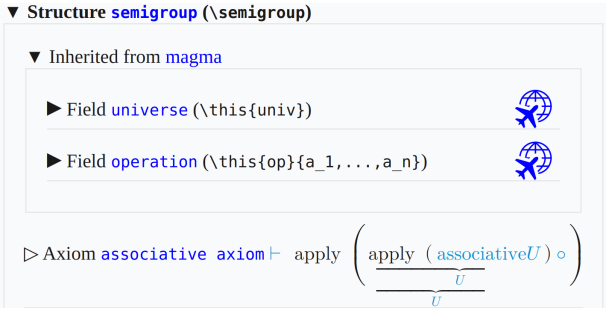



Figure 18: Axioms in OMDoc

### Conservative Extensions

For structures, there is a *critical* distinction between *defined* and *undefined* symbols; and analogously between *theorems* and *axioms*.

Remember that structures are more like *templates* that are *instantiated* by particular objects. An *undefined* field in a structure, in that sense, is like an *obligation*: If something is supposed to be a semigroup, it *has to* have a universe, an operation and the operation needs to satisfy the associative axiom.

*Defined* fields on the other hand have a *definiens* on the basis of the remaining fields – they don't need to be explicitly provided for something to instantiate the structure; if all the *undefined* fields are provided, the *defined* ones we get *"for free"*.

The same holds for *theorems*: If a statement is *provable* from the axioms, then we don't need to explicitly prove it to hold for some particular instance – we have a proof already, provided the axioms hold.

The relation between axioms and theorems is not just analogous to that between undefined and defined symbols: It is the very same. Remember the judgments as types paradigm?
For a proposition $P$, an assertion in sTeX induces a symbol of type $\vdash P$. Without a proof, this symbol is *undefined* – and hence an *axiom*. A *proof* for $P$ is a specific term of type $\vdash P$ – i.e. a potential *definiens*. To prove an assertion turns it into a *theorem*, which is to say that the symbol can be *defined*.

One consequence of this is: Extending a structure only by *defined* fields does not actually (conceptually) introduce a *new* structure – every instance of the old one *should* also be an instance of the new one. The new fields are basically just "syntactic sugar".

There is a name for extending a structure only by defined fields (or theorems): A *conservative extension*.

sTeX provides the `extstructure*` environment for that purpose. Unlike `extstructure`, it does *not* take a name (technically, sTeX generates one internally). Instead, conceptually `extstructure*` modifies the extended structure directly, rather than generating a new structure. The caveat however is, that every symbol introduced in an `extstructure*` **must** be defined.

Consider the following conservative extension:

**Example 13**
Input:

File [sTeX/MathTutorial]algebra/MagmaSquare.en.tex

```
\begin{extstructure*}{magma}
  \begin{sdefinition}[macro=sq,args=1]
    \notation{sq}[op=\cdot^2]{{#1}^{\comp 2}}
    \vardef{va}[name=a,type=\univ,bind]{a}
    Let $\inset{\va}{\univ}$. We define
    $\defnotation{\sq{\va}} := \definiens{\op{\va,\va}}$.
  \end{sdefinition}
\end{extstructure*}
```

Output:

**Definition 3.3.4.** Let $a \in U$. We define $a^2 := a \circ a$.

Via \definiens, the new symbol sq is now *defined* (note the macro= argument, taht generates a semantic macro as well). Whenever we import the containing module, we now have an additional field sq in (any extension of) magma – e.g., as semigroup extends magma, the following is now valid:

```
\usemodule[sTeX/MathTutorial]{algebra?MagmaSquare}
\vardef{vsg}[name=S,return=\semigroup]{S}
$\vsg{sq}{a}$
```

...producing $a^2$.

### 3.3.3 Nesting Structures and \this

A prehaps not too surprising, but a notable aspect of structures is that fields themselves can be instances. This is important for example for implementing *vector spaces*, but can also be used to bundle things that are not normally thought of as structures, such as objects with certain defining properties.

Take as an example, the notion of a (magma) homomorphism:

**Definition 3.3.5.** Let $M_1 = \langle U_1, \circ_1\rangle$ and $M_2 = \langle U_2, \circ_2\rangle$ magmas. A **magma homomorphism** is a function $F : U_1 \to U_2$ such that $F(a \circ_1 b) = F(a) \circ_2 F(b)$ for all $a, b \in U_1$.

So a homomorphism is a function with certain properties. And structures can be used to "bundle" the function itself with both the magmas on whose universes the function operates, as well as the *axiom* that *makes* it a homomorphism. After all, considered as a mere function, $F : U_1 \to U_2$ contains no information about the operation with respect to which it is homomorphic.

The first thing to note is that we can provide mathstructure with an optional argument for a *name* distict from the name of its semantic macro. We then add two fields that return magmas. So far, so unexciting:

```
\begin{mathstructure}{magmahom}[magma homomorphism]
  \symdef{dom}[name=domain,return={\magma[comp={##1}_1]}]{M_1}
  \symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}]{M_2}
```

For the function itself, we know how to give it a maningful type, already:

```
\symdef{f}[type=\funspace{\dom{univ}}{\cod{univ}},args=1]{???}
```

...but what should its notation be? Ideally we would want it to just be the notation of whatever particular instance it is – in informal mathematics, we rarely distinguish notationally between a homomorphism and its underlying function (to the point where it's not clear, whether *informally* the distinction is even meaningful). Similarly, we rarely distinguish e.g. between a magma (or semigroup, monoid, group, ring, vector space,...) and its underlying universe.

This is where `\this` comes into play (pun intended). Within an `mathstructure` or `exstructure`, or in the context of a particular instance of one, `\this` represents "the" instance.

We can set it in the context of `mathstructure` as a further optional argument; e.g.

```
\begin{mathstructure}{magmahom}[magma homomorphism,this=F]
```

and then use `\this` in the notation for the function. We can further provide the homomorphism condition as an axiom using `\inlineass`:

**Example 14**

Input:

File [sTeX/MathTutorial]algebra/Homomorphism.en.tex

```
\begin{mathstructure}{magmahom}[magma homomorphism,this=F]
  \symdef{dom}[name=domain,return={\magma[comp={##1}_1]}]{M_1}
  \symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}]{M_2}
  \symdef{f}[op=\this,args=1,
    type=\funspace{\dom{univ}}{\cod{univ}}
  ]{\this \dobrackets{#1}}

  \begin{sdefinition}[for={magmahom,dom,cod,f}]
    \vardef{va}[name=a,type=\dom{univ}]{a}
    \vardef{vb}[name=b,type=\dom{univ}]{b}
    Let $\dom!=\dom{}$ and $\cod!=\cod{}$ \sns{magma}.
    A \definame{magmahom} is a function
    $\fun{\f!}{\dom{univ}}{\cod{univ}}$ such that
    \inlineass[name=homomorphism condition]{\conclusion{\foral{
      $\arg[2]{\eq{
        \f{\dom{op}{\va,\vb}}, \cod{op}{\f{\va},\f{\vb}}
      }}$ \comp{for all} $\arg[1]{\inset{\va,\vb}{\dom{univ}}}$.
    }}}
  \end{sdefinition}
\end{mathstructure}
```

Output:

**Definition 3.3.6.** Let $M_1 = \langle U_1, \circ_1 \rangle$ and $M_2 = \langle U_2, \circ_2 \rangle$ magmas. A **magma homomorphism** is a function $F : U_1 \to U_2$ such that $F(a \circ_1 b) = F(a) \circ_2 F(b)$ for all $a, b \in U_1$.

Now if we instantiate our magma homomorphism:

```
\vardef{vh}[name=H,return={\magmahom[this=H]}]{H}
```

Here is a list of what we can do now:

| Syntax | Result |
|---|---|
| `$\vh!$` | $H$ |
| `$\vh{}$` | $\langle M_1, M_2, H\rangle$ |
| `$\vh{f}!$` | $H$ |
| `$\vh{f}{a}$` | $H(a)$ |
| `$\vh{dom}!$` | $M_1$ |
| `$\vh{cod}{}$` | $\langle U_2, \circ_2\rangle$ |
| `$\vh{cod}{univ}$` | $U_2$ |
| `$\vh{dom}{op}!$` | $\circ_1$ |
| `$\vh{cod}{op}{a,b,c}$` | $a \circ_2 b \circ_2 c$ |

Note how – as one would expect – we can treat `\vh{dom}` and `\vh{cod}` like any other instance of magma.

Note that some of the outputs in the above table are probably not quite what we want. Determining the precise typesetting of an expression involving *nested paths* of fields is difficult, to say the least (e.g., what exactly should `\this` refer to in a deeply nested sequence of fields?).
Using instances within structures is still very useful; at the very least when defining structures. When subsequently *using* structures, however, accessing fields of fields (of fields (of ...)) of an instance should be avoided.
Luckily, there is rarely a need for doing so – in practice, those fields we might want to access in such a way, we usually also want to provide specific notations and talk about independently of the "containing" instance, such that introducing a new variable (or symbol), and assigning the corresponding field to that variable, makes considerably more sense. And subsequently using the variable is easier than concatenating `{...}`, too.

## 3.4 Complex Inheritance and Theory Morphisms

We are starting to approach seriously experimental territory.
While the theory behind all the following is relatively well understood, and their implementation in MMT is mature, the same can not be said out the implementation in sTEX.
There are still kinks to be ironed out, but feel free to experiment.

We now have all the tools available to progress towards something more interesting. Here is a list of documents with respective modules and symbols we will build on in the following:

`[sTeX/MathTutorial]props/Idempotent.en.tex`

**Definition 3.4.1.** Let $e \in A$ and $\circ : A \times A \to A$. $e$ is called **idempotent** with respect to $\circ$, if $e \circ e = e$.

**Definition 3.4.2.** The operation $\circ : A \times A \to A$ is called **idempotent**, if every element $a \in A$ is idempotent with respect to $\circ$.

[sTeX/MathTutorial]props/Distributive.en.tex

**Definition 3.4.3.** Let $\odot : B \times A \to A$ and $\oplus : A \times A \to A$. We say $\odot$ **distributes over** $\oplus$, if $b \odot (a_1 \oplus a_2) = (b \odot a_1) \oplus (b \odot a_2)$ for all $a_1, a_2 \in A$ and $b \in B$.

[sTeX/MathTutorial]props/Absorption.en.tex

**Definition 3.4.4.** Let $\odot : A \times B \to A$ and $\oplus : A \times B \to B$. We say $\odot$ **absorbs** $\oplus$, if $a_1 \odot (a_1 \oplus b) = a_1$ for all $a_1 \in A$ and $b \in B$.

[sTeX/MathTutorial]algebra/Band.en.tex

**Definition 3.4.5.** A **band** is an idempotent semigroup.

[sTeX/MathTutorial]algebra/Semilattice.en.tex

**Definition 3.4.6.** A **semilattice** is a commutative band.

[sTeX/MathTutorial]props/Reflexive.en.tex

**Definition 3.4.7.** A binary relation $R$ on $A$ is called **reflexive**, if $R(a, a)$ for all $a \in A$ .

[sTeX/MathTutorial]props/Symmetric.en.tex

**Definition 3.4.8.** A binary relation $R$ on $A$ is called **symmetric**, if $R(a, b)$ implies $R(b, a)$ for all $a, b \in A$.

[sTeX/MathTutorial]props/Transitive.en.tex

**Definition 3.4.9.** A binary relation $R$ on $A$ is called **transitive**, if $R(a, b)$ and $R(b, c)$ implies $R(a, c)$ for all $a, b, c \in A$.

[sTeX/MathTutorial]props/Antisymmetric.en.tex

**Definition 3.4.10.** A binary relation $R$ on $A$ is called **antisymmetric**, if $R(a,b)$ and $R(b,a)$ implies $a = b$ for all $a, b \in A$.

[sTeX/MathTutorial]orders/Graph.en.tex

**Definition 3.4.11.** A **directed graph** is a structure $\langle U, R\rangle$, where $U$ is a collection and $R$ a binary relation on $U$.

**Definition 3.4.12.** An **(undirected) graph** is a directed graph $\langle U, R\rangle$, where $R$ is symmetric.

[sTeX/MathTutorial]orders/Preorder.en.tex

**Definition 3.4.13.** A structure $\langle U, \leq\rangle$ is called a **preorder** (or **quasiorder**, or **preordered set**; in short **proset**), if $\leq$ is reflexive and transitive.

[sTeX/MathTutorial]orders/Poset.en.tex

**Definition 3.4.14.** A preorder $\langle U, R\rangle$ is called a **partial order** (or **poset**), if $R$ is antisymmetric.

[sTeX/MathTutorial]orders/InfSup.en.tex

**Definition 3.4.15.** Let $\langle U, R\rangle$ a poset. An element $a \in U$ is called an **infimum** or **greatest lower bound** of $x_1$ and $x_2$, if $a\ R\ x_1$, $a\ R\ x_2$, and for any $x$ with $x\ R\ x_1$ and $x\ R\ x_2$, we have $x\ R\ a$.

**Definition 3.4.16.** Let $\langle U, R\rangle$ a poset. An element $a \in U$ is called a **supremum** or **least upper bound** of $x_1$ and $x_2$, if $x_1\ R\ a$, $x_2\ R\ a$, and for any $x$ with $x_1\ R\ x$ and $x_2\ R\ x$, we have $a\ R\ x$.

Note that infima and suprema are more generally defined on *sets* of elements. Doing so in sTeX is significantly more complicated *for now*, and will require some amount of research to make convenient – especially if we want to subsequently define *operators* on pairs of elements, as below. We therefore opt for the simpler version where it is defined as binary from the get go.

[sTeX/MathTutorial]orders/MeetJoinSemilattice.en.tex

**Definition 3.4.17.** A poset $\langle U, R\rangle$ is called a **meet semilattice** if for every two elements $a, b$ the infimum $a \wedge b$ exists.

**Definition 3.4.18.** A poset $\langle U, R\rangle$ is called a **join semilattice** if for every two elements $a, b$ the supremum $a \vee b$ exists.

**Definition 3.4.19.** An **(order) semilattice** is a meet and join semilattice.

**Exercise**
Try to implement all of the above yourself!

### 3.4.1 Glueing Structures Together

We now want to progress towards lattices, i.e. the following:

**Definition 3.4.20.** A lattice is a structure $\langle U, \wedge, \vee\rangle$ such that $\langle U, \wedge\rangle$ and $\langle U, \vee\rangle$ are semilattices, and $\vee$ absorbs $\wedge$ and vice versa; i.e. $a \vee (a \wedge b) = a$ and $a \wedge (a \vee b) = a$.
The operations $\wedge$ and $\vee$ are called **meet** and **join**, respectively.

So we make a new module, open an `extstructure` environment and... realize two problems:

1. We can't just extend semilattice: We need *two* copies of semilattice that share a universe, and importing semilattice twice is of course redundant.
2. We also want to *rename* the operations of the two semilattices to be subsequently called `join` and `meet`.

What we need is a way to *inherit* from semilattice while a) *modifying* the symbols therein, and b) not be idempotent – i.e. two imports from the same structure or module should not be identified. We can do that with the `\copymod` macro, which takes three arguments:

1. A *name* for the copy,
2. the structure or module to copy, and
3. a comma-separated list of renamings and redefinitions of the symbol. ⟨*symbol*⟩=⟨*def*⟩ redefines ⟨*symbol*⟩, ⟨*symbol*⟩@⟨*newname*⟩ renames it, ⟨*symbol*⟩=⟨*def*⟩@⟨*newname*⟩ (or ⟨*symbol*⟩@⟨*newname*⟩=⟨*def*⟩) does both.

In our case, we want two copies of semilattice, which we will call `meetsl` and `joinsl`. In the first copy, we only want to rename `op` to `meet`. In the second, we want to rename `op` to `join`, and *also* redefine the universe to be the one from `meetsl`:

```
\copymod{meetsl}{semilattice}{
  op @ meet
}
\copymod{joinsl}{semilattice}{
  univ = \univ,
  op @ join
}
```

You might have already noticed some problem with that – which of the two universes does `\univ` refer to now? (They are *defined* as equal, but LaTeX does not know that!) Or which of the two `commutative axioms` does "`commutative axiom`" refer to now? Everything is ambiguous now!

Not really - if you have wondered why the `\copymod` takes a *name* as argument: The name is prefixed to every symbol name. Hence, the `universe` in `joinsl` is now called `joinsl/universe`, and the one in `meetsl` is called `meetsl/universe`. Furthermore, `\copymod` by default generates no semantic macros for any of the imported symbols – except for those renamed with `@`. In fact, what the `@` syntax actually does, is to generated a semantic macro by that name. If we want to change the *name* (that is shown when using `\symname` et al), we add that new name in square brackets. Hence, what we really want to do is:

```
\copymod{meetsl}{semilattice}{
  univ @ univ,
  op @ [meet]meet
}
\copymod{joinsl}{semilattice}{
  univ = \univ,
  op @ [join]join
}
```

This now gives us two copies of semilattice, generates semantic macros `\univ` for `meetsl/universe`, `\meet` for `meetsl/op` and `\join` for `joinsl/op`, and renames `meetsl/op` to `meet` and `joinsl/op` to `join`.

That allows us to then add the absorption axioms, an `sdefinition` for lattice and subsequently `$\lattice!$` produces $\langle U, \wedge, \vee\rangle$, with all axioms inherited (see `[sTeX/MathTutorial]algebra/Lattice.en.tex`).

### 3.4.2 Realizations

A very common situation we find in connection with mathematical structures is that "every *this* is a *that*" (or the conrete case "*this* is a *that*").

With what we did so far, we are in this situation regarding the algebraic definition of semilattices and the order-theoretic one (exemplary meet semilattice).

In MMT parlance, this corresponds to a total (implicit) theory morphism from "that" to "this".

In sTeX words, we want to inherit from "that" by assigning all the symbols in "that" to concrete terms. In our case:

`[sTeX/MathTutorial]algebra/SemiLatticeOrder.en.tex`

**Definition 3.4.21.** Let $\langle U, \circ\rangle$ a semilattice. We let $a\ R\ b$ iff $a \circ b = a$.

**Satz 3.4.22.** $\langle U, R\rangle$ *is a meet semilattice.*

*Proof:*
We need to prove the following

1. reflexivity ($a \ R \ a$):

   We need to show $a \circ a = a$. Follows from the idempotent axiom.

2. antisymmetry ($a \ R \ b$ and $b \ R \ a$ implies $a = b$ ):

   Assume $a \circ b = a$ and $b \circ a = b = a \circ b$ (by the commutative axiom). Hence, $a = b$

3. transitivity (If $a \ R \ b$ and $b \ R \ c$, then $a \ R \ c$. ):

   Assume $a \circ b = a$ and $b \circ c = b$ Then $a \circ c = (a \circ b) \circ c = a \circ (b \circ c) = a \circ b = a$. Hence, $a \ R \ c$.

4. $a \circ b$ is the infimum of $\{a, b\}$:

   By definition (and the commutative axiom), $a \circ b \ R \ a$ and $a \circ b \ R \ b$. We need to show, that if $x \ R \ a$ and $x \ R \ b$, then $x \ R \ a \circ b$. Assume $x \circ a = x$ and $x \circ b = x$. Then $x \circ (a \circ b) = (x \circ a) \circ b = x \circ b = x$. Hence $x \ R \ a \circ b$

□

So to be precise, we want to provide *definientia* for all undefined symbols in meet semilattice (i.e. the relation and meet) and *proofs* for all *axioms* (reflexive axiom, antisymmetric axiom, transitive axiom, and infimum axiom), and by so obtain the fact that every semilattice is a meet semilattice.

For that purpose, we have the `\realize` macro. It behaves like `\copymod`, but does not take a name, and additionally requires that all undefined fields get assigned. So we could do the following:

**Example 15**
Input:

File [sTeX/MathTutorial]`algebra/SemiLatticeOrder1.en.tex`

```
8  \begin{extstructure*}{semilattice}
9    \interpretmod{meetsl}{meetsl}{
10     univ = \univ,
11     meet @ [meet]meet = \op!,
12     rel @ [order]order = \map{a,b}{\eq{\op{a,b},a}},
13     reflexive axiom = trivial,
14     transitive axiom = trivial,
15     antisymmetric axiom = trivial,
16     infimum axiom = trivial
17   }
18 \end{extstructure*}
19
20 \vardef{mysl}[return=\semilattice]{S}
21 $\mysl{order}{a,b} \qquad \mysl{}[univ,op,order]$
```

Output:

$a \leq_S b$ $\langle U_S, \circ_S, \leq_S \rangle$

As we can see, we can now access the field `order`, which is renamed from `relation` in meet semilattice and also has the desired definiens in MMT. But of course this approach is very "declarative": We do all the assigning in one macro, rather than narratively as what they *should* be: definitions and proofs.

If we want to achieve the more narrative version at the beginning of the subsection, we can use the `realization` environment instead. It behaves like the `\realize` macro, but allows us to do the assignments and renamings individually somewhere in the body of the environment, interleaved with arbitrary text. Additionally, within the environment, all STEX features that introduce *definientia* (like the `\definiens` macro) induce assignments instead.

To declaratively rename or assign fields, we can then use the `\assign` and `\renamedecl` macros instead. That allows us to do the following instead:

```
\begin{realization}{meetsl}
  \assign{univ}{\univ}
  \assign{meet}{\op!}
  \renamedecl{rel}[order]{order}
  ...
```

...and then use text to do the remaining assignments. For example, we can use the `sdefinition` environment to assign `rel` to the desired definiens:

```
\usestructure{meetsl}
\begin{sdefinition}[for=order]
  \varbind{va,vb}
  Let $\semilattice![univ,op]$ a \sn{semilattice}.
  We let $\rel{\va,\vb}$
  iff $\definiens{\eq{\op{\va,\vb},\va}}$.
\end{sdefinition}
```

And now STEX will use the `\definiens` to assign $a, b \mapsto a \circ b = a$ to the relation of meet semilattice.

Analogously, we can use the `sproof` and `subproof` environments to produce "definientia" (i.e. proofs) for the axioms (see `[sTeX/MathTutorial]algebra/SemiLatticeOrder.en.tex`)

# Chapter 4

# Extensions for Education

The last two chapters have shown generic markup and semantization facilities in sTeX. As said before, investments in semantic markup pay off, iff the impact of a document is high, e.g. if there are many more readers than authors or if the semantic services afforded by the semantic markup can help reduce the help readers need to understand the material.

Educational documents constitute one category of high-impact documents which are supported by the sTeX ecosystem, we will cover them here. In fact, educational documents have been one of the initial document categories sTeX has been developed for. The idea is that if we can mark up the meaning and didactic role of learning objects, we can base learning support services on that and embed them into the documents.

Another reason educational documents are particularly interesting is that in a sense all academic communication is educational, as all documents try to "teach" the reader new concepts and results.

Concretely, we cover a document class for combining slides and course notes (section 4.1) and functionality for marking up problems and excercises (section 4.2) and for marking up homework assignments and exams (section 4.3).

## 4.1 Slides and Course Notes

### 4.1.1 Introduction

The notesslides document class is derived from `beamer.cls` [**beamerclass:on**], it adds a "notes version" for course notes that is more suited to printing than the one supplied by `beamer.cls`.

The notesslides class takes the notion of a slide frame from Till Tantau's excellent beamer class and adapts its notion of frames for use in sTeX. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the notesslides package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study.

To achieve this, the notesslides class has two modes: *slides mode* and *notes mode* which are determined by the package option.

### 4.1.2 Package Options

The notesslides class takes a variety of class options:

`slides` `notes` The options `slides` and `notes` switch between slides mode and notes mode (see subsection 10.1.3).

`sectocframes` `topsect` If the option `sectocframes` is given, then for the `sfragment`s, special frames with the `sfragment` title (and number) are generated. The `topsect` allows to specify the top section level (e.g. `chapter`, `section`, ...) for documents that are formally stand-alone, but intended to be chapters, sections, or ....

EdN:1 [1]

`frameimages` `fiboxed` If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames (see ???). If also the `fiboxed` option is given, the slides are surrounded by a box.

### 4.1.3 Notes and Slides

`frame` (*env.*) Slides are represented with the `frame` environment just like in the beamer class, see [**Tantau:ugbc**] for details.

`note` (*env.*) The notesslides class adds the `note` environment for encapsulating the course note fragments. Content of this environment is only shown in *notes mode*.

By interleaving the `frame` and `note` environments, we can build course notes as shown here:

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

[1] EdNote: MK: maybe this option should be used in the other sTEX classes as well. Actually I think this already is treated by stex.sty; where to document?

\ifnotes Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notestrue` or `\notesfalse` is strongly discouraged however.

We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

The beamer class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

\inputref* If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

nparagraph (*env.*) nparagraph (*env.*) ndefinition (*env.*) nexample (*env.*) nsproof (*env.*) nassertion (*env.*) There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nparagraph` environment is just an `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

### 4.1.4 Managing and Customizing Themes

As the `notesslides` package is based on the `beamer` class, it can in principle (in slides mode) take advantage of all `beamer` themes. Themes can be specified/loaded with the normal `\usetheme`, but the `notesslides` package provides the `\libusetheme` macro:

\libusetheme In analogy to the `\libinput` functionality of the `stex` package, this macro allows loading `beamer` themes from the `lib` directories leading up to the current archive.

Unfortunately, there is no `beamer` functionality of using a themed `frame` environment in `article` mode. Therefore `notesslides` has to hand-craft one – for a very limited set of themes. The notesslides package and class comes with a simple default theme named sTeX that provided by the beamterthemesTeX.sty style file. It is assumed as the default theme for sTeX-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide height is variable) is

The footer line can be customized. In particular the logos.

\setslidelogo The default logo provided by the notesslides package is the STEX logo it can be customized using \setslidelogo{⟨logo name⟩}.

\setsource The default footer line of the notesslides package mentions copyright and licensing. In notesslides \source stores the author's name as the copyright holder. By default it is the author's name as defined in the \author macro in the preamble. \setsource{⟨name⟩} can change the writer's name.

\setlicensing For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package hyperref is loaded, then we can attach a hyperlink to the license logo. \setlicensing[⟨url⟩]{⟨logo name⟩} is used for customization, where ⟨url⟩ is optional.

These macros can be used set up a institution-specific theme; for instance the one for FAU simply customizes the logo:

```
% Beamer slide theme for FAU;
\typeout{Beamer FAU theme}
\RequirePackage{beamerthemesTeX}
\setslidelogo[courses/FAU/admin]{PIC/FAU_Logo_Bildmarke.png}
```

This could be placed in one of the lib folders available to the current archive and could then be activated with

```
\documentclass[slides]{notesslides}
\libusetheme{FAU}
```

in the preamble of the main noteslides file.

### 4.1.5 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add STEX notes.

\frameimage \mhframeimage In this case we can use \frameimage[⟨opt⟩]{⟨path⟩}, where ⟨opt⟩ are the options of \includegraphics from the graphicx package [**CarRah:tpp99**] and ⟨path⟩ is the file path (extension can be left off like in \includegraphics). We have added the label key that allows to give a frame label that can be referenced like a regular beamer frame.

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

```
\frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

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

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

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

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

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

`\textwarning` The `\textwarning` macro generates a warning sign: ⚠

### 4.1.6 Ending Documents Prematurely

`\prematurestop` `\afterprematurestop` For prematurely stopping the formatting of a document, SiTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `sfragment` environments as needed. After that – and before the implicit `\end{document}` it calls the internal `\afterprematurestop`, which can be customized to do additional cleanup or e.g. print the bibliography.

`\prematurestop` is useful when one has a driver file, e.g. for a course taught multiple years and wants to generate course notes up to the current point in the lecture. Instead of commenting out the remaining parts, one can just move the `\prematurestop` macro. This is especially useful, if we need the rest of the file for processing, e.g. to generate a theory graph of the whole course with the already-covered parts marked up as an overview over the progress; see `import_graph.py` from the `lmhtools` utilities [**lmhtools:github:on**].

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the SiTeX preamble of the course notes file.

### 4.1.7 Global Document Variables

To make document fragments more reusable, we sometimes want to make the content depend on the context. We use **document variables** for that.

`\setSGvar` `\useSGvar` `\setSGvar{`⟨*vname*⟩`}{`⟨*text*⟩`}` to set the global variable ⟨*vname*⟩ to ⟨*text*⟩ and `\useSGvar{`⟨*vname*⟩`}` to reference it.

`\ifSGvar` With`\ifSGvar` we can test for the contents of a global variable: the macro call `\ifSGvar{`⟨*vname*⟩`}{`⟨*val*⟩`}{`⟨*ctext*⟩`}` tests the content of the global variable ⟨*vname*⟩, only if (after expansion) it is equal to ⟨*val*⟩, the conditional text ⟨*ctext*⟩ is formatted.

### 4.1.8 Excursions

In course notes, we sometimes want to point to an "excursion" – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

```
\excursion{founif}{../fragments/founif.en}
  {We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

It generates a paragraph that references the excursion whose source is in the file `../fragments/founif.en.tex` and automatically books the file for the `\printexcursions` command that is used here to put it into the appendix. We will look at the mechanics now.

`\excursion`

The `\excursion{`⟨*ref*⟩`}{`⟨*path*⟩`}{`⟨*text*⟩`}` is syntactic sugar for

```
\begin{nparagraph}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nparagraph}
```

`\activateexcursion`
`\printexcursion`
`\excursionref`

Here `\activateexcursion{`⟨*path*⟩`}` augments the `\printexcursions` macro by a call `\inputref{`⟨*path*⟩`}`. In this way, the `\printexcursions` macro (usually in the appendix) will collect up all excursions that are specified in the main text.

Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{`⟨*label*⟩`}` for that.

`\excursiongroup`

Finally, we usually want to put the excursions into an `sfragment` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro: `\excursiongroup[id=`⟨*id*⟩`,intro=`⟨*path*⟩`]` is equivalent to

```
\begin{note}
\begin{sfragment}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{sfragment}
\end{note}
```

# Contents

## 4.2 Problems and Exercises

### 4.2.1 Background

Problems/exercises are text fragments that contain a task assigned to the learner – e.g. computing the value of a specified quantity, simplifying an expression, modeling a described situation in a mathematical structure, or judging the veracity of a given statement. Problems/exercises are used in very different contexts, in

- *homework assignments and formal exams*, where they are graded and points are awarded for course credit,

- *self-study materials* that allow learners explore applications of some concepts and/or practice,
- *automated tutoring systems* to determine learner competencies to trigger computer supported learning services.

In all of these contexts, diagnosis the correctness or suitability of an answer plays a great role, e.g. for grading, the generation of feedback, or the suggestion of remedial actions that may "heal" any diagnosed competency gaps or misconceptions. To support that we might want to specify "answer classes" that classify answers received for automating/triggering feedback and grading.

There are various types of problems/exercises in practical use. They are mainly distinguished by how they support diagnosis of student answers: there are

- open problems, where expected answers are free-form, and classification into answer classes is usually a manual process; see subsection 4.2.3
- single/multiple choice questions where the answer classes and thus feedback/grading correspond to the selection pattern of items; see subsection 4.2.7.
- fill-in-the-blanks questions where we may want to specify how the answer string is interpreted; see subsection 4.2.8.

Additionally, problems and exercises often come with text fragments that serve auxiliary functions: hints, notes, and and grading specifications. Furthermore, we can specify how long solving a given problem is estimated to take and how many points will be awarded for a perfect solution – e.g. in an exam. See subsection 4.2.10 for details.

The `problem` package gives semantic markup support for all these aspects of problems/exercises with a focus on problem libraries that collect carefully formulated and tested problems/exercises, so that they can be re-used in many contexts. It also tries to support all possible stakeholders in dealing with problems/exercises:

- *learners*, who try to solve problems and may profit from feedback
- *instructors*, who pose problems in homeworks, quiz, and exams,
- *graders*, who try to assess learner's competencies from the answers given
- *authors* of learning objects and (automated) *course generators*, who may use problems to diagnose learner's competencies (e.g. as prerequisites).

### 4.2.2 The Package, Options, and Configuration

The problem package provides functionality for marking up problems and exercises as semantic sources from which the presentations for the various contexts can be generated. E.g. documents without solutions for paper or online exams, and the corresponding exams with master solutions for exam reviews. Similarly with/without hints, or points. Their visibility is specified in the options of the problem package, which can be used in any LaTeX class. The following is a typical preamble for a problem file:

```
\documentclass[lang=de]{article}
\usepackage[solutions,hints,pts,min]{problem}
```

Here we have specified the options `solutions` (solutions should be shown), `hints` (hints should be given), `pts` (display the points awarded for solving the problem?), `min` (display the estimated minutes for problem solving). Leaving out the options would make the corresponding functionality invisible.

EdN:2

The `problem` package generates content and labels according to the language specified in the `lang` key of the main document[2], these can be localized by in the files `ldf/problem-<lang>.ldf`[6].

The `problem` package also supplies the `test` option, which specifies that the content is intended for use in an assessment situation, where certain additional content (e.g. exam legalse) should be shown while other content (e.g. solutions) should not be.

Note that documents (or document fragments) with problems are often formatted in different versions that can be configured by these options. Therefore the following variant of the preamble prefix above can be very useful:

```
\documentclass[lang=de]{article}
\providecommand\probopts{,solutions,min}
\usepackage[hints,pts\probopts]{problem}
```

We add a level of indirection via the `\probopts` macro (`\providecommand` sets it unless it already exists). If the current file is `foo.tex`, then we can make a LaTeX file `foo-test.tex` that formats in test mode as

```
\newcommand\probopts{,test}\input{foo}
```

Alternatively, we do not need a file at all: we can just (e.g. in a Makefile or a PDF generation script) issue the following command in the terminal:

```
pdflatex "\def\probopts{,test}\input{foo}"
```

### 4.2.3 (Open) Problems

The main environment provided by the problem package is (surprise surprise) the `sproblem` environment. It is used to mark up problems and exercises. The following example shows the main functionality:

**Example 16**
Input:

[2] EdNote: MK: document the attribute somewhere

[6] The current crop of language definition files is quite limited. Please feel free to contribute the to the localization effort

File `tutorial/ext/simple-problem.en.tex`

```
\begin{document}
\begin{sproblem}[id=prob.elefants,pts=10,min=2,title=Fitting Elefants]
  How many Elefants can you fit into a Volkswagen beetle?
  \begin{hint}
    Think positively, this is simple!
  \end{hint}
  \begin{exnote}
    Justify your answer
  \end{exnote}
  \begin{gnote}[title=deduct 5pts if justification is missing]
    \anscls[feedback=you forget that elephants could be small]{none}
    \anscls[feedback=you forget the back seats]{2}
  \end{gnote}
  \begin{solution}
    Four, two in the front seats, and two in the back.
```

Output:

**Exercise (Fitting Elefants)**
How many Elefants can you fit into a Volkswagen beetle?

*Lösung:* Four, two in the front seats, and two in the back.

The `sproblem` environment takes an optional key/value argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem.

The `sproblem` can contain several `solution` environments, which take the well-known `id`, `style`, and `title` attributes, and also the `testspace` and `answerclass`. The former

The additional functionality is specified in the `hint exnote` (notes in exercises), `solution`, and `gnote` (grading notes) environments. Here, the first three are shown whereas the grading notes are hidden, since the corresponding option was not given in the `\usepackage[...]{problem}`. All of these environments can occur any number of times in the `sproblem` environment. The `solution` environment takes an optional argument that is interpreted as the identifier.

### 4.2.4 Solutions and Answer Classes

Most problem libraries also contain (master) solutions, which show the intended way of solving a problem. The `solution` environment allows to specify solutions. It is only formatted if the `solutions` options is specified for the `problem` package.

We observe that many problems have more than one possible solution, so the `problem` package allows multiple `solution` environments; they can be distinguished by an `id` key in the optional first argument.

We also observe that in problem libraries we may want to keep track of classes of answers that have been observed e.g. in previous assessment situations, e.g. to provide

feedback. In this setting, the `solution` environment can be used to encode (typical representative of) **answer classes** (see also ??? below).

`solution` The `solution` environment takes an optional argument that allows the keys `id`, `title` and `style` keys that we have already seen above, further more

1. `testspace=`, which is equivalent to a `\testpsace{}` (see subsection 4.2.10).
2. `answerclass=`, which gives the name of an answer class, this solution corresponds to.[3]

EdN:3

### 4.2.5 Grading Support

When problems are graded by multiple persons (e.g. for large university exams), then consistency of grading is a major issue. Both within a cohort and between cohorts e.g. in successive instances of a course.

Therefore keeping records of how to grade a problem (which problem is worth how many "points" and where to deduct how many points in which situations) and making them accessible to graders in a "master solution" – a version of an exam formatted to have the known solutions/answer classes (see above) and the grading specifications is a good practice. And it makes sense to include grading information into a problem library.

The simplest and arguably most effective measure is to specify the points that can be reached for a problem in the problem itself via the 'pts=' key (but see subsection 4.2.9)

The `problem` package also supplies the `gnotes` environment, which is only formatted when the `gnotes=` key is given. This environment takes an optional argument that allows the `id`, `style`, and `title` keys. The latter is used to give a short version of the grading specification. More structured grading support can be expressed in `\anscls` markup which we discuss next.

We have already encountered the notion of answer classes – classes of answers that are supposed equivalent in terms of learner's competencies inscribed in them – above. These are the natural carriers of both grading feedback and points.

In the `gnote` environment we use the `\anscls` macro for specifying such information using meta-level descriptions rather than typical representatives. It takes an optional keyword argument for the grading/feedback information and a required one for the answer class description.

We distinguish two usages of `\anscls[...]{<desc>}`: in

- *answer classes* `<desc>` is a specification of a class of answers and the `pts=` key the points to be awarded to that.
- *answer traits* `<desc>` specifies the deviation from an assumed correct solution, and the value of the `pts=` key specifies changes to the points specified in the problem itself.

Consider for instance the following situation:

```
\begin{sproblem}[pts=3]
  Give an example for a foo and explain it in one sentence.
  \begin{gnote}[title=1 pt for the example and 2 for the explanation]
    \anscls[pts=0,feedback=I did not understand this]{complete gibberish}
    \anscls[pts=-2,feedback=You forgot the explanation]{no explanation}
  \end{gnote}
\end{sproblem}
```

[3] EdNote: This mechanism needs to be further worked our or deprecated; if we keep it we probably need to add pts and feedback attributes for parity with \anscls.

Here we have a problem that is worth three points, and the grading note explains how to treat deviations from a correct solutions – which is not given, since there may be hundreds of examples for foos. The `\anscls` specify this out for grading support systems[7]. The first is an answer class description for the class of answers that cannot be made sense of by the graders and specifies a default grade and feedback. The second specifies the trait of a missing explanation and specifies a deduction commensurate with the grading note title and a feedback to the learner. Note that a grading support system can distinguish answer classes from traits by the form of the value of points: absolute values vs. differences like `+5` or `-3`.

### 4.2.6 Structured Problems

Problems can be structured into subproblems via the `subproblem` environment. It takes the same arguments and allows the same content model as as `sproblem`.

**Example 17**

Input:

File `tutorial/ext/structured-problem.en.tex`

```
\begin{document}
\begin{sproblem}[id=prob.elefants-mod,title=Fitting Elefants Modularly]
  Consider the problem of fitting elephants into a Volkswagen beetle.
  \begin{subproblem}[pts=1,min=2]
    Estimate the number of elephants on the front seats.
    \begin{solution}
      Two on each side one.
    \end{solution}
  \end{subproblem}
  \begin{subproblem}[pts=1,min=2]
    Estimate the number of elephants on the back seats.
    \begin{solution}
      Three little elephants.
    \end{solution}
  \end{subproblem}
```

Output:

[7] like e.g. `https://gitos.rrze.fau.de/yn06uhoc/vollkorn-prototype`

**Exercise (Fitting Elefants Modularly)**
Consider the problem of fitting elephants into a Volkswagen beetle.

1. Estimate the number of elephants on the front seats.

   *Lösung:* Two on each side one.

2. Estimate the number of elephants on the back seats.

   *Lösung:* Three little elephants.

3. What is the total number?

   *Lösung:* A family of five; we do not want elephants in the frunk.

By default, subproblems are numbered subordinate to the "mother problem". The `problem` package does not make any assumptions about whether subproblems can be used independently from their sister problems, or on order requirements.

Note that neither `sproblem` nor `subproblem` can be nested, if you want to have further structure, you need to resort to regular LaTeX functionality. Note that you can add `solution` and `gnote` environments wherever you want, so that this need not force you to forego structure.

The `subproblem` environment is however very useful for modularizing problems: `subproblem` environments can appear outside of `sproblem` and in particular at the top-level of a file. This allows them to be shared between `sproblem` via `\inputref`.

### 4.2.7 Single/Multiple Choice Blocks

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc` macro.

`\mcc` `\mcc[`⟨*keyvals*⟩`]{`⟨*text*⟩`}` takes an optional key/value argument ⟨*keyvals*⟩ for choice metadata and a required argument ⟨*text*⟩ for the proposed answer text. The following keys are supported

- `T` for correct answers, `F` (the default value) for false ones,
- `Ttext` the verdict for correct answers, `Ftext` for false ones, and
- `feedback` for a short feedback text given to the student.

What we see when this is formatted to PDF depends on the context. In solutions mode (we start the solutions in the code fragment below) we get

**Example 18**
Input:

```
\startsolutions
\begin{sproblem}[title=Functions,id=functions1]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def} \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}
```

Output:

**Exercise (Functions)**
What is the keyword to introduce a function definition in python?

- ☑ def
- ☐ function[a]
- ☐ fun[b]
- ☐ public static void[c]

---

[a] *that is for C and C++*

[b] *that is for Standard ML*

[c] Noooooooooo
*that is for Java*

In "exam mode" where disable solutions (here via `\stopsolutions`) we get the questions without solutions (that is what the students see during the exam/quiz).

**Example 19**
Input:

```
\stopsolutions
\begin{sproblem}[title=Functions,id=functions1]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def} \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}
```

Output:

**Exercise (Functions)**
What is the keyword to introduce a function definition in python?
- ☐ def
- ☐ function
- ☐ fun
- ☐ public static void

What we have seen is the default rendering of the multiple choice block, which is as an itemize like environment with check boxes. There are alternatives to that which we can choose with the `style` key in the optional attribute of the `mcb` environment. Currently the only alternative is `inline` style:

**Example 20**
Input:

```
\startsolutions
\begin{sproblem}[title=Functions,id=functions1]
  What is the keyword to introduce a function definition in python?

  \begin{mcb}[style=inline]
    \mcc[T]{def} \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Nooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}
```

Output:

**Exercise (Functions)**
What is the keyword to introduce a function definition in python?
[ ☑ def ☐ function[a] ☐ fun[b] ☐ public static void[c] ]

[a] *that is for C and C++*
[b] *that is for Standard ML*
[c] Nooooooooo *that is for Java*

This is the solutions mode, i.e. with solutions, otherwise, the footnotes will fully disappear.

Analogously, single choice blocks are marked up by the `scb` environment and the individual choices by the `\scc` macro. In PDF, there is little difference to the multiple choice case. In interactive formats like HTML, single choice blocks are typically realized via radio buttons to enforce single choice.[4] EdN:4

Often we only want to ask whether a statement is true or false. This is essentially a single choice block. STEX provides the macros `\yesTnoF`, `\yesFnoT`, `\trueTfalseF`, and `\trueFfalseT` for that. They are used as in the example below:

[4] EdNote: MK: are there any differences between mcc and scc we need to discuss here?

**Example 21**
Input:

```
\begin{sproblem}\startsolutions
  Mark the following statements as true or false:
  \begin{enumerate}
  \item The moon is made of green cheese \trueFfalseT
  \item \ldots
  \end{enumerate}
\end{sproblem}
```

Output:

**Exercise**
Mark the following statements as true or false:

1. The moon is made of green cheese [◯ true ✓ false]
2. ...

### 4.2.8 Filling-In Concrete Solutions

The next simplest situation, where we can implement auto-grading is the case where we have fill-in-the-blanks

The `\fillinsol` macro takes a single argument, which contains a concrete solution (i.e. a number, a string, ...), which generates a fill-in-box in test mode:

**Example 22**
Input:

```
\stopsolutions
\begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
  How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
\end{sproblem}
```

Output:

**Exercise (Fitting Elefants)**

How many Elefants can you fit into a Volkswagen beetle? ☐

and the actual solution in solutions mode:

**Example 23**
Input:

```
\startsolutions
\begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
  How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
\end{sproblem}
```

Output:

**Exercise (Fitting Elefants)**

How many Elefants can you fit into a Volkswagen beetle? 4

If we do not want to leak information about the solution by the size of the blank we can also give `\fillinsol` an optional argument with a size: `\fillinsol[3cm]{12}` makes a box three cm wide.

Obviously, the required argument of `\fillinsol` can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings "soft comparisons" might be in order. Indeed the default matching of answers against the string provided in the required argument of the `\fillinsol` is up to whitespace normalization.

The `problem` package allows to specify different behavior via three additional keys whose arguments are all triples of the form

`{⟨target⟩}{⟨TF⟩}{⟨feedback⟩}`

where ⟨*target*⟩ is the target specification, ⟨*TF*⟩ is the verdict `T` or `F` on the correctness of an answer that is accepted by ⟨*target*⟩, and ⟨*feedback*⟩ a feedback string. In the following (somewhat contrived) example, we see all three at work:

**Example 24**

Input:

```
\begin{sproblem}
  In 2019, Erlangen is a city of
  \fillinsol[testspace=3cm,
  exact={111962}T{Wow, you known your numbers, according to
    Wikipedia that is exactly correct},
  numrange={0-1000}F{No,that would be a village},
  numrange={1001-100000}F{No,that would be a town},
  numrange={100000-120000}T{Yes, it is close to 109000},
  numrange={1200001-}F{No, that is too large},
  regex={-.[0-9]+}F{What do negative Inhabitants even mean?},
  regex={.*}F{Please give a natural number}]
  {111962}
  inhabitants.
\end{sproblem}
```

Output:

**Exercise**

In 2019, Erlangen is a city of 111962[a] inhabitants.

[a]

| type | case | verdict | feedback |
|---|---|---|---|
| exact | 111962 | T | |
| exact | 111962 | T | Wow, you known your numbers, according to Wikipedia that is exactly correct |
| numrange | 0-1000 | F | No,that would be a village |
| numrange | 1001-100000 | F | No,that would be a town |
| numrange | 100000-120000 | T | Yes, it is close to 109000 |
| numrange | 1200001- | F | No, that is too large |
| regex | -.[0-9]+ | F | What do negative Inhabitants even mean? |
| regex | .* | F | Please give a natural number |

- `exact=` gives the exact string (no whitespace normalization).
- `numrange=` specifies a number range, i.e. a comma-separated list of pairs ⟨*low*⟩-⟨*high*⟩, where ⟨*low*⟩ and ⟨*high*⟩ are number representations or empty.
- `regex=` gives a POSIX regular expression that specifies all acceptable strings.

### 4.2.9 Including Problems

`\includeproblem`

The `\includeproblem` macro can be used to include a problem from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys `title`, `min`, and `pts` specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the `problem` environment in the included file.

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` options are set. This allows to give students hints about the estimated time and the points to be awarded.

### 4.2.10 Testing and Spacing

The `problem` package is often used by the `hwexam` package, which is used to create homework assignments and exams. Both of these have a "test mode" (invoked by the package option `test`), where certain information –master solutions or feedback – is not shown in the presentation.

`\testspace`
`\testsmallspace`
`\testsmallspace`
`\testsmallspace`
`\testnewpage`
`\testemptypage`

`\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: `\testsmallspace`, `\testsmallspace`, `\testsmallspace` give small (1cm), medium (2cm), and big (3cm) vertical space.

`\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

TODO[8]

## 4.3 Homework Assignments and Exams

### 4.3.1 Introduction

The hwexam package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the problem package. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

### 4.3.2 Package Options

`solutions` `notes` `hints` `gnotes` `pts` `min`

The hwexam package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the problems package (cf. its documentation for a description of the intended behavior).

`multiple`

Furthermore, the hwexam package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test`

Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the LaTeX source.

### 4.3.3 Assignments

`assignment` (*env.*) `number` `title` `type` `given` `due`

This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the `assignment` environment), `title` (for the assignment title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”, or “homework”), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

### 4.3.4 Including Assignments

`\includeassignment`

The `\includeassignment` macro can be used to input an assignment from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

[8] TODO: check what is still undescribed problem.sty and make examples for it.

### 4.3.5 Typesetting Exams

testheading (*env.*) The `\testheading` takes an optional keyword argument where the keys `duration` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.

duration
min
reqpts

```
\title{320101 General Computer Science (Fall 2010)}
\begin{testheading}[duration=one hour,min=60,reqpts=27]
  Good luck to all students!
\end{testheading}
```

Will result in

Name: Matriculation Number:

## 320101 General Computer Science (Fall 2010)

2026-06-25

**You have one hour (sharp) for the test**;
Write the solutions to the sheet.
The estimated time for solving this exam is 23 minutes, leaving you 37 minutes for revising your exam.
You can reach 23 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -4 points are bonus points.

Different problems test different skills and knowledge, so do not get stuck on one problem.

| | To be used for grading, do not write here | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| prob. | 4.1 | 1.1 | 1.2 | 2.1 | 2.2 | 2.3 | 2.4 | 2.5 | 2.6 |
| total | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 |
| reached | | | | | | | | | |

good luck

[9]

[9] MK: The first three "problems" come from the stex examples above, how do we get rid of this?

# Part II
# User Manual

*The dynamic HTML version of this part can be found at*
`https://mathhub.info/?a=sTeX/Documentation&d=manual&l=en`

# Chapter 5

# Basics

## 5.1 Package and Class Options

- `debug=`⟨*prefixes*⟩: (see Developer Manual)
- `lang=`⟨*languages*⟩: If set, sTeX will load the `babel` package with the provided languages. Supported languages (currently) are:

  | | |
  |---|---|
  | `ar` | arabic |
  | `bg` | bulgarian |
  | `de` | german (with option `ngerman`) |
  | `en` | english |
  | `fi` | finnish |
  | `fr` | french |
  | `ro` | romanian |
  | `ru` | russian |
  | `tr` | turkish (with option `shorthands=:!`) |

- `mathhub=`⟨*path*⟩: Uses the provided file path as `MathHub` directory (see Section 1 (Math Archives and the `MathHub` Directory) in the sTeX Manual).
- `usesms`/`writesms`: If `writesms` is set, content loaded from external math archives (i.e. modules) is persisted in the file `\jobname.sms`.

  If `usesms` is set, the content of the `.sms`-file is loaded, obviating the need to reprocess the original files.

  The options are not mutually exclusive, but care should be taken if dependencies have changed between builds.

  This offers two advantages:

  1. If a document has many (transitive) dependencies, `usesms` should significantly speed up the build process, and
  2. setting `usesms` allows for distributing the `.sms`-file to make the document *standalone*, allowing for compilation without needing imported/used modules to be present.

The options `debug`, `mathhub`, `usesms` and `writesms` can also be set by the environment variables `STEX_DEBUG`, `MATHHUB`, `STEX_USESMS` and `STEX_WRITESMS`. In fact, the `MATHHUB` environment variable is the recommended way to set the `MathHub` directory. This is the only option where the *package option* overrides the environment variable.

The environment variables for `USE/WRITESMS` are particularly useful, in that they allow for convenient compilation workflows. For example, the `Build PDF/XHTML/OMDoc`-button in the IDE does the following:

```
STEX_WRITESMS=true pdflatex <job>.tex
[bibtex|biber] <job>
STEX_USESMS=true pdflatex <job>.tex
STEX_USESMS=true pdflatex <job>.tex
```

Guaranteeing (in the first run) that all dependencies are loaded from their respective sources and persisted, and in the two subsequent runs read from the generated `.sms` file, (likely) speeding ub the subsequent runs significantly.

## 5.2 Math Archives and the `MathHub` Directory

sTeX uses math archives to organize document content modularly, without a user having to specify absolute paths, which would differ between users and machines.

All sTeX archives need to exist in the local `MathHub`-directory. sTeX knows where this folder is via one of five means:

1. If the sTeX package is loaded with the option `mathhub=/path/to/mathhub`, then sTeX will consider `/path/to/mathhub` as the local `MathHub` directory.
2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the sTeX-package is loaded, then this macro is assumed to point to the local `MathHub` directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the `MathHub` directory as `path/to/mathhub`.
3. Otherwise, sTeX will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local `MathHub` directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.
4. If that too fails, sTeX will look for a file `~/.stex/mathhub.path`. If this file exists, sTeX will assume that it contains the path to the local `MathHub`-directory. This method is recommended on systems where it is difficult to set environment variables, and is used by the IDE setup.
5. Finally, if all else fails, sTeX considers `~/MathHub` to be the `MathHub` directory.

The sTeX IDE allows you to directly download math archives from `gl.mathhub.info` – currently available archives there are:

- `sTeX/*` – a group of semi-experimental documents showcasing sTeX3.3 features,
- `smglom/*` – a vast collection of multilingual modules of concepts in mathematics and computer science. The SMGloM predates sTeX3 and is thus largely "underannotated" with respect to (formal) semantics,
- `courses/*` – a vast collection of lecture slides and notes in computer science for courses held by Michael Kohlhase. They largely make use of SMGloM modules.

### 5.2.1 The Structure of Math Archives

An archive `group/name` is stored in the directory `<MathHub>/group/name`; e.g. assuming your local `MathHub`-directory is set as `/user/foo/MathHub`, then for the `sTeX/Documentation` archive to be found by the sTeX system, it needs to be in `/user/foo/MathHub/sTeX/Documentation`.

Each such archive needs two subdirectories:

- `/source` – this is where all your tex files go.
- `/META-INF` – a directory containing a single file `MANIFEST.MF`, the content of which we will consider shortly.

An additional `lib`-directory is optional, and is discussed in section 5.3.

### 5.2.2 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF` directory consists of key-value-pairs, informing sTeX (and associated software, e.g. MMT) of various properties of an archive. For example, the `MANIFEST.MF` of the `sTeX/Documentation` archive looks like this:

```
id: sTeX/Documentation
ns: http://mathhub.info/sTeX/Documentation
narration-base: http://mathhub.info/sTeX/Documentation
format: stex
title: The sTeX Documentation
teaser: The full Documentation for the sTeX system
url-base: https://mathhub.info
dependencies:sTeX/ComputerScience/Software,sTeX/MathTutorial
ignore: */code/*|*/tikz/*|*/tutorial/solution/*
```

Many of these are in fact ignored by sTeX, but some are important:

`id`: The name of the archive, including its group (e.g. `sTeX/Document`). This is used by the MMT system in favor of the directory, but TeX's limited access to the file system enforces the directory structure.

`source-base` or

`ns`: The namespace from which all symbol and module MMT-URIs in this archive are formed.

`narration-base`: The namespace from which all document MMT-URI in this repository are formed. It can safely match the `ns`-field.

`url-base`: A URL that is formed as a basis for *external references*. and hyperlinks. An MMT (or comparable system) instance should run there and host (sTeX-generated) HTML.

`dependencies`: All archives that this archive depends on. sTeX ignores this field, but MMT can pick up on them to resolve dependencies, e.g. when downloading archives in the IDE, which will also download all dependencies.

`ignore`: A regular expression of `.tex` files in the `source` directory that should be ignored; e.g. they will not be compiled when building a whole directory or archive in the IDE.

## 5.3 The lib-Directory

A math archive group/archive may have a lib directory primarily intended for preamble code, packages, .bib files, etc., the files in which can be referenced in various ways.

Additionally, a *group* of archives group may have an additional archive group/meta-inf. If this meta-inf archive has a /lib-subdirectory, they too will be considered by the following.

\libinput

\libinput {some/file} searches for a file some/file[.tex] in

- the lib-directory of the current archive, and
- the lib-directory of a meta-inf-archive in (any of) the archive groups containing the current archive

and \inputs all found files in reverse order; e.g. \libinput{preamble} in a .tex-file in sTeX/Documentation will *first* input .../sTeX/meta-inf/lib/preamble.tex and then ../sTeX/Documentation/lib/preamble.tex.

\libinput will throw an error if *no* candidate for some/file is found.

\libinput[some/archive]{some/file} will do the same, but starting in the lib directory of the math archive some/archive.

\libusepackage

\libusepackage [package-options]{some/file} searches for a file some/file.sty in the same way that \libinput does, but will call \usepackage[package-options]{path/to/some/file} instead of \input.

\libusepackage throws an error if not *exactly one* candidate for some/file.sty is found.

\addmhbibresource

\addmhbibresource [some/archive]{some/file} searches for a file like some/file.bib in some/archive's lib directory and calls \addbibresource to the result.

\libusetikzlibrary

\libusetikzlibrary behaves like \libusepackage but looks specifically for tikz libraries and calls \usetikzlibrary on the results.

throws an error if not *exactly one* candidate for for the library is found.

A good practice is to have individual STEX fragments follow basically this document frame:

```
\documentclass{stex}
\libinput{preamble}
\setsectionlevel{<your preference>}
\begin{document}
    \IfInputref{}{
      ...
      \maketitle
      \ifstexhtml \else \tableofcontents \fi
    }
    ...
    \IfInputref{}{\libinput{postamble}}
\end{document}
```

Then the `preamble.tex` files can take care of loading the generally required packages, setting presentation customizations etc. (per archive or archive group or both), and a `postamble.tex` can e.g. print the bibliography, index etc.

`\libusepackage` is particularly useful in such a `preamble.tex` when we want to use custom packages that are not part of a TEX distribution, or on CTAN. In this case we commit the respective packages in one of the `lib` folders and use `\libusepackage` to load them.

## 5.4 Basic Macros

`\sTeX` `\stex`

The `\sTeX` macro produces this sTEX logo. It is provided by the stex-logo package, included with the stex package.

`\ifstexhtml`

The TEX conditional `\ifstexhtml` is *true* if the current compilation generates HTML, and *false* otherwise (i.e. generates PDF).

`\STEXinvisible` `\STEXinvisible{⟨code⟩}`

Processes ⟨*code*⟩, but does not generate any output. In the HTML, ⟨*code*⟩ is exported with `display:none`.

Can be used to declare formal content and preserve its semantics in HTML without generating output.

# Chapter 6

# Document Features

## 6.1 Document Fragments

sfragment (*env.*) To make reusability of document fragments more feasible, sTeX provides the **sfragment** environment. \begin{sfragment}[id=⟨*id*⟩,short=⟨*short title*⟩]{section title} calls \part, \chapter, \section, \subsection, \subsubsection, \paragraph or \subparagraph with argument {section title} depending on the current *section level* and availability, and increases the level accordingly.

The ⟨*id*⟩ can be used for cross-document references (see Section 0.3 (Cross-Document References) in the sTeX Manual).

blindfragment (*env.*) In the case where we want to increase the section level *without* producing a corresponding section header, the **blindfragment** environment can be used. This allows e.g. typesetting \sections before the first \chapter.

\skipfragment The \skipfragment macro "skips an sfragment", i.e. it just steps the respective sectioning counter. This macro is useful, when we want to keep two documents in sync structurally, so that section numbers match up: Any section that is left out in one becomes a \skipfragment.

\setsectionlevel The \setsectionlevel macro sets the current section level to that provided as argument. This is particularly useful in the preamble of a document, as to be ignored in e.g. \inputref and make sure that sectioning proceeds as desired; e.g. \setsectionlevel{section} make sure that the first sfragment will be typeset as a \section (rather than e.g. a \part).

\currentsectionlevel
\Currentsectionlevel The \currentsectionlevel macro produces the literal string corresponding to the current section level – e.g. within a chapter (but outside of a section), \currentsectionlevel produces "chapter".

The \Currentsectionlevel macro does the same, but capitalizes the first letter; e.g. in the above situation, \Currentsectionlevel produces "Chapter".

## 6.2 Using and Referencing Document Fragments

`\inputref` `\inputref [⟨archive⟩]{⟨file⟩}`

Inputs the file ⟨*file*⟩ in ⟨*archive*⟩'s `source` directory. If `[⟨archive⟩]` is empty, the current archive's `source` directory is used. If there is no current archive, ⟨*file*⟩ is resolved relative to the current file.

The file's content is processed within a TEX group when using `pdflatex`. When converting to HTML however, the file is not processed *at all*, and instead, a reference to the file is inserted, that can be replaced by the HTML generated by the referenced file by e.g. the MMT system.

This is the recommended method to assemble documents from individual `.tex` files.

`\mhinput` Like `\inputref`, but actually calls `\input` in both PDF and HTML mode. Useful for small fragments or those without modules, but generally `\inputref` should be preferred.

`\ifinputref` `\IfInputref` `\ifinputref` is a TEX conditional for whether the current file is currently processed via `\inputref`.

`\IfInputref {⟨true code⟩}{⟨false code⟩}` behaves like `\ifinputref⟨true code⟩\else⟨false code⟩\fi` when using `pdflatex`; in HTML mode however, *both* arguments are processed and marked-up accordingly, so a hosting server (like MMT) can dynamically decide which parts to show or omit.

`\mhgraphics` `\cmhgraphics` *If* the graphicx package is loaded, `\mhgraphics` takes the same arguments as `\includegraphics`, with the additional optional key `archive`. It then resolves the file path in `\mhgraphics[archive=some/archive]{some/image}` relative to the `source`-folder of the `some/archive` archive. If no `archive` is provided, the file path `some/image` is resolved relative to the current archive (if existent).

`\cmhgraphics` additional wraps the image in a `center`-environment.

`\lstinputmhlisting` `\clstinputmhlisting` Like `\mhgraphics`, but for `\lstinputlisting` instead of `\includegraphics`. Only defined if the listings package is loaded.

## 6.3 Cross-Document References

If we take features like `\inputref` and `\mhinput` (and the `sfragment` environment) seriously and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputref`s a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also `\inputref`ed in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex`

it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or "*Definition 1 in the section on Foo*" respectively.

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary if the reference target occurs in the *same* document, but if not, we need to know where to find the label,
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

Additionally, the document in which we want to reference a label needs a title for external references.

---

`\sref` `\sref [archive=⟨archive1⟩,file=⟨file1⟩] {⟨label⟩}[archive=⟨archive2⟩,file=⟨file2⟩,title=⟨title⟩]`

---

This macro references ⟨*label*⟩ (declared in ⟨*file1*⟩ in math archive ⟨*archive1*⟩). If the object (section, figure, etc.) with that label occurs (eventually) in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the hyperref package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object's name as it occurs in the file ⟨*file2*⟩ in archive ⟨*archive2*⟩, followed by the title.

In HTML mode, the reference additionally links to the HTML of the `file1`.[10]

This works by storing labels during compilation in a file ⟨*jobname*⟩`.sref`, analogous to e.g. the `.toc`. Note that this consequently requires both `file1.tex` and `file2.tex` to have been compiled previously, to generate the `.sref` file.

For example, doing
`\sref[file=tutorial/full.en]{sec:basics}[file=tutorial.en,title=the \stex Tutorial]`
in this very document fragment (`[sTeX/Documentation]macros/sref.en.tex`) will yield Part I (The Basics) in the sTeX Tutorial if compiled itself, or if compiled as part of the sTeX manual, and will yield the `\autoref` link chapter 2 in the documentation (which includes the tutorial).

---

`\srefsetin` `\srefsetin [⟨archive2⟩]{⟨file2⟩}{⟨title⟩}`

---

Sets a default value for the optional arguments ⟨*archive2*⟩, ⟨*file2*⟩ and ⟨*title*⟩ of `\sref`. If the second set of optional arguments in `\sref` are omitted, these default values are used. Particularly useful to set in a preamble.

---

`\sreflabel` `\sreflabel {⟨label⟩}` sets a label analogous to `\label{⟨label⟩}`, but for use in `\sref`.

Note that for every sTeX macro or environment that takes an optional `id=⟨id⟩` argument, the ⟨*id*⟩ (if non-empty) generates an `\sreflabel` automatically.

For example, `\begin{sfragment}[id=foo]{Foo}` is equivalent to `\begin{sfragment}{Foo}\sreflabel{foo}`.

\extref `\extref [archive=⟨archive1⟩,file=⟨file1⟩] {⟨label⟩}{archive=⟨archive2⟩,file=⟨file2⟩,title=⟨title⟩}`

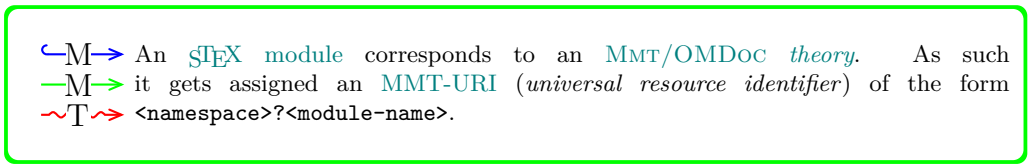
Like \sref, but with the third argument mandatory, \extref will *always* produce the output as if ⟨label⟩ would *not* occur in the current document.

# Chapter 7

# Modules and Symbols

## 7.1 Modules

A module is required to declare any new formal content such as symbols or notations (but not variables, which may be introduced anywhere).




`smodule` (*env.*) A new module is declared using the basic syntax

`\begin{smodule}[options]{ModuleName}...\end{smodule}`.

A module is required to declare any new formal content such as symbols or notations (but not variables, which may be introduced anywhere).

The `smodule`-environment takes several keyword arguments, all of which are optional:

`title` (⟨*token list*⟩) to display in customizations.

`style` (⟨*string*⟩∗) for use in customizations, see Part 1 (Customizing Typesetting) in the sTeX Manual

`id` (⟨*string*⟩) for cross-referencing, see `\sreflabel`.

`ns` (⟨*URI*⟩) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed from the containing file and archive's namespace.

`lang` (⟨*language*⟩) if not set, computed from the current file name (e.g. `foo.en.tex`).

`sig` (⟨*language*⟩) see below.

`\STEXexport` `\STEXexport{`⟨*code*⟩`}` executes ⟨*code*⟩ immediately and every time the current module is being used.

For technical reasons, \STEXexport processes its content in the expl3 category code scheme – what this means is that all spaces are ignored entirely, and the characters _ and : are valid characters in macro names.
In practice, this means you will have to use the ~ character for spaces, and if you want to use a subscript _, you should use the macro \c_math_subscript_token instead.
Also, note that no *global* macro definitions should happen in \STEXexport; this can lead to unexpected behaviour if the containing module has been used previously in the current document.

### 7.1.1 Signature Modules, Languages, and Multilinguality

if the current file is a translation of a file with the same base name but a different language suffix, setting sig=<lang> will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.

For example, we can have a file Foo.en.tex, that declares and documents a module Foo (using \begin{smodule}{Foo}). If we put a file Foo.de.tex next to it, we can do \begin{smodule}[sig=en]{Foo} to have all the content in the module Foo (as declared in Foo.en.tex) available and translate its document content to german.

The MMT back-end, when serving STEX content as HTML, will always attempt to find documentation in the language corresponding to the context; e.g. a user's preference.

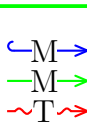
## 7.2 Symbol Declarations

\symdecl

\symdecl {⟨*mname*⟩}[⟨*options*⟩]

The \symdecl macro is the simplest way to introduce a new symbol. If ⟨*options*⟩ contains name=⟨*name*⟩, then ⟨*name*⟩ is the *name* of the symbol; otherwise, ⟨*mname*⟩ is used for the *name*. Additionally, a semantic macro \mname is generated.

The starred variant \symdecl* does not generate a semantic macro, in which case the name-option is superfluous.



\symdecl introduces a new MMT/OMDoc constant in the current module (i.e. MMT/OMDoc theory). Correspondingly, they get assigned the MMT-URI <module-URI>?<constant-name>.

\symdecl takes the following optional arguments:

name see above,

args the arity of the symbol and its semantic macro; may be a number 0...9 or a string consisting of the characters i, a, b and B of length ≤ 9,

type the symbol's type,

def the symbol's definiens,

`return` the symbol's *return code* (see below), most commonly the semantic macro of a mathematical structure,

`assoc` how to resolve arguments of mode `a` or `B`; may be `pre`, `bin`, `binl`, `binr` or `conj`,

`reorder` how to reorder the arguments in OMDoc (*advanced*),

`role` symbols with certain roles are treated in particular ways in Mmt/OMDoc (*advanced*),

`argtypes` TODO[11].

`\textsymdecl` `\textsymdecl{⟨mname⟩}[⟨options⟩]{⟨code⟩}`

Like `\symdecl`, but requires that the symbol has arity 0 (hence `\textsymdecl` does not take the `args`-option), and generates a semantic macro that takes no arguments in either text or math mode, and produces marked-up ⟨*code*⟩ as output.

Additionally, a macro `\⟨mname⟩name` is generated that produces ⟨*code*⟩ without any semantic markup.

`\symdef` `\symdef{⟨mname⟩}[⟨options⟩]{⟨notation⟩}`

Combines the functionalities and optional arguments of `\symdecl` and `\notation` in one.

### 7.2.1 Returns

Assume we have a symbol `foo` with semantic macro `\foo`, (examplary) taking two arguments, and `return=`⟨*code*⟩. If we do `\foo{a}{b}!`, the return code is simply ignored. If we do `\foo{a}{b}` *without* the `!`, here is what happens:

1. SmTeX will replace `#1` and `#2` in ⟨*code*⟩ by `a` and `b`, yielding ⟨*retcode*⟩.
2. SmTeX will insert ⟨*retcode*⟩`{\foo{a}{b}!}` in the input token stream.

This means that ⟨*code*⟩ should contain at most ⟨*arity of foo*⟩ argument markers, and eat precisely one argument appended to ⟨*code*⟩.

When in doubt, we recommend only using semantic macros for mathematical structures (with only optional arguments) and `\apply` (with only optional arguments) in `return`.

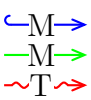
## 7.3 Referencing Symbols

`\symref` `\symref{⟨symbol⟩}{⟨text⟩}`
`\sr`

The `\symref` macro (and its short version `\sr`) is the most general variant to mark-up arbitrary LaTeX code ⟨*text*⟩ with the symbol.

[11] TODO: experimental

This is as good a place as any other to explain how sTeX resolves a string `symbol` to an actual symbol.
If `\symbol` is a semantic macro, then sTeX has no trouble resolving `symbolname` to the full MMT-URI of the symbol that is being invoked.
However, especially in `\symname` (or if a symbol was introduced using `\symdecl*` without generating a semantic macro), we might prefer to use the *name* of a symbol directly for readability – e.g. we would want to write `A \symname{natural number} is...` rather than `A \symname{Nat} is....` sTeX attempts to handle this case thusly:
If `symbol` does *not* correspond to a semantic macro `\symbol` and does *not* contain a `?`, then sTeX checks all symbols currently in scope until it finds one, whose name is `symbol`. If `symbol` is of the form `pre?name`, sTeX first looks through all modules currently in scope, whose full MMT-URI ends with `pre`, and then looks for a symbol with name `name` in those. This allows for disambiguating more precisely, e.g. by saying `\symname{Integers?addition}` or `\symname{RealNumbers?addition}` in the case where several `addition`s are in scope.



`\symref{`⟨*symbol*⟩`}{`⟨*text*⟩`}` in MMT/OMDoc generates the term `<OMS name="{`⟨*symbol URI*⟩`}"/>`.

`\symname` `\symname[pre=`⟨*pre*⟩`,post=`⟨*post*⟩`]{`⟨*symbol*⟩`}`
`\sn`
`\Symname`
`\Sn`
`\sns`
`\Sns`

If the symbol referenced by ⟨*symbol*⟩ has name `name`, this is a shortcut for `\symref{`⟨*symbol*⟩`}{`⟨*pre*⟩`name`⟨*post*⟩`}`.

For example, given a symbol `agroup` with name `abelian group`, we can do `\symname[pre=Non-,post=s]{agroup}` to produce Non-abelian groups.

`\sn` is a shorter variant for `\symname`; `\Symname` and `\Sn` additionally capitalize the first letter. `\sns` and `\Sns` are short for `\sn [post=s]` and `\Sn [post=s]`, respectively.

`\srefsym` `\srefsym{`⟨*symbol*⟩`}{`⟨*text*⟩`}`
`\srefsymuri`

turns ⟨*text*⟩ into a link to

- The documentation of ⟨*symbol*⟩, if it occurs in the same document, or
- the symbol's documentation online, based on the containing math archive's `url-base`.

`\srefsymuri` does the same, but expects a symbol's full MMT-URI as first argument. This is particularly useful for e.g. customizing highlighting (see chapter 9).

`\symuse` `\symuse{`⟨*symbol*⟩`}` behaves exactly like a semantic macro for ⟨*symbol*⟩.

## 7.4 Notations and Semantic Macros

\notation `\notation{⟨symbol⟩}[⟨options⟩]{⟨code⟩}`

introduces a new notation for the referenced symbol.

The starred variant `\notation*` sets this notation as the (new) default notation.

The optional arguments are:

- `prec=⟨opprec⟩;⟨argprec 1⟩x...x⟨argprec n⟩`: An operator precedence and one argument precedence for each argument of the semantic macro. If no argument precedences are given, all argument precedences are equal to the operator precedence. By default, all precedences are 0, unless the symbol takes no argument, in which case the operator precedence is `\neginfprec` (negative infinity).

  `prec=nobrackets` is an abbreviation for `prec=\neginfprec;\infprec x...x\infprec`.

- `op=⟨code⟩`: An operator notation. If none is given, the notation component marked with `\maincomp` is used. If no `\maincomp` occurs in the notation, the default operator notation is `\symname{⟨symbol⟩}`.

- `variant=⟨id⟩`: An id for this notation. The key `variant=` can be omitted; i.e. `\notation[foo]` is equivalent to `\notation[variant=foo]`.

\comp
\maincomp

`\comp` is used to mark notation components in a `\notation` to be highlighted. Additionally, each notation can use `\maincomp` at most once to mark the *primary* notation component.

Ideally, `\comp` would not be necessary: Everything in a notation that is *not* an argument should be a notation component. Unfortunately, it is computationally expensive to determine where an argument begins and ends, and the argument markers `#n` may themselves be nested in other macro applications or TEX groups, making it ultimately almost impossible to determine them automatically while also remaining compatible with arbitrary highlighting customizations (such as tooltips, hyperlinks, colors) that users might employ, and that are ultimately invoked by `\comp`.

Note that it is required that

1. the argument markers `#n` never occur inside a `\comp`, and
2. no semantic macros may ever occur inside a notation.

Both criteria are not just required for technical reasons, but conceptionally meaningful:
The underlying principle is that the arguments to a semantic macro represent *arguments to the mathematical operation* represented by a symbol. For example, a semantic macro application `\plus{a}{b}` would represent *the actual addition of (mathematical objects) a and b*. It should therefore be impossible for *a* or *b* to be part of a notation component of `\plus`.
Similarly, a semantic macro can not conceptually be part of the notation of `\plus`,

since a symbol represents a *distinct (mathematical) concept* with *its own semantics*, and notations are syntactic representations of the very symbol to which the notation belongs.
If you want an argument to a semantic macro to be a purely syntactic parameter, then you are likely somewhat confused with respect to the distinction between the precise *syntax* and *semantics* of the symbol you are trying to declare (which happens quite often even to experienced STEX users, like us), and might want to give those another thought - quite likely, the concept you aim to implement does not actually represent a semantically meaningful (mathematical) concept, and you will want to use `\def` and similar native LATEX macro definitions rather than semantic macros.

`\setnotation`

The first notation provided will stay the default notation unless explicitly changed:
`\setnotation{`⟨*symbol*⟩`}{`⟨*id*⟩`}` sets the default notation of ⟨`symbol`⟩ to that with id ⟨*id*⟩.

### 7.4.1 Precedences and Bracketing

`\infprec`
`\neginfprec`

`\infprec` and `\neginfprec` represent *infinitely large* and *infinitely small* precedences, respectively.

STEX decides whether to insert parentheses by comparing operator precedences to a *downward precedence* $p_d$ with initial value `\infprec`. When encountering a semantic macro, STEX takes the operator precedence $p_{op}$ of the notation used and checks whether $p_{op} > p_d$. If so, STEX inserts parentheses.
When STEX steps into an argument of a semantic macro, it sets $p_d$ to the respective argument precedence of the notation used.

**Example 25**
Consider semantic macros `\plus` and `\mult` taking two arguments, with notations $a+b$ and $a \cdot b$ respectively, and precedences 100 for `\plus` and 50 for `\mult`.
Consider `$\plus{a,\mult{b,\plus{c,d}}}$` (i.e. $a + b \cdot (c + d)$). Then:

1. STEX starts out with $p_d =$`\infprec`.
2. STEX encounters `\plus` with $p_{op} = 100$. Since $100 \not> $`\infprec`, it inserts no parentheses.
3. Next, STEX encounters the two arguments for `\plus`. Both have no specifically provided argument precedence, so STEX uses $p_d = p_{op} = 100$ for both and recurses.
4. Next, STEX encounters `\mult{b,...}`, whose notation has $p_{op} = 50$.
5. We compare to the current downward precedence $p_d$ set by `\plus`, arriving at $p_{op} = 50 \not> 100 = p_d$, so STEX again inserts no parentheses.

6. Since the notation of \mult has no explicitly set argument precedences, sTeX again uses the operator precedence for the arguments of \mult, hence sets $p_d = p_{op} = 50$ and recurses.

7. Next, sTeX encounters the inner \plus{c,...} whose notation has $p_{op} = 100$. We compare to the current downward precedence $p_d$ set by \mult, arriving at $p_{op} = 100 > 50 = p_d$ – which finally prompts sTeX to insert parentheses, and we proceed as before.

\dobracket `\dobracket{`⟨*code*⟩`}` wraps parentheses around `{`⟨*code*⟩`}`.

\withbrackets `\withbrackets{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*code*⟩`}` uses the opening and closing parentheses ⟨***left***⟩ and ⟨***right***⟩ for the next pair of parentheses automatically inserted in `{`⟨*code*⟩`}`.

### 7.4.2 Notations for Argument Sequences

The following macros can be used in notations that take mode `a` or `B` arguments:

\argsep `\argsep{`⟨*parameter token*⟩`}{`⟨*separator*⟩`}`

takes the elements of the argument sequence in position ⟨***parameter token***⟩ and separates them by ⟨***separator***⟩.

Note that the first argument *must* be a parameter token of the form `#k`, and the argument at position `k` of the notation has to have argument mode `a` or `B`.

\argmap `\argmap{`⟨*parameter token*⟩`}{`⟨*code*⟩`}{`⟨*separator*⟩`}`

takes the elements of the argument sequence in position ⟨***parameter token***⟩, applies the code `{`⟨*code*⟩`}` to each of them (which therefore should use `##1`) and separates them by ⟨***separator***⟩.

For example, the notation `{\argmap{#1}{X^{##1}}{++}}` applied to the argument `{a,b,c}` produces $X^a + +X^b + +X^c$.

\argarraymap TODO[12]

### 7.4.3 Semantic Macros

Assume we have a semantic macro \smacro taking (exemplary) two arguments. The precise behaviour of \smacro depends on whether we are in text or math mode.

**Math Mode** \smacro`!` produces the default operator notation of its symbol. Without `!`, \smacro expects at least two arguments, and \smacro`{a}{b}!` produces the default notation of its symbol.

If the symbol has a `return` code, then \smacro`{a}{b}` continues with executing the `return` code. Otherwise, \smacro`{a}{b}` also simply produces the default operator notation.

The starred variants \smacro`*` and \smacro`!*` behave as in *text mode*.

**Text Mode** \smacro!{⟨*arg*⟩} marks up ⟨*arg*⟩ similarly to how \symref{smacro}{⟨*arg*⟩} would.

Without the !, \smacro still only takes a single argument, but it is expected, that within ⟨*arg*⟩, the arguments for the symbol are explicitly marked up. The \comp macro is allowed in ⟨*arg*⟩ to determine the components of ⟨*arg*⟩ to be highlighted.

\arg The \arg macro can be used to explicitly mark the arguments of a semantic macro in text mode.

By default, they are numbered consecutively; e.g. \smacro{...\arg{a}...\arg{b}} determines a and b to be the (first and second) arguments.

The starred variant \arg* allows for marking up the arguments, but does not produce any output. This can be used to provide arguments that are not mentioned in the text we want to mark up, because they are implicitly obvious or mentioned elsewhere.

If we want to change the order of the arguments, we can provide the precise argument number as an optional argument; e.g. \smacro{...\arg[2]{a}...\arg[1]{b}} determines b to be the first and a to be the second argument.

An argument number may be used repeatedly, if the corresponding argument mode is a or B.

Applications of semantic macros with arguments are translated to MMT/OMDoc as OMA-terms with head <OMS name="⟨*symbol*⟩"/>, or <OMBIND name="⟨*symbol*⟩"/>, depending on the absence or presence of argument mode b or B arguments.
Semantic macros with no arguments or invoked with ! correspond to OMS directly.

## 7.5 Simple Inheritance

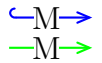
There are three macros that allow for opening a module, making its contents available for use:

\usemodule \usemodule{⟨*module*⟩} is allowed anywhere and makes the module's contents available up to the current TEX group. This is the right macro to use outside of modules, or when none of its contents use any of the used module's symbols directly (e.g. in types or definientia).

\requiremodule \requiremodule{⟨*module*⟩} is only allowed in modules and makes the required module's contents available within the current module. The imported symbols can be safely used in types and definientia, but not in the return code of symbols, and the imported content is not exported further – i.e. using the current module does not also open the required module.

\importmodule

\importmodule{⟨*module*⟩} is only allowed in modules and makes the required module's contents available within the current module. The imported symbols can be safely used anywhere, and the imported content exported to any modules subsequently importing the current one.




In MMT, every *document* and every module induces an MMT theory. \usemodule induces and MMT include in the document theory, \importmodule and \requiremodule both induce an include in the module's theory.

It is worth going into some detail how exactly \usemodule, \importmodule and \requiremodule resolve their arguments to find the desired module – which is closely related to the *namespace* generated for a module, that is used to generate its MMT-URI.

Ideally, sTEX would allow for arbitrary MMT-URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring it – like MMT in fact allows for.
Unfortunately, TEX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that sTEX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses sake, we describe how they are constructed:

- If `\begin{smodule}{Foo}` occurs in a file `/path/to/file/Foo[.⟨lang⟩].tex` which does not belong to an math archive, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.⟨lang⟩].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of math archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.
If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source` directory is replaced by the archive's namespace URI.

Conversely, here is how namespaces/URIs and file paths are computed in import statements, examplary \importmodule:

- \importmodule`{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same file or, if not, in a file `Foo[.⟨lang⟩].tex` in the same directory.
- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same file, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, is has to be declared in a file `Foo[.⟨lang⟩].tex` directly in the archive's `source` directory.
- Similarly, in \importmodule`{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory

and namespace outside of an archive, or relative to the current archive's top-level namespace and `source` directory, respectively.

The module `Foo` must either be declared in the file ⟨*top-directory*⟩`/some/path/Foo[.`⟨*lang*⟩`].tex`, or in ⟨*top-directory*⟩`/some/path[.`⟨*lang*⟩`].tex` (which are checked in that order).

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the `MathHub` directory.

## 7.6 Variables and Sequences

`\vardef` `\vardef{`⟨*mname*⟩`}[`⟨*options*⟩`]{`⟨*notation*⟩`}`

Takes the same arguments as `\symdef`, but produces a variable rather than a symbol. Variables definitions are always local to the current TEX group and are allowed anywhere (i.e. outside of modules).

⟨*options*⟩ may include the additional keyword `bind`, in which case the variable will be appropriately abstracted away in statements (see also `\varbind`).

Unlike `\symdef`, there is no starred variant `\vardef*` – variables always generate a semantic macro.

The semantic macro for a variable behaves analogously to that of a symbol.

Variables induces the same MMT/OMDoc terms as symbols do, except for the head of the term being `<OMV name="..."/>` instead of `<OMS/>`.

`\varnotation` `\varnotation{`⟨*variable*⟩`}[`⟨*options*⟩`]{`⟨*notation*⟩`}`

Takes the exact same argumens as `\notation`, but attaches an additional notation to the *variable* ⟨`variable`⟩ rather than a symbol.

`\svar` `\svar[`⟨*name*⟩`]{`⟨*text*⟩`}`

Semantically marks up ⟨*text*⟩ as representing a variable ⟨*name*⟩. The variable does not need to have been defined prior. If no ⟨*name*⟩ is givenm ⟨*text*⟩ will be used as the name.

This is useful in situations like "throwaway expressions" or remarks; e.g. `$\plus{\svar{n},\svar{m}}$ means...`

\varseq `\varseq{⟨mname⟩}[⟨options⟩]{⟨range⟩}{⟨notation⟩}`

Declares a new variable sequence. The ⟨*options*⟩ are the same as for `\vardef`. If not provided, `args=1` by default (a 0-ary sequence would just be a normal variable).

A type (given as `type=`) is interpreted to be the type of every element $a_i$ of the sequence $a_1, ..., a_n$ (not of the sequence itself). If the type is itself a sequence $A_1, ..., A_n$, the assumption is that its range is the same as the one of the new sequence, and the type of every $a_i$ in the sequence is $A_i$.

⟨*range*⟩ needs to be a comma-separated sequence of either `args` many arguments, or `\ellipses`.

The resulting semantic macro is allowed anywhere SᴛEX expects an argument mode `a` or `B` argument.

\ellipses Represents ellipses in a range; produces `\ellipses` in math mode.

\seqmap `\seqmap{⟨code⟩}{⟨sequence⟩}`

Maps the function ⟨*code*⟩ (containing `#1`) over every element of the ⟨*sequence*⟩.

Is allowed anywhere SᴛEX expects an argument mode `a` or `B` argument.

## 7.7 Structures

Mathematical structure bundle interdependent symbols together.

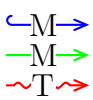
`mathstructure` (*env.*) `\begin{mathstructure}{⟨mname⟩}[⟨name⟩,this=⟨code⟩]` opens a new mathematical structure with name ⟨*mname*⟩ (if provided) or ⟨*mname*⟩ (otherwise), and semantic macro `\mname`. It subsequently behaves like the `smodule` environment.

\this The optional `this=⟨code⟩` option allows for setting the typesetting of the `\this` macro within a `mathstructure`. In particular, `\this` can be used in notations for symbols declared in the structure. `\this` can be though of as representing "*the*" (current) instance of this structure.

`extstructure` (*env.*) `\begin{extstructure}{⟨mname⟩}[⟨name⟩,this=⟨code⟩]{⟨structs⟩}` opens a new mathematical structure extending the structures given in ⟨*structs*⟩ (a comma-separated list of names).

`extstructure*` (*env.*) `\begin{extstructure*}{⟨struct⟩}` opens a new mathematical structure *conservatively* extending the (single) structure ⟨*struct*⟩. *Conservative* meaning: Every symbol newly introduced in this structure needs to have a *definiens*. The new symbols are attached as fields directly to ⟨*struct*⟩.

\usestructure The `\usestructure` macro behaves like `\usemodule` for mathematical structures, making the symbols available to use directly.



`mathstructure` make use of the *Theories-as-Types* paradigm (see [**MueRabKoh:tat18**]):

↪M↠ `\begin{mathstructure}{⟨name⟩}` creates a nested theory with name ⟨name⟩`-module`. The constant ⟨name⟩ is defined as a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in ⟨name⟩`-module`.
—M→
⇝T↝

### 7.7.1 Semantic Macros for Structures

Assume we have a mathematical structure with semantic macro `\struct`:

**Example 26**

```
\begin{mathstructure}{struct}
  \symdef{fielda}{a}
  \symdef{fieldb}[args=2]{#1 \maincomp{b} #2}
  \symdef{fieldc}[args=2,def=\sn{fieldb}]{#1 \maincomp{c} #2}
  \inlineass[name=axiom1]{\conclusion{some axiom}}
\end{mathstructure}
\notation{struct}{StRuCt}
```

- If `\struct` has no notations, then `$\struct!$` produces $\langle a,b,c\rangle$. Otherwise, it produces the notation, i.e. $StRuCt$. In both cases, `$\struct{}{}$` produces $\langle a,b,c\rangle$ (for reasons that will become clearer in a moment).
- `$\struct{}{}$` (or `$\struct!$` in the case where no notations are around) can be modified in the following ways:
  - `$\struct{}{}[`⟨fieldname⟩`,...]$` lets you pick, which precise fields to show, so e.g. `$\struct{}{}[fielda,fieldb]$` produces $\langle a,b\rangle$. By default, `$\struct{}{}$` shows exactly the fields that have semantic macros (which are also used to access the fields in `$\struct{...}{fieldname}`).
  - `$\struct{}{}[`⟨id⟩`]$` lets you pick the notation of the "mathematical structure" symbol to use to typeset the structure; e.g. `$\struct{}{}[parens]` yields $(a,b,c)$, and (combining both) `$\struct{}{}[fielda,fieldb][angle]` yields $\langle a,b\rangle$.
- The two arguments in `$\struct{first}{second}$` represent 1. the term that is to be treated as an instance of `\struct`, and 2. the precise field to invoke. If the first is empty, then there is no instance. If the second is empty, sTeX will present all of them (that are not assertions). Hence, `$\struct{S}{}$` yields $\langle a_S,b_S,c_S\rangle$, `$\struct{S}{fielda}$` produces $a_S$, `$\struct{}{fielda}$` produces $a$, and `$\struct{S}{fieldb}{x}{y}$` produces $xb_Sy$.
- For the sake of completion, `$\struct{first}!$` simply produces the given argument; e.g. `$\struct{S}!$` simply produces $S$.

More precisely: `$\struct{`⟨code⟩`}$` acts like a "type coercion" of ⟨code⟩ to be an instance of `\struct`.

Of course, it is (occasionally, but) rarely useful to use the semantic macro `\struct` by giving it two arguments *manually*; but this is what sTeX does when using `\struct` in the `return` of a symbol (or variable).

Continuing:

- `$\struct[comp=⟨code⟩]{...}{...}$` applies ⟨*code*⟩ (using `#1`) to every occurence of `\maincomp` in the notations of the fields, as a replacement for the default modification `{#1}_{\this}`. For example, `$\struct[comp={#1}^{Foo}]{S}{}$` produces $\langle a^{Foo}, b^{Foo}, c^{Foo}\rangle$, and `$\struct[comp={#1}^{\this}]{S}{fieldb}{x}{y}$` yields $xb^S y$.

- `$\struct[this=⟨code⟩]{...}{...}$` modifies the way `\this` is being typeset; i.e. the presentation of the first `{...}` argument. For example `$\struct[this=T]{S}{}$` produces $\langle a, b, c\rangle$ – since `\this` is not being used in the notations of the fields. Note that the `this=` and `comp=` variants are (as of yet) mutually incompatible.

- Finally, `$\struct[⟨fieldname⟩=⟨code⟩]{...}{...}$` *assigns* the field ⟨*fieldname*⟩ to the term ⟨*code*⟩. ⟨*code*⟩ is subsequently used when using `{...}{}`, but not in fields directly. For example, `$\struct[fielda=A]{S}{}$` produces $\langle A!, b_S, c_S\rangle$, but `$\struct[fielda=A]{S}{fielda}$` produces $a_S$.

  Note the insertion of `!` behind the `A` – this is to make sure that assignments to semantic macros that takes arguments don't accidentally eat more than they should.

  Also note that multiple assignments can be done in the same pair of `[]`, or chained – i.e. both `$\struct[fielda=A,fieldb=B]...$` and `$\struct[fielda=A][fieldb=B]...$` are valid and equivalent. `$\struct[fielda=A,fielda=B]...$` however is not – every field may be assigned at most once.

# Chapter 8

# Statements

sTeX provides four environments to semantically annotate various kinds of statements:

`sdefinition` (*env.*) The `sdefinition` environment represents (primarly mathematical) *definitions*; in particular for symbols. The contents of the environment will be used as *documentation* for any symbol that either orrurs as a `\definiendum` (or `\definame`) within the `sdefinition`, or that is listed in the optional `for=` argument of the environment.

If a `\definiens` occurs, this will be used by MMT as the formal definiens for the respective symbol.

`sassertion` (*env.*) The `sassertion` environment represents *assertions*, i.e. propositions such as *theorems*, *lemmata*, *axioms*, etc. If a `\conclusion` occurs within the `sassertion`, its argument will be used as the formal *statement* of the assertion.

`sexample` (*env.*) The `sexample` environment represents examples, the `for=` attribute specifies the concept this is an example for. For counterexamples use `style=counterexample`

`sparagraph` (*env.*) The `sparagraph` environment represents all other kinds of (logical) paragraphs, such as remarks, comments, transitions bwetween topics, recaps, reminders, etc.

All of these take the same arguments:

- `for=`⟨*csl*⟩: a comma-separated list of symbols.
- The same optional arguments as `\symdecl`, with `macro=` replacing the name of the semantic macro. All of them are only relevant, if either `name=` or `macro=` are provided.

  As with `\symdecl`, if no `name` is given, but `macro` is, then the same name is used for both the semantic macro and the symbol itself.

  If `name` is given but `macro` isn't, no semantic macro is generated. Subsequently, the newly generated symbol is added the `for`-list.
- `style`: see chapter 9.
- `title`: a title to use in various styles (see chapter 9).
- `id`: a label to use for `\sref`.

\inlineass
\inlinedef
\inlineex

The macros \inlineass, \inlinedef and \inlineex behave like the sassertion, sdefinition and sexample environments respectively, but take the text to annotate as an argument, rather than as the body of an environment, and do not break paragraphs.

The same macros available in the environments are also available in the argument of the \inline* macros.

\varbind

\varbind{⟨cls⟩}

retroactively attaches the bind option to every variable provided (as a comma-separated list).

## 8.1 More on Definitions

In sdefinition (and sparagraph with style=symdoc), the following additional macros are available:

\definiendum
\definame
\Definame
\defnotation

The \definiendum macro behaves largely like \symref, but it uses a dedicated highlighting for *definienda* and adds the referenced symbol to the for= list of the environment.

\definame is to \definiendum as \symname is to \symref. Analogously, \Definame behaves like \Symname.

\defnotation can be used in math mode to apply the \definiendum highlighting to notations.

\definiens

The \definiens macro can be used to semantically annotate the *definiens* in a sdefinition.

If the sdefinition environment has several elements in its for list, an optional argument \definiens[⟨symbol⟩]{...} can be used to tell STEX which symbol's definiens this is. By default, the *first* symbol in the for list is used.


Here is how MMT will treat the fragment marked up with \definiens:
Firstly, it will attempt to translate its contents into an MMT/OMDoc term. This succeeds easily if \definiens is some semantic macro (applied to arguments).
Secondly, it will check for all variables currently in scope, that were defined with the optional argument bind. It then will check, whether a symbol is in scope, that has role=lambda. If so, it will use that lambda symbol to bind all these variables (in the order in which they were defined) in the term. If no lambda symbol is found, it will use the bind symbol that ships with STEX.
The final term will be attached as definiens to the corresponding MMT constant, if it was declared in the same module as the \definiens occurrence.

## 8.2 More on Assertions

`\premise`
`\conclusion`

The `\conclusion` macro can be used to mark up the actual statement within an `sassertion`. The `\premise` macro can be used to additionally mark up *premises*.

If the `sassertion` environment has several elements in its `for` list, an optional argument `\conclusion[`⟨*symbol*⟩`]{...}` can be used to tell SᴛEX which symbol's statement this is. By default, the *first* symbol in the `for` list is used.

Here is how MMT will treat the fragments marked up with `\conclusion` and `\premise`:
Firstly, it will attempt to translate the contents of `\conclusion` into an MMT/OMDoc term $c$. This succeeds easily if the `\conclusion` is some semantic macro (applied to arguments).
Secondly, it will collect all fragments marked up with `\premise` and do the same to them $(p_1, ..., p_n)$.
It will then check, whether a symbol is in scope, that has `role=implication`. If so, it will use that `implication` symbol to attach all the premises to the conclusion, resulting in $t := \texttt{imply}(p_1, ..., p_n, c)$.
Next, it will check for all variables currently in scope, that were defined with the optional argument `bind`. It then will check, whether a symbol is in scope, that has `role=forall`. If so, it will use that `forall` symbol to bind all these variables (in the order in which they were defined) in the term $t$.
Finally, it will check, whether a symbol is in scope, that has `role=judgment`. If so, it will set $t := \texttt{judgment}(t)$.
If no `forall` symbol is found, it will first apply the `judgment` symbol (if existent) and then use the bind symbol that ships with SᴛEX to bind the variables.
The final term will be attached as type to the corresponding MMT constant, if it was declared in the same module as the `\definiens` occurrence.

`sproof` (*env.*) TODO[13]

[13] TODO: proofs

# Chapter 9

# Customizing Typesetting

There are two kinds of typesetting that can be customized in STEX: symbol references (notation components, \symref, variables, etc.) are highlighted using a small set of macros that can be simply redefined by authors.

Other macros and environments usually have more complicated "typesetting rules" associated with them – often in the form of other already existing environments that should be used.

Lastly, in HTML we can provide custom CSS rules in math archives that determine the styling of certain environments, so that the actual presentation depends on the document in which the fragments are included (e.g. via \inputref), rather than the file the fragment is implemented in.

It is generally recommended to implement these customizations in a preamble in the lib directory (see Section 0.3 (The lib-Directory) in the STEX Manual)

## 9.1 Highlighting Symbol References

\symrefemph
\symrefemph@uri

\symrefemph governs how references via \symref (or \symname, or theirt short variants) are highlighted;

Doing \symref{⟨*symbol*⟩}{⟨*text*⟩} ultimately expands to \symrefemph@uri{⟨*text*⟩}{⟨*symbol URI*⟩}, the default implementation of which is just \symrefemph{⟨*text*⟩}. The default implementation of \symrefemph, in turn, is just \emph{⟨*text*⟩}.

If you only want to change e.g. the color of \symrefs, you only need to redefine \symrefemph, e.g. using

```
\renewcommand\symrefemph[1]{\textcolor{red}{#1}}
```

the @uri variant is useful if you want to link somewhere, or show the URI in a tooltip. The stex-highlighting package does both, using:

```
\usepackage{pdfcomment}
\protected\def\symrefemph@uri#1#2{%
  \pdftooltip{%
    \srefsymuri{#2}{\symrefemph{#1}}%
  }{%
    URI:~\detokenize{#2}%
  }%
```

```
  }
  \def\symrefemph#1{%
    \ifcsname textcolor\endcsname
      \textcolor{teal}{#1}%
    \else#1\fi
  }
```

`\compemph` `\compemph@uri` Like `\symrefemph` and `\symrefemph@uri`, but governs the highlighting of components (marked with `\comp` or `\maincomp`) in notations.

`\defemph` `\defemph@uri` Like `\symrefemph` and `\symrefemph@uri`, but governs the highlighting of definienda marked with `\definiendum` (or `\definame`).

`\varemph` `\varemph@uri` Like `\compemph` and `\compemph@uri`, but governs the highlighting of components (marked with `\comp` or `\maincomp`) in the notations of variables.

The second argument to `\varemph@uri` is the *name* of the variable.

## 9.2 Styling Environments and Macros

A variety of environments and macros provided by SᴛEX are *stylable* using the macros `\stexstyle`⟨*name*⟩`[`⟨*style*⟩`]{...}`. These stylable environments and macros bind various of their parameters to macros `\this`⟨*parameter*⟩, which can then be used in the styles.

For example, if we have a `definition` environment that we would want to use to style our `sdefinition`s, we can do (in the simplest case)

```
\stexstyledefinition{\begin{definition}}{\end{definition}}
```

This tells SᴛEX to insert `\begin{definition}` at the beginning of every `sdefinition` environment, and `\end{definition}` at the end.

If we have a environments `theorem` and `lemma`, we probably want the `sassertion` environment to use those for theorems and lemma. We can achieve that by doing

```
\stexstyleassertion[theorem]{\begin{theorem}}{\end{theorem}}
\stexstyleassertion[lemma]{\begin{lemma}}{\end{lemma}}
```

Now if we do `\begin{sassertion}[style=theorem]`, it will wrap the environment with `\begin{theorem}...\end{theorem}`.

Of course, many such statements might have a title, as e.g. in

> **Satz 9.2.1** (Gödel's First Incompleteness Theorem)**.** ...

In `sasssertion`, we can provide that title as optional argument using `title=...`. Before calling the styling provided, `sassertion` will store that title in the macro `\thistitle`, which we can use in the styling. For example, we might prefer to pass it on to the `theorem` environment:

```
\stexstyleassertion[theorem]{\ifx\thistitle\@empty
  \begin{theorem}\else\begin{theorem}[\thistitle]\fi}
  {\end{theorem}}
```

\stexstylemodule
\stexstylecopymodule
\stexstyleinterpretmodule
\stexstylerealization
\stexstylemathstructure
\stexstyleextstructure
\stexstyledefinition
\stexstyleassertion
\stexstyleexample
\stexstyleparagraph
\stexstyleproof
\stexstylesubproof

TODO[14]

Additionally, we can style certain macros, if we want them to produce output. For example, we might (for debuggin or documentation purposes) `\symdecl` to give a short summary of the symbol.

We can achieve that by doing, for example:

```
\stexstylesymdecl[debug]{
  Symbol \thisdeclname~(with arity \thisargs) of type $\thistype$.
}
```

in which case

```
\symdecl{foo}[args=2,type=\mathbb{N},style=debug]
```

will yield

> Symbol foo (with arity ii) of type $\mathbb{N}$.

\stexstyleusemodule
\stexstyleimportmodule
\stexstylerequiremodule
\stexstyleassign
\stexstylerenamedecl
\stexstyleassignMorphism
\stexstylecopymod
\stexstyleinterpretmod
\stexstylerealize
\stexstylesymdecl
\stexstyletextsymdecl
\stexstylenotation
\stexstylevarnotation
\stexstylesymdef
\stexstylevardef
\stexstylevarseq
\stexstylespfsketch
\stexstyleMMTinclude

TODO[15]

## 9.3 Custom CSS for Environments

# Chapter 10

# Additional Packages

## 10.1 NotesSlides Manual

### 10.1.1 Introduction

The notesslides document class is derived from `beamer.cls` [**beamerclass:on**], it adds a "notes version" for course notes that is more suited to printing than the one supplied by `beamer.cls`.

The notesslides class takes the notion of a slide frame from Till Tantau's excellent beamer class and adapts its notion of frames for use in the STEX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the notesslides package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the notesslides class has two modes: *slides mode* and *notes mode* which are determined by the package option.

### 10.1.2 Package Options

The notesslides class takes a variety of class options:

`slides` `notes` The options `slides` and `notes` switch between slides mode and notes mode (see subsection 4.1.3).

`sectocframes` If the option `sectocframes` is given, then for the `sfragment`s, special frames with the `sfragment` title (and number) are generated.

`frameimages` `fiboxed` If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames (see ???). If also the `fiboxed` option is given, the slides are surrounded by a box.

### 10.1.3 Notes and Slides

`frame` (*env.*) Slides are represented with the `frame` environment just like in the beamer class, see [**Tantau:ugbc**] for details.

`note` (*env.*) The notesslides class adds the `note` environment for encapsulating the course note fragments.

Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else LaTeX becomes confused and throws error messages that are difficult to decipher.

By interleaving the `frame` and `note` environments, we can build course notes as shown here:

```
1 \ifnotes\maketitle\else
2 \frame[noframenumbering]\maketitle\fi
3
4 \begin{note}
5   We start this course with ...
6 \end{note}
7
8 \begin{frame}
9   \frametitle{The first slide}
10  ...
11 \end{frame}
12 \begin{note}
13  ... and more explanatory text
14 \end{note}
15
16 \begin{frame}
17  \frametitle{The second slide}
18  ...
19 \end{frame}
20 ...
```

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notestrue` or `\notesfalse` is strongly discouraged however.

We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

The beamer class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

`\inputref*` If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

`nparagraph` (*env.*) There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nparagraph` environment is just an `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

### 10.1.4 Customizing Header and Footer Lines

The notesslides package and class comes with a simple default theme named sTeX that provided by the beamterthemesTeX. It is assumed as the default theme for STEX-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide hight is variable) is


The footer line can be customized. In particular the logos.

`\setslidelogo` The default logo provided by the notesslides package is the STEX logo it can be customized using `\setslidelogo{`⟨*logo name*⟩`}`.

`\setsource` The default footer line of the notesslides package mentions copyright and licensing. In notesslides `\source` stores the author's name as the copyright holder. By default it is the author's name as defined in the `\author` macro in the preamble. `\setsource{`⟨*name*⟩`}` can change the writer's name.

`\setlicensing` For licensing, we use the Creative Commons Attribuition-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[`⟨*url*⟩`]{`⟨*logo name*⟩`}` is used for customization, where ⟨*url*⟩ is optional.

### 10.1.5 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add STEX notes.

\frameimage
\mhframeimage

In this case we can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where ⟨*opt*⟩ are the options of `\includegraphics` from the graphicx package [**CarRah:tpp99**] and ⟨*path*⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.

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

```
1 \frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

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

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

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

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

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

\textwarning

The `\textwarning` macro generates a warning sign: ⚠

### 10.1.6 Ending Documents Prematurely

\prematurestop
\afterprematurestop

For prematurely stopping the formatting of a document, sTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `sfragment` environments as needed. After that – and before the implicit `\end{document}` it calls the internal `\afterprematurestop`, which can be customized to do additional cleanup or e.g. print the bibliography.

`\prematurestop` is useful when one has a driver file, e.g. for a course taught multiple years and wants to generate course notes up to the current point in the lecture. Instead of commenting out the remaining parts, one can just move the `\prematurestop` macro. This is especially useful, if we need the rest of the file for processing, e.g. to generate a theory graph of the whole course with the already-covered parts marked up as an overview over the progress; see `import_graph.py` from the `lmhtools` utilities [**lmhtools:github:on**].

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the sTeX preamble of the course notes file.

### 10.1.7 Global Document Variables

To make document fragments more reusable, we sometimes want to make the content depend on the context. We use **document variables** for that.

\setSGvar
\useSGvar

`\setSGvar{⟨vname⟩}{⟨text⟩}` to set the global variable ⟨*vname*⟩ to ⟨*text*⟩ and `\useSGvar{⟨vname⟩}` to reference it.

\ifSGvar

With`\ifSGvar` we can test for the contents of a global variable: the macro call `\ifSGvar{⟨vname⟩}{⟨val⟩}{⟨ctext⟩}` tests the content of the global variable ⟨*vname*⟩, only if (after expansion) it is equal to ⟨*val*⟩, the conditional text ⟨*ctext*⟩ is formatted.

### 10.1.8 Excursions

In course notes, we sometimes want to point to an "excursion" – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

```
\excursion{founif}{../fragments/founif.en}
  {We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

It generates a paragraph that references the excursion whose source is in the file `../fragments/founif.en.tex` and automatically books the file for the `\printexcursions` command that is used here to put it into the appendix. We will look at the mechanics now.

\excursion

The `\excursion{⟨ref⟩}{⟨path⟩}{⟨text⟩}` is syntactic sugar for

```
\begin{nparagraph}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nparagraph}
```

\activateexcursion
\printexcursion
\excursionref

Here `\activateexcursion{⟨path⟩}` augments the `\printexcursions` macro by a call `\inputref{⟨path⟩}`. In this way, the `\printexcursions` macro (usually in the appendix) will collect up all excursions that are specified in the main text.

Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{⟨label⟩}` for that.

\excursiongroup

Finally, we usually want to put the excursions into an `sfragment` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro: `\excursiongroup[id=⟨id⟩,intro=⟨path⟩]` is equivalent to

```
\begin{note}
\begin{sfragment}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{sfragment}
\end{note}
```

When option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `sfragment` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made. This is a problem of the underlying document-structure package.

Local Variables: mode: latex TeX-master: `../stex-manual` End:

## 10.2 Problem Manual

### 10.2.1 Introduction

The problem package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the problem package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

### 10.2.2 Problems and Solutions

`solutions`
`notes`
`hints`
`gnotes`
`pts`
`min`
`boxed`
`test`

The problem package takes the options `solutions` (should solutions be output?), `notes` (should the problem notes be presented?), `hints` (do we give the hints?), `gnotes` (do we show grading notes?), `pts` (do we display the points awarded for solving the problem?), `min` (do we display the estimated minutes for problem soling). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.

The `boxed` option specifies that problems should be formatted in framed boxes so that they are more visible in the text. Finally, the `test` option signifies that we are in a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

`problem` (*env.*) The main environment provided by the problempackage is (surprise surprise) the `problem` environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem.

**Example 27**
Input:

```
\documentclass{article}
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{sproblem}[id=elefants,pts=10,min=2,title=Fitting Elefants]
    How many Elefants can you fit into a Volkswagen beetle?
    \begin{hint}
      Think positively, this is simple!
    \end{hint}
    \begin{exnote}
      Justify your answer
    \end{exnote}
\begin{solution}
  Four, two in the front seats, and two in the back.
  \begin{gnote}
    if they do not give the justification deduct 5 pts
  \end{gnote}
\end{solution}
\end{sproblem}
\end{document}
```

Output:

**Exercise (Fitting Elefants)**
How many Elefants can you fit into a Volkswagen beetle?

*Lösung:* Four, two in the front seats, and two in the back.

`solution` (*env.*) The `solution` environment can be to specify a solution to a problem. If the package option `solutions` is set or `\solutionstrue` is set in the text, then the solution will be presented in the output. The `solution` environment takes an optional KeyVal argument with the keys `id` for an identifier that can be reference `for` to specify which problem this is a solution for, and `height` that allows to specify the amount of space to be left in test situations (i.e. if the `test` option is set in the `\usepackage` statement).

`hint` (*env.*) The `hint` and `exnote` environments can be used in a `problem` environment to give hints
`exnote` (*env.*) and to make notes that elaborate certain aspects of the problem. The `gnote` (grading
`gnote` (*env.*) notes) environment can be used to document situations that may arise in grading.

`\startsolutions` `\stopsolutions` Sometimes we would like to locally override the `solutions` option we have given to the package. To turn on solutions we use the `\startsolutions`, to turn them off, `\stopsolutions`. These two can be used at any point in the documents.

`\ifsolutions` Also, sometimes, we want content (e.g. in an exam with master solutions) conditional on whether solutions are shown. This can be done with the `\ifsolutions` conditional.

### 10.2.3 Markup for Added-Value Services

The `problem` package is all about specifying the meaning of the various moving parts of practice/exam problems. The motivation for the additional markup is that we can base added-value services from these, for instance auto-grading and immediate feedback.

The simplest example of this are multiple-choice problems, where the `problem` package allows to annotate answer options with the intended values and possibly feedback that can be delivered to the users in an interactive setting. In this section we will give some infrastructure for these, we expect that this will grow over time.

#### Multiple Choice Blocks

`mcb` (*env.*) Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc` macro.

`\mcc` `\mcc[⟨keyvals⟩]{⟨text⟩}` takes an optional key/value argument ⟨*keyvals*⟩ for choice metadata and a required argument ⟨*text*⟩ for the proposed answer text. The following keys are supported

- `T` for true answers, `F` for false ones,
- `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `feedback` for a short feedback text given to the student.

What we see when this is formatted to PDF depends on the context. In solutions mode (we start the solutions in the code fragment below) we get

**Example 28**

Input:

```
\startsolutions
\begin{sproblem}[title=Functions,name=functions1]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}
```

Output:

**Exercise (Functions)**

What is the keyword to introduce a function definition in python?

- ☑ def
- ☐ function[a]
- ☐ fun[b]
- ☐ public static void[c]

---

[a] *that is for C and C++*

[b] *that is for Standard ML*

[c] Noooooooooo *that is for Java*

In "exam mode" where disable solutions (here via `\stopsolutions`)

**Example 29**

Input:

```
\stopsolutions
\begin{sproblem}[title=Functions,name=functions1]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}
```

Output:

**Exercise (Functions)**

What is the keyword to introduce a function definition in python?

- ☐ def
- ☐ function
- ☐ fun
- ☐ public static void

we get the questions without solutions (that is what the students see during the exam/quiz).

### Filling-In Concrete Solutions

The next simplest situation, where we can implement auto-grading is the case where we have fill-in-the-blanks

\fillinsol The \fillinsol macro takes a single argument, which contains a concrete solution (i.e. a number, a string, ...), which generates a fill-in-box in test mode:

**Example 30**

Input:

```
\stopsolutions
\begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
\end{sproblem}
```

Output:

**Exercise (Fitting Elefants)**

How many Elefants can you fit into a Volkswagen beetle? ☐

and the actual solution in solutions mode:

**Example 31**

Input:

```
\begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
\end{sproblem}
```

Output:

**Exercise (Fitting Elefants)**

How many Elefants can you fit into a Volkswagen beetle? ☐

If we do not want to leak information about the solution by the size of the blank we can also give \fillinsol an optional argument with a size: \fillinsol[3cm]{12} makes a box three cm wide.

Obviously, the required argument of \fillinsol can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings "soft comparisons" might be in order. [16]

### 10.2.4 Including Problems

\includeproblem The \includeproblem macro can be used to include a problem from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys title, min, and pts specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the problem environment in the included file.

[16] For the moment we only assume a single concrete value as correct. In the future we will almost certainly want to extend the functionality to multiple answer classes that allow different feedback like im MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` options are set. This allows to give students hints about the estimated time and the points to be awarded.

### 10.2.5 Testing and Spacing

The `problem` package is often used by the `hwexam` package, which is used to create homework assignments and exams. Both of these have a "test mode" (invoked by the package option `test`), where certain information –master solutions or feedback – is not shown in the presentation.

`\testspace` `\testsmallspace` `\testsmallspace` `\testsmallspace` `\testnewpage` `\testemptypage`

`\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: `\testsmallspace`, `\testsmallspace`, `\testsmallspace` give small (1cm), medium (2cm), and big (3cm) vertical space.

`\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

Local Variables: mode: latex TeX-master: `../stex-manual` End:

LocalWords: solutions,notes,hints,gnotes,pts,min,boxed,test gnotes elefants,pts gnote LocalWords: 2,title exnote hint,exnote,gnote ifsolutions mcb keyvals Ttext Ftext LocalWords: Functions,name F,feedback Nooooooooo,feedback 2,title includeproblem LocalWords: elefants.fillin,title

## 10.3 HWExam Manual

### 10.3.1 Introduction

The hwexam package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the problem package. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

### 10.3.2 Package Options

`solutions` `notes` `hints` `gnotes` `pts` `min`

The hwexam package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the problems package (cf. its documentation for a description of the intended behavior).

`multiple`

Furthermore, the hwexam package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test`

Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage`

have an effect: they generate space for the students to solve the given problems. Thus they can be left in the LaTeX source.

### 10.3.3 Assignments

`assignment` (*env.*)
`number`
`title`
`type`
`given`
`due`

This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the `assignment` environment), `title` (for the assignment title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. "quiz", or "homework"), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

### 10.3.4 Including Assignments

`\inputassignment`

The `\inputassignment` macro can be used to input an assignment from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

### 10.3.5 Typesetting Exams

`testheading` (*env.*)
`duration`
`min`
`reqpts`

The `\testheading` takes an optional keyword argument where the keys `duration` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.

```
1 \title{320101 General Computer Science (Fall 2010)}
2 \begin{testheading}[duration=one hour,min=60,reqpts=27]
3   Good luck to all students!
4 \end{testheading}
```

Will result in

Name: Matriculation Number:

## 320101 General Computer Science (Fall 2010)

2026-06-25

**You have one hour (sharp) for the test**;
Write the solutions to the sheet.
The estimated time for solving this exam is 23 minutes, leaving you 37 minutes for revising your exam.
You can reach 23 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -4 points are bonus points.

Different problems test different skills and knowledge, so do not get stuck on one problem.

| | To be used for grading, do not write here | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| prob. | 4.1 | 1.1 | 1.2 | 2.1 | 2.2 | 2.3 | 2.4 | 2.5 | 2.6 |
| total | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 |
| reached | | | | | | | | | |

good luck

[17]

Local Variables: mode: latex TeX-master: `../stex-manual` End:
LocalWords: hwexam solutions,notes,hints,gnotes,pts,min gnotes testemptypage reqpts LocalWords: inputassignment reqpts hour,min 60,reqpts

## 10.4 Tikzinput Manual

image The behavior of the ikzinput package is determined by whether the `image` option is given. If it is not, then the tikz package is loaded, all other options are passed on to it and `\tikzinput{`⟨*file*⟩`}` inputs the TIKZ file ⟨*file*⟩`.tex`; if not, only the graphicx package is loaded and `\tikzinput{`⟨*file*⟩`}` loads an image file ⟨*file*⟩.⟨*ext*⟩ generated from ⟨*file*⟩`.tex`.

The selective input functionality of the tikzinput package assumes that the TIKZ pictures are externalized into a standalone picture file, such as the following one

[17] MK: The first three "problems" come from the stex examples above, how do we get rid of this?

```
\documentclass{standalone}
\usepackage{tikz}
\usetikzpackage{...}
\begin{document}
  \begin{tikzpicture}
     ...
  \end{tikzpicture}
\end{document}
```

The standalone class is a minimal LaTeX class that when loaded in a document that uses the standalone package: the preamble and the `documenat` environment are disregarded during loading, so they do not pose any problems. In effect, an `\input` of the file above only sees the `tikzpicture` environment, but the file itself is standalone in the sense that we can run LaTeX over it separately, e.g. for generating an image file from it.

`\tikzinput` `\ctikzinput`

This is exactly where the tikzinput package comes in: it supplies the `\tikzinput` macro, which – depending on the `image` option – either directly inputs the TIKZ picture (source) or tries to load an image file generated from it.

Concretely, if the `image` option is not set for the tikzinput package, then `\tikzinput[`⟨*opt*⟩`]{`⟨*file*⟩`}` disregards the optional argument ⟨*opt*⟩ and inputs ⟨*file*⟩`.tex` via `\input` and resizes it to as specified in the `width` and `height` keys. If it is, `\tikzinput[`⟨*opt*⟩`]{`⟨*file*⟩`}` expands to `\includegraphics[`⟨*opt*⟩`]{`⟨*file*⟩`}`.

`\ctizkinput` is a version of `\tikzinput` that is centered.

`\mhtikzinput` `\cmhtikzinput`

`\mhtizkinput` is a variant of `\tikzinput` that treats its file path argument as a relative path in a math archive in analogy to `\inputref`. To give the archive path, we use the `mhrepos=` key. Again, `\cmhtizkinput` is a version of `\mhtikzinput` that is centered.

`\libusetikzlibrary`

Sometimes, we want to supply archive-specific TIKZ libraries in the `lib` folder of the archive or the `meta-inf/lib` of the archive group. Then we need an analogon to `\libinput` for `\usetikzlibrary`. The stex-tikzinput package provides the `libusetikzlibrary` for this purpose.

Local Variables: mode: latex TeX-master: `../stex-manual` End:

# Part III
# Documentation

# Chapter 11

# Utilities

Various utility methods

`\stex_ignore_spaces_and_pars:`

Ignores space characters and empty lines (which would close the current paragraph)

`\stex_undefine:N` Locally undefines a macro
`\stex_undefine:c`

`\stex_str_if_starts_with_p:nn` ⋆ `\stex_str_if_starts_with:nn {`⟨*str1*⟩`} {`⟨*str2*⟩`}`
`\stex_str_if_starts_with:nn`*TF* ⋆

Tests if ⟨*str1*⟩ starts with ⟨*str2*⟩.

`\stex_str_if_ends_with_p:nn` ⋆ `\stex_str_if_ends_with:nn {`⟨*str1*⟩`} {`⟨*str2*⟩`}`
`\stex_str_if_ends_with:nn`*TF* ⋆

Tests if ⟨*str1*⟩ ends with ⟨*str2*⟩.

`\stex_deactivate_macro:Nn` `\stex_deactivate_macro:Nn \macro {`⟨*scope*⟩`}`
`\stex_reactivate_macro:N`

redefines `\macro` to throw an error, that the `\macro` is only allowed in ⟨*scope*⟩. This allows for more informative error messages than `undefined control sequence`.

`\stex_reactivate_macro:N` makes the macro valid again.

## 11.1 `kpsewhich` and Environment Variables

`\stex_kpsewhich:Nn` `\stex_kpsewhich:Nn \macro {`⟨*args*⟩`}`

Calls "`kpsewhich` ⟨*args*⟩" and stores the result in `\macro`,

**TEXhackers note:** (Usually) does not require `shell-escape`, since by default, `kpsewhich` is in the list of "allowed" commands.

\stex_get_env:Nn `\stex_get_env:Nn \macro {⟨envvar⟩}`

Stores the value of the environment variable ⟨*envvar*⟩ in `\macro`.

## 11.2 Logging

\stex_debug:nn `\stex_debug:nn {⟨prefix⟩} {⟨msg⟩}`

Logs the debug message `{⟨msg⟩}` under the prefix `{⟨prefix⟩}`. A message is shown if its prefix is in a list of prefixes given either via the package option `debug=⟨prefixes⟩` or the environment variable `STEX_DEBUG=⟨prefixes⟩`, where the latter overrides the former.

## 11.3 File Paths

Since sTeX needs to handle files from various *math archives* in a user-file-system-independent manner in arbitrary *MathHub* directories, we primarily use *absolute* file paths, for operations such as `\inputref`, `\usemodule`, `\importmodule`, etc.

We therefore provide macros to explicitly handle file paths independent of the underlying operating system, resolving and canonicalizing `..` segments and relative paths, etc.

\stex_file_set:Nn
\stex_file_set:(No|Ne)
`\stex_file_set:Nn \macro {⟨string⟩}`

represents an already canonicalized file path string as a LaTeX3 sequence and stores it in `\macro`.

\stex_file_resolve:Nn
\stex_file_resolve:(No|Ne)
`\stex_file_resolve:Nn \macro {⟨string⟩}`

resolves and canonicalizes the file path string ⟨*string*⟩ and stores the result in `\macro`.

\stex_file_use:N ⋆ expands to a string representation of the given file path (using `/` as separator, regardless of file system).

\stex_file_split_off_ext:NN `\stex_file_split_off_ext:NN \target \source`

splits off the file extension of `\source` and stores the resulting file path in `\target`.

\stex_file_split_off_lang:NN `\stex_file_split_off_lang:NN \target \source`

splits off *both* the file extension *and* language component of `\source` and stores the resulting file path in `\target`; e.g. if `\source` is `foo.en.tex`, `\target` will be `foo`. Explicitly checks that the segment before the file extension is a valid language identifier; otherwise, it will not be stripped.

\c_stex_pwd_file
\c_stex_main_file
represent the *parent working directory* and absolute path to the top-level `.tex` file currently being processed (as absolute, canonicalized paths)

`\c_stex_home_file` represents the absolute path of the user's home directory (as absolute, canonicalized path)

`\g_stex_current_file` the current file (as absolute, canonicalized path)

`\stex_input_with_hooks:Nn` `\stex_input_with_hooks:Nn \document-URI {⟨file string⟩}`
`\stex_input_with_hooks:Ne`
Inputs the given file (as a string) by passing it on to `\input`, setting `\g_stex_current_file`, `\l_stex_current_language_str` and `\l_stex_current_document_uri`, and reverting them again afterwards

`\stex_with_file_hooks:Nnn` `\stex_with_file_hooks:Nnn \document-URI {⟨file string⟩} {⟨code⟩}`
sets `\g_stex_current_file`, `\l_stex_current_language_str` and `\l_stex_current_document_uri`, executes ⟨*code*⟩, and reverts them again afterwards

`\stex_if_file_absolute_p:N` ⋆
`\stex_if_file_absolute:NTF` ⋆
tests whether the given file path (represented as a canonicalized LaTeX3 sequence) is absolute.

`\stex_if_file_starts_with:NNTF` `\stex_if_file_starts_with:NN \child \parent`
tests whether the file path `\child` is a child of `\parent`.

## 11.4 Group-like Behaviours

Macros for mimicking the behaviour of TeX groups without actually opening a TeX group. This is relevant for two purposes:

- Temporarily redefining a macro, doing a bunch of stuff, and then reverting it to its previous definition; that is what a "*pseudogroup*" does.
- Defining macros at a specific group level further up. That is what a "*metagroup*" does – code wrapped in `\stex_metagroup_do_in:n` will be repeatedly executed in `\aftergroup` calls, up to the group level of the innermost metagroup. That allows for e.g. declaring new symbols in `sdefinition` paragraphs that are valid in the entire `mathstructure` or `module`.

`\stex_pseudogroup:nn` `\stex_pseudogroup:nn {⟨code⟩} {⟨reset code⟩}`
will *expand* ⟨*reset code*⟩ first, then process ⟨*code*⟩, then process the expanded ⟨*reset code*⟩.

`\stex_pseudogroup_restore:N ⋆ \stex_pseudogroup_restore:N \macro`

will expand to code that will redefine `\macro` as its current definition. Only works for *token lists* and similar macros that do not take arguments.

In combination, that allows for things like

**Example 32**

```
% \stex_pseudogroup:nn{
% \def \foo {..}
% ...
% }{\stex_pseudogroup_restore:N \foo}
%
```

`\stex_pseudogroup_with:nn` `\stex_pseudogroup_with:nn {⟨macro-list⟩} {⟨code⟩}`

will store the definitions of the macros in ⟨*macro-list*⟩, execute ⟨*code*⟩, and then revert the macros to their previous definitions without opening a TEX group. TODO: this might be unnecessary? Check/profile

`\stex_metagroup_new:` opens a new *metagroup*. All code executed in `\stex_metagroup_do_in:n` will survive up to the current group level. Metagroups can be nested.

`\stex_metagroup_do_in:n` `\stex_metagroup_do_in:e` Executes the given code in the current metagroup. If there is no metagroup currently, the code will simply be executed once, so there is little danger in calling this spuriously.

**TEXhackers note:** This repeatedly stores the provided code in a macro and uses `\aftergroup` until the group level of the current metagroup is reached.

## 11.5 Key Handling

Key handling mechanisms on top of l3keys for inheriting key sets and initializing the associated macros

`\stex_keys_define:nnnn` `\stex_keys_define:nnnn {⟨id⟩} {⟨init⟩} {⟨spec⟩} {⟨exts⟩}`

defined the key set ⟨*id*⟩ with the l3keys-style key-parsing code {⟨*spec*⟩}; ⟨*init*⟩ is called every time before parsing and is intended to e.g. clear the relevant macros. ⟨*exts*⟩ may be a comma-separated list of key set ids from which this key set inherits.

`\stex_keys_set:nn` Like `\keys_set:nn`, but additionally calling the relevant initialization code.

## 11.6 Languages

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

Two inverse property lists; `\c_stex_languages_prop` maps language abbreviations (e.g. `en`, `de`) to their (`babel`) names (e.g. `english`, `ngerman`), `\c_stex_language_abbrevs_prop` does the inverse.

`\l_stex_current_language_str`

The language of the current file; may change, if a file with a different language is imported.

## 11.7 Styling

`\stex_new_stylable_cmd:nnnn`
`\stex_style_apply:`

`\stex_new_stylable_cmd:nnnn {`⟨*macroname*⟩`} {`⟨*argument spec*⟩`} {`⟨*definition*⟩`} {`⟨*default*⟩`}`

Defines a new stylable command `\macroname` with ⟨*argument spec*⟩ (passed on to `\NewDocumentCommand`) with ⟨*definition*⟩ as its macro body and default typesetting style ⟨*default*⟩.

⟨*default*⟩ may be empty. ⟨*definition*⟩ should at some point use `\stex_style_apply:` to do the typesetting according to the chosen style for this macro, which by default will be defined as ⟨*default*⟩.

Also generates the macro `\stexstyle`⟨*macroname*⟩`[`⟨*name*⟩`]{`⟨*code*⟩`}` to define a new typesetting style for `\macroname`.

`\stex_new_stylable_env:nnnnnnn`

`\stex_new_stylable_env:nnnnnnn {`⟨*environment name*⟩`} {`⟨*argument spec*⟩`} {`⟨*begin code*⟩`} {`⟨*end code*⟩`} {`⟨*default begin*⟩`} {`⟨*default end*⟩`} {`⟨*prefix*⟩`}`

Defines a new stylable environment ⟨*prefix*⟩⟨*environment name*⟩ with ⟨*argument spec*⟩ (passed on to `\NewDocumentEnironment`) with ⟨*begin code*⟩ and ⟨*end code*⟩ as its macro body and default typesetting style dictated by ⟨*default begin*⟩ and ⟨*default end*⟩.

⟨*begin code*⟩ and ⟨*end code*⟩ should at some point use `\stex_style_apply:` to do the typesetting according to the chosen style for this environment.

Also generates the macro `\stexstyle`⟨*environmentname*⟩`[`⟨*name*⟩`]{`⟨*begin*⟩`}{`⟨*end*⟩`}` to define a new typesetting style for `environment name`.

A prefix `s` is e.g. used to generate the environment `sparagraph` and the macro `\stexstyleparagraph`.

## 11.8 Persisting Dependencies

`\stex_persist:n`
`\stex_persist:e`
`\loadsms`

Writes the given code in the `.sms`-file (if smsmode is on)

# Chapter 12

# HTML Output

Macros for dealing with HTML output.

`\stex@backend` Which engine is running; possible values are `pdflatex`, `rustex`, `tex4ht` (TODO) and `latexml` (TODO).

`\stex_if_html_backend_p:` ⋆
`\stex_if_html_backend:`*TF* ⋆
LaTeX3 conditional testing whether the engine running compiles to HTML.

`\ifstexhtml` ⋆ LaTeX2 conditional testing whether the engine running compiles to HTML.

`\stex_if_do_html_p:` ⋆
`\stex_if_do_html:`*TF* ⋆
Whether to (locally) produce HTML output. May be turned off locally even if the backend produces HTML; e.g. when parsing dependencies in `\importmodule` or `\usemodule`.

`\stex_suppress_html:n` temporarily suppresses HTML output when processing the provided code.

`\stex_annotate:nn` `\stex_annotate:nn {`⟨*keyvals*⟩`} code`

attaches the provided comma-separated key-value-pairs (as `key=val`) as HTML attributes to the HTML node(s) generated from ⟨*code*⟩. If ⟨*code*⟩ results in multiple nodes, this needs to insert a new `<div>`, `<span>` or `<mrow>` node for the attributes. Multiple nested calls to `\stex_annotate:nn` may be merged into a single node.

`stex_annotate_env` (*env.*) like `\stex_annotate:nn`, but as an environment; i.e. `\begin{stex\_annotate\_env}{`⟨*keyvals*⟩`}{`⟨ is equivalent to `\stex_annotate:nn{`⟨*keyvals*⟩`}{`⟨*code*⟩`}`

`\stex_html_node:nnn` `\stex_html_node:nnn {`⟨*tag*⟩`} {`⟨*keyvals*⟩`} code`

like `\stex_annotate:nn`, but makes sure to wrap ⟨*code*⟩ in a new `<tag>` node.

`stex_env_node` (*env.*) like `\stex_html_node:nnn`, but as an environment; i.e. `\begin{stex\_env\_node}{`⟨*tag*⟩`}{`⟨*keyva*

is equivalent to `\stex_html_node:nnn{⟨tag⟩}{⟨keyvals⟩}{⟨code⟩}`

`\stex_annotate_invisible:nn` `\stex_annotate_invisible:nn {⟨keyvals⟩} code`
`\stex_annotate_invisible:n`

like `\stex_annotate:nn`, but additionally adds the key values `data-ftml-invisible="true"` and `style="display:none;"`. `\stex_annotate_invisible:n` does not take additional attribute-value-pairs.

`\STEXinvisible` public wrapper for `\stex_annotate_invisible:n`.

`\stex_css_link:n` `\stex_css_link:n {⟨url⟩}`

inserts a `<link rel="stylesheet" href="⟨url⟩">` node in the header of the generated HTML.

`\stex_css_literal:n` `\stex_css_literal:n {⟨text⟩}`

inserts a `<style>⟨text⟩</style>` node in the header of the generated HTML.

`\_stex_annotate_force_break:n`

should guarantee that the HTML nodes generated by the provided code are not merged with any outer nodes with respect to attached attributes.

`\mmlintent` `\mmlintent {⟨value⟩} {⟨code⟩}`
`\mmlarg`

annotates ⟨*code*⟩ with the provided MathML *intent* attribute (or *intent* `arg` attribute, respectively).

and style patching:

```
⟨@@=stex_styles⟩
\stex_if_html_backend:TF{
  \cs_new_protected:Nn \stex_style_apply: {}
  \cs_new_protected:Nn \__stex_styles_patch:nnnn {}
  \stex_keys_define:nnnn{ csspatch }{
    \str_clear:N \l_stex_keys_cls_id
    \str_clear:N \l_stex_keys_counter_id
    \str_clear:N \l_stex_keys_counter_parent
  }{
    counter    .str_set_x:N  = \l_stex_keys_counter_id,
    parent     .str_set_x:N  = \l_stex_keys_counter_parent,
    unknown    .code:n       = {
      \exp_args:NNo \str_set:Nn \l_stex_keys_cls_id {\l_keys_key_tl}
    }
  }{}
  \cs_new_protected:Npn \__stex_styles_css_patch:nnnn #1 #2 {
    \stex_keys_set:nn{ csspatch }{ #2 }
    \group_begin:
    \catcode`\ =12\relax
    \catcode`^^J=12\relax
    \__stex_styles_patch_i:nnn {#1}
  }
```

```
23
24  \seq_new:N \g__stex_styles_counters_seq
25
26  \cs_new:Nn \__stex_styles_patch_html_c: {
27    \stex_html_literal:n{
28      <span~data-ftml-counter="\l_stex_keys_counter_id"~style="display:none;"~
29        data-ftml-counter-parent="\l_stex_keys_counter_parent"
30      ></span>
31    }
32  }
33
34  \cs_new:Nn \__stex_styles_patch_html: {
35    \stex_html_literal:n{
36      <span~data-ftml-style="\l_stex_keys_cls_id"~style="display:none;"~
37        data-ftml-counter="\l_stex_keys_counter_id"
38      ></span>
39    }
40  }
41
42  \cs_new_protected:Nn \__stex_styles_patch_i:nnn {
43    \str_if_empty:NTF \l_stex_keys_cls_id {
44      \str_set:Nn \l_stex_keys_cls_id { #1 }
45    }{
46      \str_set:Ne \l_stex_keys_cls_id { #1-\l_stex_keys_cls_id }
47    }
48    \exp_args:No \str_case:nnF \l_stex_keys_counter_parent {
49      {}{}
50      {part}{\str_set:Nn \l_stex_keys_counter_parent { 0 }}
51      {chapter}{\str_set:Nn \l_stex_keys_counter_parent { 1 }}
52      {section}{\str_set:Nn \l_stex_keys_counter_parent { 2 }}
53      {subsection}{\str_set:Nn \l_stex_keys_counter_parent { 3 }}
54      {subsubsection}{\str_set:Nn \l_stex_keys_counter_parent { 4 }}
55      {paragraph}{\str_set:Nn \l_stex_keys_counter_parent { 5 }}
56      {subparagraph}{\str_set:Nn \l_stex_keys_counter_parent { 6 }}
57    }{
58      \msg_error:nnx{stex}{error/notasection}\l_stex_keys_counter_parent
59    }
60    \str_set:Ne \l__stex_styles_first_str { &~.ftml-title-paragraph~ { display:inline-block
61    \str_if_empty:NF \l_stex_keys_counter_id {
62      %\str_put_left:Ne \l__stex_styles_first_str { counter-increment:~ ftml-\l_stex_keys_c
63      \exp_args:NNo \seq_if_in:NnF \g__stex_styles_counters_seq \l_stex_keys_counter_id {
64        \seq_gpush:No \g__stex_styles_counters_seq \l_stex_keys_counter_id
65        \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
66          \__stex_styles_patch_html_c:
67          % in body
68        }{
69          \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html_c:
70          % in preamble
71        }
72      }
73      \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
74          \__stex_styles_patch_html:
75          % in body
76      }{
```

```
        \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html:
        % in preamble
      }
    }
    \exp_args:Ne \stex_css_literal:n { .ftml-\l_stex_keys_cls_id { \l__stex_styles_first_st

    % TODO export counter reset somehow

    \group_end:
  }
}{
  \cs_new_protected:Nn \__stex_styles_patch:nnnn {
    \str_if_empty:nTF {#2}{
      \tl_set:cn{__stex_styles_style_#1_start:}{#3}
      \tl_set:cn{__stex_styles_style_#1_end:}{#4}
    }{
      \tl_set:cn{__stex_styles_style_#1_#2_start:}{#3}
      \tl_set:cn{__stex_styles_style_#1_#2_end:}{#4}
    }
  }
  \cs_new_protected:Nn \__stex_styles_css_patch:nnnn {}
}
```

# Chapter 13

# Math Archives

`\c_stex_mathhub_files`
`\mathhub`

represents the comma-separated, absolute path *strings* of the user's MathHub directories. If the `\mathhub` macro is set when the `stex` package is loaded, this one is used for finding archives. Otherwise it is determined from the `MATHHUB` environment variable, if set, or from the path in `<HOME>/.stex/mathhub.path`, if existent, or set to `<HOME>/MathHub` if all else fails. In all cases, `\mathhub` will be defined.

MathHub directories are scanned in order; i.e. earlier paths are prioritized over later ones.

A *math archive* is represented as a triple `{<archive id>}{<base uri>}{<file path string>}`, where an invariant is that ⟨*file path string*⟩ is of the form `<mh>/<archive id>` for some directory `<mh>` in `\c_stex_mathhub_files`.

Such a triple is stored in the macro `\c_stex_mathhub_<id>_archive` when the archives metadata is first loaded.

`\stex_archive_id:N` ⋆ expands to the id of the given archive (a macro defined as a triple as described above)

`\stex_archive_base:N` ⋆
`\stex_archive_base:n` ⋆

expands to the base URI of the given archive; either given as a macro defined as a triple as described above, or as the archive's id. The latter requires the archive to be loaded beforehand!

`\stex_archive_path:N` ⋆
`\stex_archive_path:n` ⋆

expands to the file path *as a string* of the given archive; either given as a macro defined as a triple as described above, or as the archive's id. The latter requires the archive to be loaded beforehand!

`\stex_in_archive:nn` `\stex_in_archive:nn {`⟨*archive id*⟩`} {`⟨*code*⟩`}`

Temporarily sets the current archive `\l_stex_current_archive` to ⟨*archive id*⟩, executes ⟨*code*⟩ and reverts back to the previous archive. Passes the id of the *now current* archive to ⟨*code*⟩ as `#1`: If the first argument is empty, this will be the current archive's id without changing it. Otherwise, the archive with the desired id will be loaded; throws an error, if it does not exist.

`\c_stex_no_archive_str`
`\c_stex_no_archive`

The archive ID `\c_stex_no_archive_str`=`no/archive` is used for documents and other content that is not contained in a *math archive*. `\c_stex_no_archive` is the corresponding archive triple.

`\c_stex_main_archive`
`\l_stex_current_archive`

Point to the main file's archive and the *current* archive, respectively, if existent; undefined otherwise.

`\stex_source_path:n` ⋆
`\stex_source_path:nn` ⋆

`\stex_source_path:n {`⟨*relative path*⟩`}`

expands to the absolute file path (string) of the given ⟨***relative path***⟩ relative to the `source` directory of the current archive – or relative to the current file, if we are not in an archive. In the latter case, the file path will *not* be canonicalized as to remain expandable.

The variant `\stex_source_path:nn` takes the ID of an archive as first argument (instead of the current archive).

# Chapter 14

# URIs

There are kinds of URIs used in iMmt/OMDoc:

- *Base URIs*, i.e. namespaces,
- *Archive URIs* of the form `<base uri>?a=<archive id>`,
- *Document URIs* of the form `<archive uri>[&p=<path>]&d=<name>&l=<language>`,
- *Document element URIs* of the form `<document uri>&e=<name>`,
- *Module URIs* of the form `<archive uri>[&p=<path>]&m=<name>`, and
- *Declaration URIs* of the form `<module uri>&s=<name>`.

We do not need all of these in sTeX itself: the *base uri* of any URI is uniquely determined by the archive ID, and we conceptually merge document URIs and document element URIs.

We therefore represent document (element) URIs as 5-tuples `{<archive id>}{<path>}{<name>}{<language>}{<element name>}`, where `<path>` and `<element name>` may be empty; module URIs as triples `{<archive id>}{<path>}{<name>}`, and declaration URIs as 4-tuples `{<archive id>}{<path>}{<module name>}{<name>}`.

`\stex_use_archive_uri:n` ⋆ expands to the string representation of the archive URI of the archive with the given ID. Requires, that the archive has been loaded first

## 14.1 Document URIs

`\stex_use_document_uri:N` ⋆
`\stex_use_document_uri:n` ⋆ expands to the string representation of the document uri (represented as a macro containing a 5-tuple as above). The variant `\stex_document_uri:n` expects a 5-tuple directly.

`\stex_document_uri_archive:N` ⋆

expands to the archive ID of the document URI

`\stex_document_uri_path:N` ⋆

expands to the path of the document URI (possibly empty)

`\stex_document_uri_name:N` ⋆

expands to the document name of the document URI

`\stex_document_uri_language:N` ⋆

expands to the language of the document URI

`\stex_document_uri_element:N` ⋆

expands to the element name of the document (element) URI (possibly empty)

`\stex_document_uri_with_language:Nn` ⋆

expands to the document URI with the given language

`\stex_new_document_element_uri:n` ⋆

expands to a new document element URI in the current document

`\stex_document_uri_from_archive_file:Nn` `\stex_uri_from_archive_file:Nn \macro {`⟨*relative path*⟩`}`

Constructs the document URI of the file ⟨*relative path*⟩ in the current archive and stores it in `\macro`

`\c_stex_main_document_uri`
`\l_stex_current_document_uri`

store the document URIs of the top-level file being processed and the *current* file, respectively.

## 14.2 Module URIs

`\stex_use_module_uri:N` ⋆
`\stex_use_module_uri:n` ⋆

expands to the string representation of the module uri (represented as a macro containing a triple as above). The variant `\stex_module_uri:n` expects a triple directly.

`\stex_module_uri_archive:N` ⋆

expands to the archive ID of the module URI

`\stex_module_uri_path:N` ⋆
`\stex_module_uri_path:n` ⋆

expands to the path of the module URI (possibly empty)

`\stex_module_uri_name:N` ⋆
`\stex_module_uri_name:n` ⋆

expands to the name of the module URI

`\stex_module_uri_split_name:NNN`
`\stex_module_uri_split_name:NNn`

sets `#1` as the parent module of the module URI and `#2` as the last name

`\stex_module_uri_as_qm:n` ⋆

`\stex_new_module_uri:n` ⋆ `\stex_new_module_uri:n {`⟨*name*⟩`}`

expands to a new module URI triple with the given ⟨*name*⟩ based on the current document URI or module URI, if existent

`\stex_uri_from_pair:Nnn` `\stex_uri_from_pair:Nnn {`⟨*archive id*⟩`} {`⟨*module path*⟩`}`

resolves the pair `[`⟨*archive id*⟩`]{`⟨*module path*⟩`}` to a module URI and stores the result in `#1`

`\stex_uri_from_pair:Nn` like `\stex_uri_from_pair:Nnn`, but takes a token list as argument that may or may not be of the form `[archive]module`

## 14.3 Symbol URIs

`\stex_use_symbol_uri:N` ⋆
`\stex_use_symbol_uri:n` ⋆

expands to the string representation of the module uri (represented as a macro containing a 4-tuple as above). The variant `\stex_symbol_uri:n` expects a 4-tuple directly.

`\stex_new_symbol_uri:n` ⋆ `\stex_new_symbol_uri:n {`⟨*name*⟩`}`

expands to a new symbol URI tuple with the given ⟨*name*⟩ based on the current module URI, which is assumed to exist(!).

`\stex_new_symbol_uri:nn` ⋆

expands to a new symbol URI tuple with the given ⟨*name*⟩ based on the *given* module URI (either as macro or as tuple), which is assumed to exist(!).

`\stex_symbol_uri_archive:N` ⋆

expands to the archive ID of the symbol URI

`\stex_symbol_uri_path:N` ⋆ expands to the path of the symbol URI (possibly empty)

`\stex_symbol_uri_module:N` ⋆
`\stex_symbol_uri_module:n` ⋆

expands to the URI of the containing module of the symbol URI

`\stex_symbol_uri_name:N` ⋆
`\stex_symbol_uri_name:n` ⋆

expands to the name of the symbol URI

# Chapter 15

# Documents

\STEXftmllink
\STEXlinkftml
\STEXsetlinkftml

\STEXlinkftml inserts the following link/disclaimer:

> *The FTML version of this document can be found at*
> *http://unknown.source?a=no/archive&p=/home/jazzpirate/work/Software/sTeX/sTeX/doc&d=stex-doc&l=en*

The precise text can be set using \STEXsetlinkftml{⟨*text*⟩}; The link on its own can be inserted using \STEXftmllink.

\stexdoctitle

\stexdoctitle {⟨*title*⟩}

Sets ⟨*title*⟩ to be the title of the document. Only the first call of this command does something.

## 15.1 Sectioning

`\l_stex_current_section_level_int`
`\l_stex_current_section_level_str`

integer keeping track of the current sectioning level:

0 part

1 chapter

2 section

3 subsection

4 subsubsection

5 paragraph

>5 subparagraph

`\setsectionlevel` sets `\l_stex_current_section_level_int` to the corresponding integer value. `\l_stex_current_section_level_str` stores a string representation of the same value, but will initially contain `document`.

`\currentsectionlevel` `\Currentsectionlevel`

will leave the current section level as text in the token input. The variant `\Currentsectionlevel` will produce the capitalized version.

If the xspace package is loaded, this will insert an `\xspace` afterwards.

In HTML mode, this will insert an annotation, so it can be adapted dynamically.

`sfragment` (*env.*)

A section at the current section level; preferred over the primitives `\section`, `\subsection`, etc., because a) environments are better suited for such structuring, and b) it allows for adapting the inserted section header based on document context, rather than it being hardcoded – e.g. what's a subsection in one document may be a section in another.

`titlefragment` (*env.*)

like `sfragment`, but will use `\maketitle` instead of (one of) the sectioning macros, if no title has been used yet.

`blindfragment` (*env.*)

skips one sectioning level in its body so that e.g. subsection 0.1 can be used before the first section

`\skipfragment`

Skips one counter at the current sectioning level; e.g. increase the subsection counter to 2 if called immediately after `\section` (or, rather, after `\begin{sfragment}`).

`\setsectionlevel`

`\setsectionlevel {`⟨*name*⟩`}`

Sets the current section level to the one specified (e.g. `\setsectionlevel{subsection}`). Should ideally be called at most once in the preamble of a document.

## 15.2 Inter-Document References

`[id=foo]` sets the current fragment's (e.g. `Definition 3.1 (Foo)`) name to `foo`, resulting in a DocumentElementURI `..&e=foo`. It then delegates to `\label{...&e=foo}` (which defines `\r@...&e=foo`), stores `:::&e=foo` in `\g_stex_sref_label_foo`, and writes `\STeXInternalSRefLabel{foo}{..&e=foo}{definition.3.1}{Foo}` to the sref file.

## 15.3 Inputting From MathHub Resources

`\inputref` `\inputref [⟨archive id⟩] {⟨filepath⟩}`

Inputs the referenced file in the referenced archive (or current archive, if empty) in a tex group, while setting all the relevant macros for the current archive, document URI, etc.

In HTML, will instead just insert an annotation, so that the HTML of the referenced document can be dynamically inserted instead.

`\mhinput` `\inputref [⟨archive id⟩] {⟨filepath⟩}`

Like `\inputref`, but *always* actually inputs the referenced file (also in HTML mode!) and without opening a tex group.

`\ifinputref` LaTeX2 conditional; is true, if we currently are in an `\inputref` or `\mhinput`.

`\IfInputref` `\IfInputref {⟨true⟩} {⟨false⟩}`

Executes ⟨*true*⟩ if we currently are in an `\inputref` or `\mhinput`, otherwise executes ⟨*false*⟩.

In HTML, *both* branches are executed(!) and inserted in the HTML with corresponding attributes, so that the relevant parts can be shown or hidden.

`\libinput` `\libinput [⟨archive id⟩] {⟨file name⟩}`

inputs all files ⟨*file name*⟩`.tex` in the `lib` directories of the current archive. Will throw an error if we are not in an archive or no suitable file is found.

`\libusepackage` `\libusepackage [⟨package options⟩] {⟨package name⟩}`

inputs all packages ⟨*package name*⟩`.sty` in the `lib` directories of the current archive. Will throw an error if we are not in an archive, or there is not *exactly one* file ⟨*package name*⟩`.sty` somewhere in the `lib` directories.

Note that unlike `\libinput`, the optional argument does *not* refer to an archive ID, since this would conflict with package options.

Will give a spurious warning that package `mathhub/directory/foo` was requested, but package `foo` was found. This is a side-effect of using absolute file paths, which is necessary to make the package findable in the first place.

`\libusetikzlibrary` `\libusetikzlibrary [⟨archive id⟩] {⟨name⟩}`

like `\libusepackage`, but for `\usetikzlibrary`.

\addmhbibresource `\addmhbibresource [⟨archive id⟩] {⟨file name⟩}`

Calls `\addbibresource` on all ⟨*file name*⟩`.bib` files in the `lib` directories of the current archive. Will throw an error if we are not in an archive or no suitable file is found.

`\mhgraphics`
`\cmhgraphics`

`\mhgraphics` adds the `archive` key to the optional arguments of `\includegraphics` to allow for using images in math archives (relative to its source directory). `\cmhgraphics` additionally wraps the image in a `\begin{center}`.

Only defined if the graphicx package is loaded (but may be loaded after the stex package).

`\lstinputmhlisting`
`\clstinputmhlisting`

like `\mhgraphics`, but for `\lstinputlisting`.

Only defined if the listings package is loaded (but may be loaded after the stex package).

`\mhtikzinput`
`\cmhtikzinput`

like `\mhgraphics`, but for `\tikzinput`.

Only defined if the tikzinput package is loaded (but may be loaded after the stex package).

# Chapter 16

# Modules

`\l_stex_current_module_uri` stores the URI of the current module, if existent

`\l_stex_all_modules_seq` stores the URIs of all modules *currently in scope*.

`smodule` (*env.*) Parses the optional arguments, (if relevant) inserts HTML annotations, and then delegates to `\stex_module_setup:n`

`\stex_module_setup:n` Sets up a new module with the given name by checking whether it is a nested module, and if not, has a signature etc. Loads signature and metatheory, if necessary TODO: deprecate in favor of every document being a module

`\stex_module_setup_top_nosig:n`

Sets up a new top-level module with the given name and no signature. Does not load a meta theory. TODO: deprecate in favor of every document being a module

`\stex_close_module:` closes the current module.

`\stex_every_module:n` adds code to be executed every time a new module is opened (e.g. activating certain macros).

`\stex_do_up_to_module:n` `\stex_do_up_to_module:e` Execute code in the current module (i.e. as if the `\begin{`⟨*smodule*⟩`}` was the current tex group)

`\stex_execute_in_module:n` `\stex_execute_in_module:e` Execute code in the current module (i.e. as if the `\begin{`⟨*smodule*⟩`}` was the current tex group) and also adds it to the initialization code of the module; i.e. exports all macros defined.

`\stex_if_in_module_p:` ⋆
`\stex_if_in_module:TF` ⋆

conditional whether we currently are in a module. TODO: deprecate in favor of every document being a module

`\stex_if_module_exists_p:N` ⋆
`\stex_if_module_exists:NTF` ⋆
`\stex_if_module_exists_p:n` ⋆
`\stex_if_module_exists:nTF` ⋆

conditional whether a module with the given URI exists

`\stex_activate_module:N`
`\stex_activate_module:n`

`\setmetatheory` `\setmetatheory [⟨archive id⟩] {⟨module⟩}`

sets the metatheory of all subsequent modules to the specified one

`\STEXexport` `\STEXexport {⟨code⟩}`

executes the provided code every time the current module is opened (including immediately). Processes its content in the `\ExplSyntaxOn` category code scheme.

`\stex_structural_feature_module:nn`
`\stex_structural_feature_module_end:`

## 16.1 SMS Mode

sTeX has to extract formal content (i.e. modules and their symbols) from LaTeX-files, that may otherwise contain arbitrary code, including macros that may not be defined unless the file is fully processed by TeX. Those modules and symbols also may depend on other modules that have not yet been loaded. The naive way to achieve this, which would be to just suppress output (e.g. by storing it in a box register) and then `\input` the required file, does not work thanks to TeX's limited *file stack*, which would overflow quickly for modules that have a deeply nested list of dependencies.

To solve those problems, sTeX reads dependencies in what we call *sms mode*, which can be summarized thusly:

- In a first pass, we parse the file token by token, ignoring everything other than a select list of macros and environments that introduce dependencies (such as `\importmodule` and `\begin{smodule}[sig=...]`). Instead of loading those, we remember them for later.
- After the file as been fully parsed thusly, the dependencies found are loaded, again in sms-mode.
- In a second pass, we parse the file *again* in the same way, but this time execute all macros that are explicitly allowed in sms mode, such as `\importmodule`, `\symdecl`, `\notation`, `\symdef`, etc.

- all this parsing happens additionally in a `\setbox\throwaway\vbox{...}`-block to suppress any accidental output.

This means that TEX's input stack never grows by more than +1, but still behaves *as if* the dependencies were loaded recursively, at the detriment of being somewhat slow.

`\stex_sms_allow:N` registers the provided macro to be allowed in sms mode.

This only works, if the macro takes no arguments and/or does not touch the subsequent tokens.

`\stex_sms_allow_escape:N` registers the provided macro to be allowed in sms mode.

If the macro is subsequently encountered in sms-mode, parsing is halted and it can process arguments as desired. It then needs to continue parsing manually though, by calling `\stex_smsmode_do:` as (usually) its last token.

`\stex_sms_allow_env:n` registers the provided environment to be allowed in sms mode.

As with `\stex_sms_allow_escape:N`, the `\begin{`⟨*envname*⟩`}` is escaped, hence the begin-code of the environment needs to call `\stex_smsmode_do:`. Since `\end{`⟨*envname*⟩`}` never takes arguments, it does not need to be escaped.

`\stex_sms_allow_import:Nn` `\stex_sms_allow_import:Nn \macro {`⟨*code*⟩`}`

registers the provided macro to be allowed in the *first pass* in sms mode. The provided ⟨*code*⟩ is executed at the start of the first pass, to define macros accordingly.

`\stex_sms_allow_import_env:nn` `\stex_sms_allow_import_env:nn \envname {`⟨*code*⟩`}`

registers the provided environment to be allowed in the *first pass* in sms mode. The provided ⟨*code*⟩ is executed at the start of the first pass, to define macros accordingly.

`\g_stex_sms_import_code` inserted between the first and second pass; macros/environments allowed via `\stex_sms_allow_import_*` should store their "results" in here.

`\stex_if_smsmode_p:` ⋆
`\stex_if_smsmode:`*TF* ⋆
tests for whether we are currently in sms-mode.

`\stex_file_in_smsmode:Nn` `\stex_file_in_smsmode:Nn \document-URI {`⟨*file string*⟩`}`

Loads `{`⟨*file string*⟩`}` in sms mode, setting `\l_stex_current_document_uri` etc. accordingly and reverting them afterwards.

`\stex_smsmode_do:` should be called in escaped macros or environments allowed in sms mode; switches back to parsing file contents ignoring everything not explicitly allowed in sms mode.

Does nothing outside of sms mode, so can be called safely.

## 16.2 Inheritance and Morphisms

`\stex_require_module:N`
`\stex_require_module_noerr:N`
`\stex_require_module_noerr:n`

makes sure that the module with the given URI is in scope. If not, it will attempt to load the module from disk. `\stex_require_module:N` throws an error if the module can't be found; `\stex_require_module_noerr:N` silently fails.

The `\*:n`-variant takes the URI tuple directly rather than a macro.

`\usemodule` `\usemodule [⟨archive ID⟩] {⟨module⟩}`

loads the specified module without exporting it further.

`\importmodule` `\importmodule [⟨archive ID⟩] {⟨module⟩}`

loads the specified module and exports it further.

`\requiremodule` `\requiremodule [⟨archive ID⟩] {⟨module⟩}`

loads the specified module without exporting it further, but generates FTML equivalent to `\importmodule`.

`\stex_add_morphism:nnnn` `\stex_add_morphism:nnnn {⟨name⟩} {⟨module URI⟩} {⟨kind⟩} {⟨contents⟩}`

adds a morphism to the current module

## 16.3 Theory Morphisms

`\stex_structural_feature_morphism:nnnnn`
`\stex_structural_feature_morphism_with_macros:nnnnn`
`\stex_structural_feature_morphism_end:`

`#1` : name
`#2` : kind
archive ID
`#3` : module spec
`#4` : additional attributes

`\stex_iterate_morphisms:nn`

`\stex_get_in_morphism:n`

`copymodule` (*env.*) (and `\copymod`)

`interpretmodule` (*env.*) (and `\interpretmod`)

\assign

\renamedecl

\assignMorphism

# Chapter 17

# Symbols

## 17.1 Declarations

\symdecl `\symdecl[*] {⟨iname⟩} [⟨options⟩]`

takes care of the optional arguments, styling, generates the symbol, defines the relevant macros and adds them to the current module.

\symdef `\symdef {⟨iname⟩} [⟨options⟩] {⟨notation⟩}`

\textsymdecl

\stex_symdecl_do: does the actual work. Requires all the `\l_stex_key_*` macros are set.

_stex_symdecl_html: not namespaced so it can be reused in e.g. `sdefinitions` that generate new symbols. Requires that the relevant `\l_stex_key_*` macros and `\l_stex_macroname_str` are set.

\stex_add_symbol:nnnnnnN
`#1` : {⟨*Macro name*⟩}
`#2` : {⟨*Name*⟩}
`#3` : {⟨*arity*⟩}
`#4` : {({⟨*Arg num*⟩}{⟨*Arg str*⟩})*}
`#5` : Definiens
`#6` : type
`#7` : Return
`#8` : Command
adds a new symbol to the current module.

\stex_has_definiens_p:N ⋆
\stex_has_definiens:NTF ⋆
\stex_has_definiens_p:n ⋆
\stex_has_definiens:nTF ⋆

## 17.2 Retrieval

`\stex_iterate_symbols:n`
`\stex_iterate_break:`
`\stex_iterate_break:n`

iterates over all symbols currently in scope. The provided code will receive nine arguments: The URI of the containing module and the eight parameters as in `\stex_add_symbol:nnnnnnnN`.

iteration is stopped in the code using `\stex_iterate_break:` or `\stex_iterate_break:n`.

`\stex_iterate_symbols:nn` `\stex_iterate_symbols:nn {⟨modules⟩} {⟨code⟩}`

iterates over all symbols in the comma-separated list of module URIs in ⟨*modules*⟩. The provided code will receive nine arguments: The URI of the containing module and the eight parameters as in `\stex_add_symbol:nnnnnnnN`.

iteration is stopped in the code using `\stex_iterate_break:` or `\stex_iterate_break:n`.

`\stex_get_symbol:n`
`\stex_get_symbol:nTF`
`\stex_get_symbol:nF`

attempts fo find the symbol with the given name/id/URI suffix. If not successful, will execute the `F` code or throw an error. Otherwise, defines the following macros:

- `\l_stex_get_symbol_uri`,
- `\l_stex_get_symbol_arity_int`,
- `\l_stex_get_symbol_args_tl`,
- `\l_stex_get_symbol_def_tl`,
- `\l_stex_get_symbol_type_tl`,
- `\l_stex_get_symbol_return_tl`,
- `\l_stex_get_symbol_invoke_cs`

## 17.3 Variables

`\l_stex_variables_prop` contains all variables currently in scope.

`\vardef`

`\newvar`

`\newseq`

\stex_get_var:n

### 17.3.1 Variable Sequences

\varseq

## 17.4 Expressions

\symuse retrieves a symbol by name/id and invokes it as if using a semantic macro.

_stex_next_symbol:n code to be executed the next time a symbol is invoked.

_stex_invoke_symbol:NnnnnnN
_stex_invoke_symbol:NeooooN
_stex_invoke_symbol:nnnnnnN

invokes the symbol. Takes as arguments the following data, and defines the corresponding macros accordingly:

- \l_stex_current_symbol_uri,
- \l_stex_current_symbol_arity_int,
- \l_stex_current_symbol_args_tl,
- definiens (currently not used),
- \l_stex_current_symbol_type_tl,
- \l_stex_current_symbol_return_tl,
- the invokation macro (is called after setup).

Also defines \l_stex_current_full_tl and \l_stex_current_display_tl

For a "normal" symbol defined via \symdecl or \symdef, \l_stex_current_symbol_invoke_cs is defined as \stex_invoke_symbol:.

Opens a new TEX group that needs to be closed by \l_stex_current_symbol_invoke_cs!

\stex_invoke_symbol: Invokes \l_stex_current_symbol_uri; assumes all the \l_stex_current_*-macros have been defined prior.

\stex_invoke_text_symbol: Invokes \l_stex_current_symbol_uri as a symbol declared via \textsymdecl; assumes all the \l_stex_current_*-macros have been defined prior.

`\stex_invoke_sequence:` Invokes `\l_stex_current_symbol_uri` as a sequence variable declared via `\varseq`; assumes all the `\l_stex_current_*`-macros have been defined prior.

`\stex_invoke_structure:` Invokes a mathstructure symbol; assumes all the `\l_stex_current_*`-macros have been defined prior.

`\_stex_invoke_variable:nnnnnn`

invokes the variable. Takes as arguments the following data, and defines the corresponding macros accordingly:

- variable name,
- `\l_stex_current_symbol_arity_int`,
- `\l_stex_current_symbol_args_tl`,
- definiens (currently not used),
- `\l_stex_current_symbol_type_tl`,
- `\l_stex_current_symbol_return_tl`,
- the invokation macro (is called after setup).

`\svar`

### 17.4.1 Term Annotations

`\_stex_eat_exclamation_point:`

removes spurious `!` characters

`\_stex_term_oms:nn` `\_stex_term_oms:nn {⟨notation id⟩} {⟨code⟩}`

annotes ⟨*code*⟩ with `OMID="\l_stex_current_symbol_uri"`

`\_stex_term_omv:nn` `\_stex_term_omv:nn {⟨notation id⟩} {⟨code⟩}`

annotes ⟨*code*⟩ with `OMV="\l_stex_current_symbol_uri"`

TODO: This shouldn't use `\l_stex_current_symbol_uri`!

`\_stex_term_oma:nn` `\_stex_term_oma:nn {⟨notation id⟩} {⟨code⟩}`

annotes ⟨*code*⟩ with `OMA="\l_stex_current_symbol_uri"`

`\_stex_term_omb:nn` `\_stex_term_omb:nn {⟨notation id⟩} {⟨code⟩}`

annotes ⟨*code*⟩ with `OMBIND="\l_stex_current_symbol_uri"`

### 17.4.2 Symbol Arguments

`\_stex_term_arg:nnnnn` marks an argument of a symbol application, sets the precedence accordingly, sets `\l_stex_allow_semantic_bool` etc.
`#1` : argument number
`#2` : argument mode
`#3` : precedence
`#4` : argument name (WiP)
`#5` : code / body

`\_stex_term_arg:nnn` `\_stex_term_arg:nnn {⟨mode⟩} {⟨num⟩} {⟨code⟩}`
inserts the HTML annotations for the argument

`\_stex_term_arg_aB:nnnnn` takes care of sequence arguments

`\seqmap`

### 17.4.3 Notation components

`\comp`
`\compemph@uri`
`\compemph`

`\varemph@uri`
`\varemph`

`\defemph@uri`
`\defemph`

`\symrefemph@uri`
`\symrefemph`

The following commands do the actual attributes:

`\_do_comp:nnNn` `\_do_comp:nNn {⟨key-suffix⟩} {⟨html-value⟩} \comp-macro {⟨code⟩}`
annotated ⟨*code*⟩ with `data-ftml-`⟨*key-suffix*⟩ using the highlighting dictated by `\comp-macro`.

### 17.4.4 Symbol References

\symref
\sr

\symname
\sn
\sns
\Symname
\Sn\Sns

\varref
\varname
\Varname

\definiendum
\definame
\Definame
\defnotation

\foldexpr

## 17.5 Checking

\stex_if_check_terms_p: ⋆
\stex_if_check_terms:*TF* ⋆

conditional whether terms (types and definientia of symbols) should get syntactically checked. In HTML mode, this is always false, since they get written to the HTML anyway.

\stex_check_term:nn

typesets the given expression in a `\tiny\marginpar`, checking its syntactic validity. Does nothing, if term checking is not explicitly activated.

_stex_check_terms:

checks the type and definiens components of a symbol declaration (i.e. `\stex_key_type_tl`, `\stex_key_def_tl` and `\stex_key_return_tl!`).

# Chapter 18

# Notations

`\notation`

`\varnotation`

`\stex_notation_parse:n` parses the body of a notations and saves it in `\l_stex_notation_macrocode_cs`

`\_stex_notation_add:` adds the fully parsed notation to the current module

`\stex_add_notation:nnnnn`
`\stex_add_notation:ooeoo`
`#1` : URI
`#2` : variant
`#3` : arity
`#4` : macro body
`#5` : op
adds the notation to the current module

`\_stex_map_args:N`
`\_stex_map_notation_args:N`

`\_stex_var_notation_macro:`

`\setnotation`
`\_stex_notation_set_default:n`

`\stex_iterate_notations:nn` `\stex_iterate_notations:nn {⟨modules⟩} {⟨code⟩}`

iterates over all notations in the comma-separated list of module URIs and executes `{⟨code⟩}` with `#1`,`#2`,`#3`,`#4`,`#5` as the notation parameters.

`\stex_use_notation:nnTF` `\stex_use_notation:nn {⟨uri⟩} {⟨id⟩} {⟨exists code⟩} {⟨missing code⟩}`
`\stex_use_op_notation:nnTF`

sets `\l_stex_notation_cs`

`\_stex_notation_check:`

`\_stex_notation_do_html:n`

`\_stex_notation_make_args:`

`\stex_do_default_notation:`
`\stex_do_default_notation_op:`

`\STEXInternalNotation`

`#1` : notation id
`#2` : operator precedence
`#3` : intent (WiP)
`#4` : arguments
`#5` : notation code
`#6` : continuation

`\argsep`

`\argmap`

`\argarraymap`

## 18.1 Automated Bracketing

`\infprec`
`\neginfprec`

infinite and negative infinite precedences.

`\_stex_maybe_brackets:nn` `\_stex_maybe_brackets:nn {⟨prec⟩} {⟨body⟩}`

inserts parentheses around ⟨`body`⟩ iff precedences and context call for it.

`\dobrackets` actually inserts parentheses around its argument.

`\withbrackets` `\withbrackets {⟨left⟩} {⟨right⟩} {⟨body⟩}`
sets ⟨*left*⟩ and ⟨*right*⟩ as the parentheses to use in ⟨*body*⟩.

`\dowithbrackets` `\dowithbrackets {⟨left⟩} {⟨right⟩} {⟨body⟩}`
combines `\withbrackets` and `\dobrackets`.

# Chapter 19

# Structures

`mathstructure` (*env.*)

`extstructure` (*env.*)

`\this`

`\stex_get_mathstructure:n` sets `\l_stex_get_structure_module_uri` or throws an error if no structure by the given name/id exists.

`\usestructure`

# Chapter 20

# Statements

`\_stex_do_for_list:` resolves each id in the `for={...}` comma-separated list of ids `\l_stex_key_for_clist`. Stores the full resolved URIs in `\l_stex_fors_seq`.

`\stex_new_statement:nnn` `\stex_new_statement:nnn {⟨env name⟩} {⟨macro name⟩} {⟨code⟩}`

defines a new statement environment `s`⟨*env name*⟩ and the optional macro-variant `\inline`⟨*macro name*⟩ (may be empty). {⟨*code*⟩} does arbitrary additional setup and is called in the `\begin` part of the environment and the macro after optional arguments have been parsed. e.g. `\stex_new_statement:nnn{definition}{def}{...}` defines the `sdefinition` environment and the `\inlinedef` macro.

`sdefinition` (*env.*) (and `\inlinedef`)

`sassertion` (*env.*) (and `\inlineass`)

`sexample` (*env.*) (and `\inlineex`)

`sparagraph` (*env.*)

`definiens`

`\_stex_add_definiens:nn` marks `#2` as the definiens of the symbol with uri `#1`. Also marks the symbol as being defined, iff the symbol is declared in the current module.

`\varbind`

`\conclusion`

`\premise`

# Chapter 21

# Proofs

sproof (*env.*)

subproof (*env.*)

spfsketchenv (*env.*)

\spfsketch

spfstepenv (*env.*)

spfblock (*env.*)

\spfstep

\assumption

\conclude

\eqstep

\yield

\spfjust

\spfby

\spfarg

\sproofend

\stexcommentfont

# Chapter 22

# Metatheory

TODO

# Chapter 23

# Others

# Chapter 24

# Additional Packages

## 24.1 NotesSlides Documentation

TODO

## 24.2 Problem Documentation

TODO

## 24.3 HWExam Documentation

TODO

## 24.4 Tikzinput Documentation

TODO

# Part IV
# Implementation

# Chapter 25

# Setting up

Setup code for the document class

```
⟨*cls⟩
%%%%%%%%%%%%   stex.dtx   %%%%%%%%%%%%

\RequirePackage{expl3,l3keys2e}
\ProvidesExplClass{stex}{2026/06/24}{4.1.0}{sTeX document class}
\IfFileExists{stex-expl-compat.sty}{
  \usepackage{stex-expl-compat}
}{}

\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
\ProcessOptions

\RequirePackage{stex}
\str_set:Nn \g_stex_document_kind_str{fragment}

\LoadClass{article}
⟨/cls⟩
```

Setup code for the package

```
⟨*package⟩
\RequirePackage{expl3,l3keys2e,ltxcmds}
\ProvidesExplPackage{stex}{2026/06/24}{4.1.0}{sTeX package}
\IfFileExists{stex-expl-compat.sty}{
  \usepackage{stex-expl-compat}
}{}
\RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
\RequirePackage{standalone}

\bool_new:N \c_stex_use_sref_bool
\message{^^J*~This~is~sTeX~version~4.1.0~*^^J}
```

Package options:

```
\keys_define:nn { stex / package } {
  debug      .str_set_x:N  = \c_stex_debug_clist ,
  lang       .clist_set:N  = \c_stex_languages_clist ,
  mathhub    .tl_set_x:N   = \mathhub ,
  usesms     .bool_set:N   = \c_stex_persist_mode_bool ,
```

```
  writesms   .bool_set:N   = \c_stex_persist_write_mode_bool ,
  forcenousesms .bool_set:N   = \c_stex_persist_force_bool ,
  checkterms .bool_set:N   = \c_stex_check_terms_bool ,
  metadata   .bool_set:N   = \c_stex_metadata_bool ,
  image      .bool_set:N   = \c_tikzinput_image_bool,
  nofrontmatter .bool_set:N = \c_stex_no_frontmatter_bool,
  unknown    .code:n       = {}
}
\exp_args:NNo \clist_set:Nn \c_stex_debug_clist \c_stex_debug_clist
\ProcessKeysOptions { stex / package }
```

Error messages:

```
\input{stex-en.ldf}
```

# Chapter 26

# Utilities

\stex_ignore_spaces_and_pars:

```
\cs_new_protected:Nn \stex_ignore_spaces_and_pars:{
  \begingroup\catcode13=10\relax
  \@ifnextchar\par{
    \endgroup\expandafter\stex_ignore_spaces_and_pars:\@gobble
  }{
    \endgroup
  }
}
```

(*End of definition for* `\stex_ignore_spaces_and_pars:`. *This function is documented on page 123.*)

sfunction

`\stex_undefine:N`
`\stex_undefine:c`

```
\cs_new_protected:Nn \stex_undefine:N {
  \cs_set_eq:NN #1 \tex_undefined:D
}
\cs_generate_variant:Nn \stex_undefine:N {c}
```

(*End of definition for* `\stex_undefine:N`. *This function is documented on page 123.*)

sfunction

\stex_str_if_starts_with_p:nn
`\stex_str_if_starts_with:nn`*TF*

```
\prg_new_conditional:Nnn \stex_str_if_starts_with:nn {p,T,F,TF} {
  \exp_args:Ne \str_if_eq:nnTF {
    \str_range:nnn{#1}{1}{\str_count:n{#2}}
  }{#2}\prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_str_if_starts_with:nnTF`. *This function is documented on page 123.*)

sfunction

`\stex_str_if_ends_with_p:nn`
`\stex_str_if_ends_with:nn`*TF*

```
\prg_new_conditional:Nnn \stex_str_if_ends_with:nn {p,T,F,TF} {
  \exp_args:Ne \str_if_eq:nnTF {
    \str_range:nnn{#1}{- \str_count:n{#2}}{-1}
  }{#2}\prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_str_if_ends_with:nnTF`. *This function is documented on page 123.*)

sfunction

`\stex_deactivate_macro:Nn`
`\stex_reactivate_macro:N`

```
\cs_new_protected:Nn \stex_deactivate_macro:Nn {
  \tl_set_eq:cN{\tl_to_str:n{#1}~~~orig}#1
  \cs_set_protected:Npn#1{
    \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
  }
}
\cs_new_protected:Nn \stex_reactivate_macro:N {
  \cs_set_eq:Nc #1{\tl_to_str:n{#1}~~~orig}
}
```

(*End of definition for* `\stex_deactivate_macro:Nn` *and* `\stex_reactivate_macro:N`. *These functions are documented on page 123.*)

sfunction

## 26.1 `kpsewhich` and Environment Variables

⟨@@=stex_kpse⟩

`\stex_kpsewhich:Nn`

```
\cs_new_protected:Nn \stex_kpsewhich:Nn {
  \group_begin:
    \catcode`\ =12
    \sys_get_shell:nnN { kpsewhich ~ #2 } { } \l_tmpa_tl
    \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
  \group_end:
  \exp_args:NNo\str_set:Nn #1 \l_tmpa_tl
  \tl_trim_spaces:N #1
}
```

(*End of definition for* `\stex_kpsewhich:Nn`. *This function is documented on page 123.*)

sfunction

`\stex_get_env:Nn`

```
\sys_if_platform_windows:TF{
  \cs_new_protected:Nn \stex_get_env:Nn {\group_begin:
    \escapechar=-1\catcode`\\=12
    \exp_args:NNe \stex_kpsewhich:Nn #1 {-expand-var~\c_percent_str#2\c_percent_str}
    \exp_args:NNx \str_if_eq:VnT #1 {\c_percent_str #2 \c_percent_str}{
      \str_clear:N #1
    }
    \exp_args:NNx\use:nn\group_end:{
      \str_set:Nn \exp_not:N #1 { #1 }
    }
  }
}{
  \cs_new_protected:Nn \stex_get_env:Nn {
    \stex_kpsewhich:Nn #1 {-var-value~#2}
  }
}
```

(*End of definition for* `\stex_get_env:Nn`. *This function is documented on page 124.*)

sfunction

## 26.2 Logging

102 ⟨@@=stex_debug⟩

`\stex_debug:nn`

```
103 \cs_new_protected:Nn \stex_debug:nn {
104   \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist { \tl_to_str:n{all} }{
105     \__stex_debug_:nn{#1}{#2}
106   }{
107     \exp_args:NNo \clist_if_in:NnT \c_stex_debug_clist { \tl_to_str:n{#1} }{
108       \__stex_debug_:nn{#1}{#2}
109     }
110   }
111 }
112
113 \cs_new_protected:Nn \__stex_debug_:nn {
114   \msg_set:nnn{stex}{debug / #1}{
115     \\Debug~#1:~#2\\
116   }
117   \msg_none:nn{stex}{debug / #1}
118 }
```

We check an environment variable for debugging and set things up:

```
119 \stex_get_env:Nn\__stex_debug_env_str{STEX_DEBUG}
120 \str_if_empty:NTF\__stex_debug_env_str {
121   \clist_set_eq:NN \l__stex_debug_cl \c_stex_debug_clist
122 }{
123   \clist_set:No \l__stex_debug_cl {\__stex_debug_env_str}
124 }
125 \clist_clear:N \c_stex_debug_clist
126 \clist_map_inline:Nn \l__stex_debug_cl {
127   \exp_args:NNo \clist_put_right:Nn \c_stex_debug_clist
128   { \tl_to_str:n{#1} }
129 }
130
131 \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist {\tl_to_str:n{all}} {
132     \msg_redirect_module:nnn{ stex }{ none }{ warning }
133     \stex_debug:nn{all}{Logging~everything!}
134 }{
135   \clist_map_inline:Nn \c_stex_debug_clist {
136     \msg_redirect_name:nnn{ stex }{ debug / #1 }{ warning }
137     \stex_debug:nn{#1}{Logging~#1}
138   }
139 }
```

(*End of definition for* `\stex_debug:nn`. *This function is documented on page 124.*)

sfunction

## 26.3 File Paths

```
⟨@@=stex_path⟩
```

\stex_file_set:Nn
\stex_file_set:No
\stex_file_set:Ne

```
\cs_new_protected:Nn \stex_file_set:Nn {
  \str_if_empty:nTF {#2} { \seq_clear:N #1 }{
    \exp_args:NNno \seq_set_split:Nnn #1 / { \tl_to_str:n{#2} }
  }
}
\cs_generate_variant:Nn \stex_file_set:Nn {No, Ne}
```

(*End of definition for* `\stex_file_set:Nn`. *This function is documented on page 124.*)

sfunction

\stex_file_resolve:Nn
\stex_file_resolve:No
\stex_file_resolve:Ne

```
\sys_if_platform_windows:TF{
  \cs_new_protected:Npn \__stex_path_win_take:w #1#2#3 \__stex_path_: {
    \uppercase{ \str_set:Nn \l__stex_path_str{#1}}
    \str_set:Ne \l__stex_path_win_drive {\l__stex_path_str #2}
    \str_set:Nn \l__stex_path_str{#3}
  }
  \cs_new_protected:Nn \stex_file_resolve:Nn {
    \str_set:Nn \l__stex_path_str {#2}
    \str_clear:N \l__stex_path_win_drive
    \exp_args:NNo \str_replace_all:Nnn \l__stex_path_str \c_backslash_str /
    \exp_args:Ne \str_if_eq:nnT {\str_item:Nn \l__stex_path_str 2} : {
      \exp_after:wN \__stex_path_win_take:w \l__stex_path_str \__stex_path_:
    }
    \stex_file_set:No #1 \l__stex_path_str
    \__stex_path_canonicalize:N #1
    \str_if_empty:NF \l__stex_path_win_drive {
      \seq_pop_left:NN #1 \l__stex_path_str
      \seq_put_left:No #1 \l__stex_path_win_drive
    }
    %\stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
  }
}{
  \cs_new_protected:Nn \stex_file_resolve:Nn {
    \str_set:Nn \l__stex_path_str {#2}
    \stex_file_set:No #1 \l__stex_path_str
    \__stex_path_canonicalize:N #1
    % \stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
  }
}
\cs_generate_variant:Nn \stex_file_resolve:Nn {No, Ne}
```

Auxiliary methods:

```
\cs_new_protected:Nn \__stex_path_canonicalize:N {
  \seq_if_empty:NF #1 {
    \seq_pop:NN #1 \l__stex_path_can_str
    \seq_clear:N \l__stex_path_can_seq
    \str_if_empty:NTF \l__stex_path_can_str {
      \seq_map_function:NN #1 \__stex_path_dodots:n
      \seq_put_left:Nn \l__stex_path_can_seq {}
```

```
    }{
      \seq_push:No #1 \l__stex_path_can_str
      \seq_map_function:NN #1 \__stex_path_dodots:n
    }
    \seq_set_eq:NN #1 \l__stex_path_can_seq
  }
}

\cs_new_protected:Nn \__stex_path_dodots:n {
  \str_if_empty:nF{#1}{
    \str_if_eq:nnF {#1} {.} {
      \str_if_eq:nnTF {#1} {..} {
        \seq_if_empty:NF \l__stex_path_can_seq {
          \seq_pop_right:NN \l__stex_path_can_seq \l__stex_path_can_str
        }
      }{
        \seq_put_right:Nn \l__stex_path_can_seq {#1}
      }
    }
  }
}
```

(*End of definition for* `\stex_file_resolve:Nn`. *This function is documented on page 124.*)

sfunction

`\stex_file_use:N`

```
\cs_new:Nn \stex_file_use:N {
  \seq_use:Nn #1 /
}
```

(*End of definition for* `\stex_file_use:N`. *This function is documented on page 124.*)

sfunction

`\stex_file_split_off_ext:NN`

```
\cs_new_protected:Nn \stex_file_split_off_ext:NN {
  \seq_set_eq:NN #1 #2
  \seq_pop_right:NN #1 \l__stex_path_str
  \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
  \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
  \seq_put_right:Ne #1 {\seq_use:Nn \l__stex_path_seq .}
}
```

(*End of definition for* `\stex_file_split_off_ext:NN`. *This function is documented on page 124.*)

sfunction

`\stex_file_split_off_lang:NN`

```
\cs_new_protected:Nn \stex_file_split_off_lang:NN {
  \seq_set_eq:NN #1 #2
  \seq_pop_right:NN #1 \l__stex_path_str
  \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
  \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str

  \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
  \exp_args:NNo \prop_if_in:NnF \c_stex_languages_prop \l__stex_path_str {
```

```
    \seq_put_right:No \l__stex_path_seq \l__stex_path_str
  }

  \seq_put_right:Ne #1 {\seq_use:Nn \l__stex_path_seq .}
}
```

(*End of definition for* `\stex_file_split_off_lang:NN`. *This function is documented on page 124.*)

sfunction

`\c_stex_pwd_file` `\c_stex_main_file`

We determine the pwd

```
\sys_if_platform_windows:TF{
  \stex_get_env:Nn\l__stex_path_str{CD}
}{
  \stex_get_env:Nn\l__stex_path_str{PWD}
}
\stex_file_resolve:No \c_stex_pwd_file \l__stex_path_str
\seq_set_eq:NN \c_stex_main_file \c_stex_pwd_file
\seq_put_right:Ne \c_stex_main_file {\jobname\tl_to_str:n{.tex}}

\stex_debug:nn {files} {PWD:~\stex_file_use:N \c_stex_pwd_file}
```

(*End of definition for* `\c_stex_pwd_file` *and* `\c_stex_main_file`. *These variables are documented on page 124.*)

`\c_stex_home_file`

```
\sys_if_platform_windows:TF{
  \stex_get_env:Nn \l__stex_path_str {homedrive\c_percent_str\c_percent_str homepath}
}{
  \stex_get_env:Nn \l__stex_path_str {HOME}
}
\stex_file_resolve:No \c_stex_home_file \l__stex_path_str
```

(*End of definition for* `\c_stex_home_file`. *This variable is documented on page 125.*)

`\g_stex_current_file`

```
\seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
```

(*End of definition for* `\g_stex_current_file`. *This variable is documented on page 125.*)

`\stex_input_with_hooks:Nn` `\stex_input_with_hooks:Ne`

```
\cs_new_protected:Nn \stex_input_with_hooks:Nn {
  \stex_with_file_hooks:Nnn #1 {#2} {\input{#2}}
}
\cs_generate_variant:Nn \stex_input_with_hooks:Nn {Ne}
```

(*End of definition for* `\stex_input_with_hooks:Nn`. *This function is documented on page 125.*)

sfunction

`\stex_with_file_hooks:Nnn`

```
\cs_new_protected:Nn \stex_with_file_hooks:Nnn {
  \stex_pseudogroup:nn {
    \stex_file_resolve:Nn \l__stex_path_seq {#2}
    \seq_gset_eq:NN \g_stex_current_file \l__stex_path_seq
    \tl_set_eq:NN \l_stex_current_document_uri #1
    \str_set:Ne \l_stex_current_language_str { \stex_document_uri_language:N \l_stex_curren
```

```
    #3
  }{
    \tl_gset:Nn \exp_not:N \g_stex_current_file { \exp_args:No \exp_not:n \g_stex_current_f
    \stex_pseudogroup_restore:N \l_stex_current_language_str
    \stex_pseudogroup_restore:N \l_stex_current_document_uri
  }
}
```

(*End of definition for* `\stex_with_file_hooks:Nnn`. *This function is documented on page 125.*)

sfunction

`\stex_if_file_absolute_p:N`
`\stex_if_file_absolute:NTF`

```
\sys_if_platform_windows:TF {
  \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
    \seq_if_empty:NTF #1 \prg_return_false: {
      \exp_args:Ne \tl_if_empty:nTF {\seq_item:Nn #1 1} \prg_return_true: {
        \exp_args:Ne \str_if_eq:nnTF { \exp_args:Ne \str_item:nn {\seq_item:Nn #1 1} 2} :
          \prg_return_true: \prg_return_false:
      }
    }
  }
}{
  \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
    \seq_if_empty:NTF #1 \prg_return_false: {
      \exp_args:Ne \tl_if_empty:nTF {\seq_item:Nn #1 1}
        \prg_return_true: \prg_return_false:
    }
  }
}
```

(*End of definition for* `\stex_if_file_absolute:NTF`. *This function is documented on page 125.*)

sfunction

`stex_if_file_starts_with:NNTF`

```
\prg_new_protected_conditional:Nnn \stex_if_file_starts_with:NN {T,F,TF} {
  \seq_set_eq:NN \l__stex_path_a_seq #1
  \seq_set_eq:NN \l__stex_path_b_seq #2
  \tl_clear:N \l__stex_path_return_tl
  \bool_while_do:nn{
    \bool_not_p:n{
      \bool_lazy_any_p:n{
        {\seq_if_empty_p:N \l__stex_path_a_seq}
        {\seq_if_empty_p:N \l__stex_path_b_seq}
        {\bool_not_p:n{\tl_if_empty_p:N \l__stex_path_return_tl}}
      }
    }
  }{
    \seq_pop_left:NN \l__stex_path_a_seq \l__stex_path_a_tl
    \seq_pop_left:NN \l__stex_path_b_seq \l__stex_path_b_tl
    \str_if_eq:NNF \l__stex_path_a_tl \l__stex_path_b_tl {
      \tl_set:Nn \l__stex_path_return_tl {\prg_return_false:}
    }
  }
  \tl_if_empty:NTF \l__stex_path_return_tl {
```

```
    \seq_if_empty:NTF \l__stex_path_b_seq \prg_return_true: \prg_return_false:
  } \l__stex_path_return_tl
}
```

(*End of definition for* `\stex_if_file_starts_with:NNTF`. *This function is documented on page 125.*)

sfunction

## 26.4 Group-like Behaviours

```
⟨@@=stex_groups⟩
```

`\stex_pseudogroup:nn`

```
\cs_new_protected:Npn \stex_pseudogroup:nn {
  \exp_args:Nne \use:nn
}
```

(*End of definition for* `\stex_pseudogroup:nn`. *This function is documented on page 125.*)

sfunction

`\stex_pseudogroup_restore:N`

```
\cs_new:Nn \stex_pseudogroup_restore:N {
  \tl_if_exist:NTF #1 {
    \tl_set:Nn \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
  }{
    \stex_undefine:N \exp_not:N #1
  }
}
```

(*End of definition for* `\stex_pseudogroup_restore:N`. *This function is documented on page 126.*)

sfunction

`\stex_pseudogroup_with:nn`

```
\cs_new_protected:Nn \stex_pseudogroup_with:nn {
  \tl_map_inline:nn{#1}{
    \cs_set_eq:cN{__stex_groups_\tl_to_str:n{##1}}##1
  }
  #2
  \tl_map_inline:nn{#1}{
    \cs_set_eq:Nc##1{__stex_groups_\tl_to_str:n{##1}}
    \stex_undefine:c{__stex_groups_\tl_to_str:n{##1}}
  }
}
```

(*End of definition for* `\stex_pseudogroup_with:nn`. *This function is documented on page 126.*)

sfunction

`\stex_metagroup_new:` Current metagroup group level

```
\int_new:N \l__stex_groups_lvl_int
```

start a new metagroup at the current group level

```
\cs_new_protected:Nn \stex_metagroup_new: {
  \int_set_eq:NN \l__stex_groups_lvl_int \currentgrouplevel
}
```

(*End of definition for* `\stex_metagroup_new:`. *This function is documented on page 126.*)

sfunction

`\stex_metagroup_do_in:n`
`\stex_metagroup_do_in:e`

```
\cs_new_protected:Npn \stex_metagroup_do_in:n {
  \int_compare:nNnTF \l__stex_groups_lvl_int = \currentgrouplevel
    \use:n \__stex_groups_do_in:n
}
\cs_generate_variant:Nn \stex_metagroup_do_in:n {e}

\cs_new_protected:Nn \__stex_groups_do_in:n {
  \tl_if_exist:cTF{g__stex_groups_\the\currentgrouplevel _tl}{
    \exp_args:Nno \tl_gput_right:cn{g__stex_groups_\the\currentgrouplevel _tl}
  }{
    \exp_args:Nno \tl_gset:cn{g__stex_groups_\the\currentgrouplevel _tl}
  }{ #1 }
  \bool_if_exist:cF {l__stex_groups_\the\currentgrouplevel _bool} {
      \group_insert_after:N \__stex_groups_do:
      \bool_set_true:c {l__stex_groups_\the\currentgrouplevel _bool}
  }
  #1
}

\cs_new_protected:Nn \__stex_groups_do: {
  \tl_if_exist:cT{g__stex_groups_\int_eval:n{\currentgrouplevel+1}_tl}{
    \exp_args:NNo \exp_args:No \stex_metagroup_do_in:n {
      \csname g__stex_groups_\int_eval:n{\currentgrouplevel+1}_tl \endcsname
    }
    \cs_undefine:c{g__stex_groups_\int_eval:n{\currentgrouplevel+1}_tl}
  }
  \bool_if_exist:cF {l__stex_groups_\the\currentgrouplevel _bool} {
    \group_insert_after:N \__stex_groups_do:
    \bool_set_true:c {l__stex_groups_\the\currentgrouplevel _bool}
  }
}
```

(*End of definition for* `\stex_metagroup_do_in:n`. *This function is documented on page 126.*)

sfunction

## 26.5 Key Handling

⟨`@@=stex_keys`⟩

`\stex_keys_define:nnnn`

```
\cs_new_nopar:Nn \stex_keys_define:nnnn {
  \tl_gset:cn {__stex_keys_keys_#1_pre_tl}{#2}
  \tl_gset:cn {__stex_keys_keys_#1_def_tl}{#3}
  \tl_if_empty:nF{#4}{
    \clist_map_inline:nn{#4}{
      \tl_set_eq:Nc \l__stex_keys_tl {__stex_keys_keys_##1_pre_tl}
      \tl_gput_left:co{__stex_keys_keys_#1_pre_tl} \l__stex_keys_tl
      \tl_set_eq:Nc \l__stex_keys_tl {__stex_keys_keys_##1_def_tl}
      \tl_gput_left:cn{__stex_keys_keys_#1_def_tl} ,
```

```
      \tl_gput_left:co{__stex_keys_keys_#1_def_tl} \l__stex_keys_tl
    }
  }
  \tl_set_eq:Nc \l__stex_keys_tl {__stex_keys_keys_#1_def_tl}
  \exp_args:Nno \keys_define:nn {stex / #1} {\l__stex_keys_tl}
}
```

(*End of definition for* `\stex_keys_define:nnnn`. *This function is documented on page 126.*)

sfunction

`\stex_keys_set:nn`

```
\cs_new_nopar:Nn \stex_keys_set:nn {
  \use:c{__stex_keys_keys_#1_pre_tl}
  \keys_set:nn {stex / #1} { #2 }
}
```

(*End of definition for* `\stex_keys_set:nn`. *This function is documented on page 126.*)

sfunction

Some ubiquitous key sets:

```
\stex_keys_define:nnnn{archive file}{
  \str_clear:N \l_stex_key_archive_str
  \str_clear:N \l_stex_key_file_str
}{
  archive .str_set_x:N = \l_stex_key_archive_str ,
  file    .str_set_x:N = \l_stex_key_file_str
}{}

\stex_keys_define:nnnn{id}{
  \str_clear:N \l_stex_key_id_str
}{
  id .str_set_x:N = \l_stex_key_id_str
}{}

\stex_keys_define:nnnn{title}{
  \tl_clear:N \l_stex_key_title_tl
}{
  title .tl_set:N = \l_stex_key_title_tl
}{}

\stex_keys_define:nnnn{style}{
  \clist_clear:N \l_stex_key_style_clist
}{
  style .clist_set:N = \l_stex_key_style_clist
}{}

\stex_keys_define:nnnn{deprecate}{
  \str_clear:N \l_stex_key_deprecate_str
}{
  deprecate     .str_set_x:N  = \l_stex_key_deprecate_str
}{}

\stex_keys_define:nnnn{uses}{}{
  uses .code:n = {
    \clist_map_inline:nn{#1}{
```

```
      \stex_str_if_starts_with:nnTF{##1}[{
        \__stex_keys_split_at_bracket:w ##1 \_stex_end:
      }{
        \usemodule{##1}
      }
    }
  }
}{}

\cs_new_protected:Npn \__stex_keys_split_at_bracket:w [ #1 ] #2 \_stex_end: {
  \usemodule[#1]{#2}
}
```

## 26.6 Languages

```
⟨@@=stex_lang⟩
```

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

```
\exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
  en = english ,
  de = ngerman ,
  ar = arabic ,
  bg = bulgarian ,
  ru = russian ,
  fi = finnish ,
  ro = romanian ,
  tr = turkish ,
  fr = french ,
  sl = slovenian
}}

\exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
  english   = en ,
  ngerman   = de ,
  arabic    = ar ,
  bulgarian = bg ,
  russian   = ru ,
  finnish   = fi ,
  romanian  = ro ,
  turkish   = tr ,
  french    = fr ,
  slovenian = sl ,
}}
% todo: chinese simplified (zhs)
%       chinese traditional (zht)
```

(*End of definition for* `\c_stex_languages_prop` *and* `\c_stex_language_abbrevs_prop`. *These variables are documented on page 127.*)

\l_stex_current_language_str

```
\str_new:N \l_stex_current_language_str
```

(*End of definition for* `\l_stex_current_language_str`. *This variable is documented on page 127.*)

Loading babel (if necessary), depending on the `lang` package option:

```
\clist_if_empty:NF \c_stex_languages_clist {
  \bool_set_false:N \l__stex_lang_turkish_bool
  \seq_clear:N \l_tmpa_seq
  \clist_map_inline:Nn \c_stex_languages_clist {
    \str_if_empty:NF \l_stex_current_language_str {
      \str_set:Nn \l_stex_current_language_str {#1}
    }
    \str_set:Ne \l_tmpa_str {#1}
    \str_if_eq:nnT {#1}{tr}{
      \bool_set_true:N \l__stex_lang_turkish_bool
    }
    \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
      \tl_set_rescan:Nno \l_tmpa_str {} \l_tmpa_str
      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
    } {
      \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
    }
  }
  \stex_debug:nn{lang} {Languages:~\seq_use:Nn \l_tmpa_seq {,~} }
  \bool_if:NTF \l__stex_lang_turkish_bool {
    \exp_args:NNe \use:nn \RequirePackage
      {[main=\seq_use:Nn \l_tmpa_seq, ,shorthands=:!]}{babel}
  }{
    \exp_args:NNe \use:nn \RequirePackage
      {[main=\seq_use:Nn \l_tmpa_seq, ]}{babel}
  }
}
```

## 26.7 Styling

```
⟨@@=stex_styles⟩
```

`\stex_new_stylable_cmd:nnnn`
`\stex_style_apply:`

```
\cs_new_protected:Nn \stex_new_stylable_cmd:nnnn {
  \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[2][]{
    \__stex_styles_patch:nnn{#1}{##1}{##2}
  }
  \exp_after:wN \NewDocumentCommand\cs:w #1\cs_end:{#2}{
    \cs_set:Npn \stex_style_apply: {
      \__stex_styles_apply_patch:n{#1}
    }
    #3
  }
  \tl_set:cn {__stex_styles_style_#1:} { #4 }
}

\cs_new_protected:Nn \__stex_styles_patch:nnn {
  \str_if_empty:nTF {#2}{
    \tl_set:cn{__stex_styles_style_#1:}{#3}
  }{
    \tl_set:cn{__stex_styles_style_#1_#2:}{#3}
  }
}
```

```
\cs_new_protected:Nn \__stex_styles_apply_patch:n {
  \clist_if_empty:NTF \l_stex_key_style_clist {
    \tl_clear:N \thisstyle
    \use:c{__stex_styles_style_#1:}
  }{
    \clist_get:NN \l_stex_key_style_clist \thisstyle
    \tl_if_exist:cTF{__stex_styles_style_#1_\thisstyle :}{
      \use:c{__stex_styles_style_#1_\thisstyle :}
    }{
      \use:c{__stex_styles_style_#1:}
    }
  }
}
```

(*End of definition for* `\stex_new_stylable_cmd:nnnn` *and* `\stex_style_apply:`*. These functions are documented on page 127.*)

sfunction

`\stex_new_stylable_env:nnnnnnn`

```
\cs_new_protected:Nn \stex_new_stylable_env:nnnnnnn {
  \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[3][]{
    \__stex_styles_patch:nnnn{#1}{##1}{##2}{##3} % <- defined after \stex_if_html_backend
  }
  \exp_after:wN \newcommand \cs:w stexcss#1 \cs_end:[1][]{
    \__stex_styles_css_patch:nnnn{#1}{##1}
  }
  \NewDocumentEnvironment{#7#1}{#2}{
    \cs_set:Npn \stex_style_apply: {
      \_stex_apply_patch_begin:n{#1}
    }
    #3
  }{
    \stex_if_html_backend:F{
      \cs_set:Npn \stex_style_apply: {
        \_stex_apply_patch_end:n{#1}
      }
    }
    #4
  }
  \tl_set:cn {__stex_styles_style_#1_start:} { #5 }
  \tl_set:cn {__stex_styles_style_#1_end:} { #6 }
}


\cs_new_protected:Nn \_stex_apply_patch_begin:n {
  \clist_if_empty:NTF \l_stex_key_style_clist {
    \tl_clear:N \thisstyle
    \use:c{__stex_styles_style_#1_start:}
  }{
    \clist_get:NN \l_stex_key_style_clist \thisstyle
    \stex_debug:nn{styling}{dominant~style:~\thisstyle}
    \tl_if_exist:cTF{__stex_styles_style_#1_\thisstyle _start:}{
      \use:c{__stex_styles_style_#1_\thisstyle _start:}
```

```
    }{
      \use:c{__stex_styles_style_#1_start:}
    }
  }
}

\cs_new_protected:Nn \_stex_apply_patch_end:n {
  \tl_if_empty:NTF \thisstyle {
    \use:c{__stex_styles_style_#1_end:}
  }{
    \tl_if_exist:cTF{__stex_styles_style_#1_\thisstyle _end:}{
      \use:c{__stex_styles_style_#1_\thisstyle _end:}
    }{
      \use:c{__stex_styles_style_#1_end:}
    }
  }
}
```

(*End of definition for* `\stex_new_stylable_env:nnnnnnn`. *This function is documented on page 127.*)

sfunction

## 26.8 Persisting Dependencies

```
⟨@@=stex_persist⟩
```

We check the environment variables:

```
\stex_get_env:Nn\__stex_persist_env_str{STEX_USESMS}
\str_if_empty:NF\__stex_persist_env_str{
  \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
    \bool_set_true:N \c_stex_persist_mode_bool
  }
}
\stex_get_env:Nn\__stex_persist_env_str{STEX_WRITESMS}
\str_if_empty:NF\__stex_persist_env_str{
  \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
    \bool_set_true:N \c_stex_persist_write_mode_bool
  }
}

\bool_if:NT \c_stex_persist_force_bool {
  \bool_set_false:N \c_stex_persist_mode_bool
}

\iow_new:N \c__stex_persist_sms_iow
```

`\stex_persist:n` defined later; requires `\stex_if_html_backend:TF`

`\stex_persist:e`

`\loadsms`

(*End of definition for* `\stex_persist:n` *and* `\loadsms`. *These functions are documented on page 127.*)

sfunction

Is called at the end of the `.sty`-file:

```

\cs_new_protected:Npn \loadsms #1 {
  \stex_str_if_ends_with:nnTF{#1}{.sms}{
    \__stex_persist_load_file:n{#1}
```

```
  }{
    \stex_str_if_ends_with:nnTF{#1}{.tex}{
      \__stex_persist_load_file:n{#1}
    }{
      \__stex_persist_load_file:n{#1.sms}
    }
  }
}

\cs_new_protected:Nn \_stex_persist_read_now: {
  \bool_if:NTF \c_stex_persist_mode_bool {
    \bool_if:NTF \c_stex_persist_write_mode_bool
      \__stex_persist_read_and_write:
      {
        \__stex_persist_load_file:n{\jobname.sms}
      }
  }{
    \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
  }
}

\cs_new_protected:Nn \__stex_persist_load_file:n {
  \file_if_exist:nT{#1}{
    \group_begin:
      \cs_set:Npn \stex_persist:n ##1 {}
      \cs_set:Npn \stex_persist:e ##1 {}
      \stex_debug:nn{persist}{restoring~from~sms~file}
      \catcode`\:=11\relax
      \catcode`\_=11\relax
      \catcode`\ =10\relax
      \cs:w @ @ input \cs_end:#1\relax
    \group_end:
  }
}

\cs_new_protected:Nn \__stex_persist_write_only: {
  \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
  \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
}

\cs_new_protected:Nn \__stex_persist_read_and_write: {
  \file_if_exist:nTF{\jobname.sms}{
    \ior_open:Nn \g_tmpa_ior {\jobname.sms}
    \iow_open:Nn \g_tmpa_iow {\jobname.sms2}
    \ior_str_map_inline:Nn \g_tmpa_ior {
      \iow_now:Nn \g_tmpa_iow {##1}
    }
    \iow_close:N \g_tmpa_iow
    \ior_close:N \g_tmpa_ior
    \__stex_persist_write_only:
    \ior_open:Nn \g_tmpa_ior {\jobname.sms2}
    \ior_str_map_inline:Nn \g_tmpa_ior {
      \iow_now:Nn \c__stex_persist_sms_iow {##1}
    }
```

```
    \ior_close:N \g_tmpa_ior
    \__stex_persist_load_file:n{\jobname.sms2}
  }\__stex_persist_write_only:
}
```

## 26.9 Auxiliary Methods

⟨@@=stex_aux⟩

_stex_do_deprecation:n

```
\cs_new:Nn \_stex_do_deprecation:n {
  \str_if_empty:NF \l_stex_key_deprecate_str {
    \msg_warning:nnxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
  }
}
```

(*End of definition for* _stex_do_deprecation:n. *This function is documented on page* **??**.)

_stex_do_id:

```
\cs_new_protected:Nn \_stex_do_id: {
  \stex_if_smsmode:F {
    \str_if_empty:NF \l_stex_key_id_str {
      \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str
    }
  }
}
```

(*End of definition for* _stex_do_id:. *This function is documented on page* **??**.)

# Chapter 27

# HTML Output

661 ⟨@@=stex_annotate⟩

\stex@backend We determine and \input the backend config file:

```
662 \ifcsname if@rustex\endcsname\else
663   \expandafter\newif\csname if@rustex\endcsname
664   \@rustexfalse
665 \fi
666
667 \stex_get_env:Nn\__stex_annotate_env_str{STEX_FORCE_PDF}
668 \exp_args:No \str_if_eq:nnTF \__stex_annotate_env_str {true} {
669   \def\stex@backend{pdflatex}
670 }{
671   \tl_if_exist:NF\stex@backend{
672     \if@rustex
673       \def\stex@backend{rustex}
674     \else
675       \cs_if_exist:NTF\HCode{
676         \def\stex@backend{tex4ht}
677       }{
678         \def\stex@backend{pdflatex}
679       }
680     \fi
681   }
682 }
683
684 \input{stex-backend-\stex@backend.cfg}
```

(*End of definition for* \stex@backend*. This variable is documented on page 128.*)

\stex_if_html_backend_p:
\stex_if_html_backend:TF is provided by the backend config file.

(*End of definition for* \stex_if_html_backend:TF*. This function is documented on page 128.*)

\ifstexhtml sfunction

```
685 \bool_new:N \l__stex_annotate_do_output_bool
686
687 \newif\ifstexhtml
688 \stex_if_html_backend:TF {
```

```
  \stexhtmltrue
  \bool_set_true:N \l__stex_annotate_do_output_bool
}{
  \stexhtmlfalse
  \bool_set_false:N \l__stex_annotate_do_output_bool
}
```

(*End of definition for* `\ifstexhtml`. *This function is documented on page 128.*)

sfunction

`\stex_if_do_html_p:`
`\stex_if_do_html:TF`

```
\prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
  \bool_if:NTF \l__stex_annotate_do_output_bool
    \prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_if_do_html:TF`. *This function is documented on page 128.*)

sfunction

`\stex_suppress_html:n`

```
\cs_new_protected:Nn \stex_suppress_html:n {
  \stex_pseudogroup:nn{
    \bool_set_false:N \l__stex_annotate_do_output_bool
    #1
  }{
    \stex_if_do_html:T {
      \bool_set_true:N \l__stex_annotate_do_output_bool
    }
  }
}
```

(*End of definition for* `\stex_suppress_html:n`. *This function is documented on page 128.*)

sfunction

`\stex_annotate:nn` is provided by the backend config file.

(*End of definition for* `\stex_annotate:nn`. *This function is documented on page 128.*)

sfunction

`stex_annotate_env` (*env.*) is provided by the backend config file.

`\stex_html_node:nnn` is provided by the backend config file.

(*End of definition for* `\stex_html_node:nnn`. *This function is documented on page 128.*)

sfunction

`stex_env_node` (*env.*) is provided by the backend config file.

`\stex_annotate_invisible:nn` is provided by the backend config file.
`\stex_annotate_invisible:n`

(*End of definition for* `\stex_annotate_invisible:nn` *and* `\stex_annotate_invisible:n`. *These functions are documented on page 129.*)

sfunction

\STEXinvisible

```
\cs_new_protected:Npn \STEXinvisible #1 {
  \stex_annotate_invisible:n { #1 }
}
```

(*End of definition for* `\STEXinvisible`. *This function is documented on page 129.*)

sfunction

\stex_css_link:n is provided by the backend config file.

(*End of definition for* `\stex_css_link:n`. *This function is documented on page 129.*)

sfunction

\stex_css_literal:n is provided by the backend config file.

(*End of definition for* `\stex_css_literal:n`. *This function is documented on page 129.*)

sfunction

_stex_annotate_force_break:n is provided by the backend config file.

(*End of definition for* `\_stex_annotate_force_break:n`. *This function is documented on page 129.*)

sfunction

\mmlintent
\mmlarg

```
\stex_if_html_backend:TF {
  \cs_new_protected:Npn \mmlintent #1 #2 {
    \stex_annotate:nn{mml:intent={#1}}{#2}
  }
  \cs_new_protected:Npn \mmlarg #1 #2 {
    \stex_annotate:nn{mml:arg={#1}}{#2}
  }
}{
  \cs_new_protected:Npn \mmlintent #1 #2 { #2 }
  \cs_new_protected:Npn \mmlarg #1 #2 { #2 }
}
```

(*End of definition for* `\mmlintent` *and* `\mmlarg`. *These functions are documented on page 129.*)

sfunction

We can now define \stex_persist:n:

```
⟨@@=stex_persist⟩
\bool_if:NTF \c_stex_persist_write_mode_bool {
  \stex_if_html_backend:TF{
    \cs_new:Npn \stex_persist:n #1 {}
    \cs_new:Npn \stex_persist:e #1 {}
  }{
    \cs_new_protected:Nn \stex_persist:n {
      \iow_now:Nn \c__stex_persist_sms_iow {#1}
    }
    \cs_generate_variant:Nn \stex_persist:n {e}
  }
}{
  \cs_new:Npn \stex_persist:n #1 {}
  \cs_new:Npn \stex_persist:e #1 {}
}
```

and style patching:

```
⟨@@=stex_styles⟩
\stex_if_html_backend:TF{
  \cs_new_protected:Nn \stex_style_apply: {}
  \cs_new_protected:Nn \__stex_styles_patch:nnnn {}
  \stex_keys_define:nnnn{ csspatch }{
    \str_clear:N \l_stex_keys_cls_id
    \str_clear:N \l_stex_keys_counter_id
    \str_clear:N \l_stex_keys_counter_parent
  }{
    counter    .str_set_x:N  = \l_stex_keys_counter_id,
    parent     .str_set_x:N  = \l_stex_keys_counter_parent,
    unknown    .code:n       = {
      \exp_args:NNo \str_set:Nn \l_stex_keys_cls_id {\l_keys_key_tl}
    }
  }{}
  \cs_new_protected:Npn \__stex_styles_css_patch:nnnn #1 #2 {
    \stex_keys_set:nn{ csspatch }{ #2 }
    \group_begin:
    \catcode`\ =12\relax
    \catcode`^^J=12\relax
    \__stex_styles_patch_i:nnn {#1}
  }

  \seq_new:N \g__stex_styles_counters_seq

  \cs_new:Nn \__stex_styles_patch_html_c: {
    \stex_html_literal:n{
      <span~data-ftml-counter="\l_stex_keys_counter_id"~style="display:none;"~
        data-ftml-counter-parent="\l_stex_keys_counter_parent"
      ></span>
    }
  }

  \cs_new:Nn \__stex_styles_patch_html: {
    \stex_html_literal:n{
      <span~data-ftml-style="\l_stex_keys_cls_id"~style="display:none;"~
        data-ftml-counter="\l_stex_keys_counter_id"
      ></span>
    }
  }

  \cs_new_protected:Nn \__stex_styles_patch_i:nnn {
    \str_if_empty:NTF \l_stex_keys_cls_id {
      \str_set:Nn \l_stex_keys_cls_id { #1 }
    }{
      \str_set:Ne \l_stex_keys_cls_id { #1-\l_stex_keys_cls_id }
    }
    \exp_args:No \str_case:nnF \l_stex_keys_counter_parent {
      {}{}
      {part}{\str_set:Nn \l_stex_keys_counter_parent { 0 }}
      {chapter}{\str_set:Nn \l_stex_keys_counter_parent { 1 }}
      {section}{\str_set:Nn \l_stex_keys_counter_parent { 2 }}
      {subsection}{\str_set:Nn \l_stex_keys_counter_parent { 3 }}
```

```
      {subsubsection}{\str_set:Nn \l_stex_keys_counter_parent { 4 }}
      {paragraph}{\str_set:Nn \l_stex_keys_counter_parent { 5 }}
      {subparagraph}{\str_set:Nn \l_stex_keys_counter_parent { 6 }}
    }{
      \msg_error:nnx{stex}{error/notasection}\l_stex_keys_counter_parent
    }
    \str_set:Ne \l__stex_styles_first_str { &~.ftml-title-paragraph~ { display:inline-block
    \str_if_empty:NF \l_stex_keys_counter_id {
      %\str_put_left:Ne \l__stex_styles_first_str { counter-increment:~ ftml-\l_stex_keys_c
      \exp_args:NNo \seq_if_in:NnF \g__stex_styles_counters_seq \l_stex_keys_counter_id {
        \seq_gpush:No \g__stex_styles_counters_seq \l_stex_keys_counter_id
        \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
          \__stex_styles_patch_html_c:
          % in body
        }{
          \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html_c:
          % in preamble
        }
      }
      \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
          \__stex_styles_patch_html:
          % in body
      }{
        \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html:
        % in preamble
      }
    }
    \exp_args:Ne \stex_css_literal:n { .ftml-\l_stex_keys_cls_id { \l__stex_styles_first_st

    % TODO export counter reset somehow

    \group_end:
  }
}{
  \cs_new_protected:Nn \__stex_styles_patch:nnnn {
    \str_if_empty:nTF {#2}{
      \tl_set:cn{__stex_styles_style_#1_start:}{#3}
      \tl_set:cn{__stex_styles_style_#1_end:}{#4}
    }{
      \tl_set:cn{__stex_styles_style_#1_#2_start:}{#3}
      \tl_set:cn{__stex_styles_style_#1_#2_end:}{#4}
    }
  }
  \cs_new_protected:Nn \__stex_styles_css_patch:nnnn {}
}
```

# Chapter 28

# Math Archives

⟨@@=stex_mathhub⟩

\c_stex_mathhub_files
\mathhub

```
\str_if_empty:NTF\mathhub{
  \stex_get_env:Nn \l__stex_mathhub_str {MATHHUB}
  \str_if_empty:NTF \l__stex_mathhub_str {
    \ior_open:NnTF \g_tmpa_ior{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{
      \group_begin:
        \escapechar=-1\catcode`\\=12
        \ior_str_get:NN \g_tmpa_ior \l__stex_mathhub_str
        \str_gset_eq:NN \l__stex_mathhub_str \l__stex_mathhub_str
      \group_end:
      \ior_close:N \g_tmpa_ior
      \stex_debug:nn{mathhub}{MathHub~directory~determined~from~home~directory}
    }{
      \str_clear:N \l__stex_mathhub_str
    }
  }{
    \stex_debug:nn{mathhub}{MathHub~directory~determined~from~environment~variable}
  }
}{
  \str_set_eq:NN \l__stex_mathhub_str \mathhub
}

\str_if_empty:NTF \l__stex_mathhub_str {
  \msg_warning:nn{stex}{warning/nomathhub}
  \stex_file_set:Ne \l_tmpa_seq {\stex_file_use:N \c_stex_home_file \tl_to_str:n{/MathHub}}
  \seq_clear:N \c_stex_mathhub_files
  \seq_push:Ne \c_stex_mathhub_files {\stex_file_use:N \l_tmpa_seq}
}{
  \seq_clear:N \c_stex_mathhub_files
  \seq_set_split:NnV \l_tmpa_seq , \l__stex_mathhub_str
  \seq_map_inline:Nn \l_tmpa_seq {
    \stex_file_resolve:Nn \l__stex_mathhub_str {#1}
    \stex_if_file_absolute:NF \l__stex_mathhub_str {
      \stex_file_resolve:Ne \l__stex_mathhub_str {
        \stex_file_use:N \c_stex_pwd_file / #1
      }
```

```
    }
    \seq_put_right:Ne \c_stex_mathhub_files {
      \stex_file_use:N \l__stex_mathhub_str
    }
  }
}

\exp_args:NNe \str_set:Nn \mathhub {\seq_use:Nn \c_stex_mathhub_files ,}
\stex_debug:nn{mathhub}{MATHHUBs:~\mathhub}
```

(*End of definition for* `\c_stex_mathhub_files` *and* `\mathhub`*. These variables are documented on page 132.*)

`\stex_archive_id:N`

```
\cs_new:Npn \stex_archive_id:N {
  \exp_after:wN \use_i:nnn
}
```

(*End of definition for* `\stex_archive_id:N`*. This function is documented on page 132.*)

sfunction

`\stex_archive_base:N`
`\stex_archive_base:n`

```
\cs_new:Npn \stex_archive_base:N {
  \exp_after:wN \use_ii:nnn
}

\cs_new:Nn \stex_archive_base:n {
  \exp_args:NNe \use:nn \use_ii:nnn { \use:c {c_stex_mathhub_#1_archive} }
}
```

(*End of definition for* `\stex_archive_base:N` *and* `\stex_archive_base:n`*. These functions are documented on page 132.*)

sfunction

`\stex_archive_path:N`
`\stex_archive_path:n`

```
\cs_new:Npn \stex_archive_path:N {
  \exp_after:wN \use_iii:nnn
}

\cs_new:Nn \stex_archive_path:n {
  \exp_args:NNe \use:nn \use_iii:nnn { \use:c {c_stex_mathhub_#1_archive} }
}
```

(*End of definition for* `\stex_archive_path:N` *and* `\stex_archive_path:n`*. These functions are documented on page 132.*)

sfunction

`\stex_in_archive:nn`

```
\cs_new_protected:Nn \stex_in_archive:nn {
  \stex_pseudogroup:nn{
    \cs_set:Npn \l__stex_mathhub_cs ##1 {#2}
    \tl_if_empty:nTF{#1}{
      \tl_if_exist:NTF \l_stex_current_archive {
        \exp_args:Ne \l__stex_mathhub_cs {\stex_archive_id:N \l_stex_current_archive }
      }{
```

```
        \l__stex_mathhub_cs {}
      }
    }{
      \__stex_mathhub_set_current:n{#1}
      \l__stex_mathhub_cs {#1}
    }
  }{
    \stex_pseudogroup_restore:N \l_stex_current_archive
    \cs_set:Npn \exp_not:N \l__stex_mathhub_cs ##1 {
      \exp_args:No \exp_not:n {\l__stex_mathhub_cs {##1}}
    }
  }
}

\cs_set:Npn \l__stex_mathhub_cs #1 {}
```

Auxiliary macros for loading archives:

```
\cs_new_protected:Nn \__stex_mathhub_set_current:n {
  \__stex_mathhub_require:n { #1 }
  \stex_debug:nn{mathhub}{switching~to~archive~#1}
  \tl_set_eq:Nc \l_stex_current_archive {
    c_stex_mathhub_#1_archive
  }
}

\cs_new_protected:Nn \_stex_set_meta_archive: {
  \prop_if_exist:cF { c_stex_mathhub_FTML/meta_archive } {
    \seq_if_empty:NF \c_stex_mathhub_files {
      \stex_debug:nn{mathhub}{Opening~archive:~FTML/meta}
      \__stex_mathhub_do_manifest:nn { FTML/meta }{
        \tl_gset:ce {c_stex_mathhub_FTML/meta_archive}{
          {\tl_to_str:n{FTML/meta}}
          {\tl_to_str:n{http://mathhub.info}}
          {}
        }
      }
    }
  }

}

\cs_new_protected:Nn \__stex_mathhub_require:n {
  \prop_if_exist:cF { c_stex_mathhub_#1_archive } {
    \seq_if_empty:NTF \c_stex_mathhub_files {
      \msg_fatal:nn{stex}{warning/nomathhub}
    }{
      \stex_debug:nn{mathhub}{Opening~archive:~#1}
      \__stex_mathhub_do_manifest:nn { #1 }{
        \msg_fatal:nnee{stex}{error/noarchive}
        {#1}{\seq_use:Nn \c_stex_mathhub_files ,}
      }
    }
  }
}
```

Attempts to find the archive's manifest file:

```
\cs_new_protected:Nn \__stex_mathhub_do_manifest:nn {
  \seq_map_inline:Nn \c_stex_mathhub_files {
    \__stex_mathhub_find_manifest:nn {##1}{#1}
    \str_if_empty:NF \l__stex_mathhub_manifest_str \seq_map_break:
  }
  %\exp_args:Ne \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1
  \str_if_empty:NTF \l__stex_mathhub_manifest_str { #2 }{
    \__stex_mathhub_parse_manifest:n {#1}
  }
}

\cs_new_protected:Nn \__stex_mathhub_find_manifest:nn {
  \str_clear:N \l__stex_mathhub_manifest_str
  \seq_set_split:Nnn \l_tmpa_seq / { #1 }
  \seq_set_split:Nnn \l__stex_mathhub_seq / {#1 / #2}
  \bool_set_true:N \l__stex_mathhub_bool
  \bool_while_do:Nn \l__stex_mathhub_bool {
    \tl_if_eq:NNTF \l__stex_mathhub_seq \l_tmpa_seq {
      \bool_set_false:N \l__stex_mathhub_bool
    }{
      \__stex_mathhub_check_manifest:
      \bool_if:NT \l__stex_mathhub_bool {
        \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
      }
    }
  }
}

\cs_new_protected:Nn \__stex_mathhub_check_manifest: {
  \__stex_mathhub_check_manifest:n {MANIFEST.MF}
  \bool_if:NT \l__stex_mathhub_bool {
    \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
    \bool_if:NT \l__stex_mathhub_bool {
      \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
    }
  }
}

\cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
  \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
  \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {
    \bool_set_false:N \l__stex_mathhub_bool
    \str_set:Ne \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
  }
}
```

Parses the archive's manifest file:

```
\ior_new:N \c__stex_mathhub_manifest_ior

\str_set:Nn \l_tmpa_str {https://}

\exp_after:wN \cs_new:Npn \exp_after:wN \__stex_mathhub_replace_https:w \l_tmpa_str #1 \__s
  http:// #1
```

```
}

\cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
  \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
  \str_set:Ne \l__stex_mathhub_file_str {\stex_file_use:N \l__stex_mathhub_seq}
  \str_set:Nn \l__stex_mathhub_id_str {#1}
  \str_set:Nn \l__stex_mathhub_base_str {http://mathhub.info}

  \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
    \exp_args:NNNo \exp_args:NNNx
      \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
    \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
      \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
      \str_set:Ne \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
      \str_case:Nn \l__stex_mathhub_key {
        {id}              { \str_set_eq:NN \l__stex_mathhub_id_str \l__stex_mathhub_val }
        {url-base}        { \str_set_eq:NN \l__stex_mathhub_base_str \l__stex_mathhub_val }
      }
    }
  }
  \ior_close:N \c__stex_mathhub_manifest_ior
  \exp_args:No \stex_str_if_starts_with:nnT \l__stex_mathhub_base_str {https://} {
    \str_set:Ne \l__stex_mathhub_base_str { \exp_after:wN \__stex_mathhub_replace_https:w \
  }
  \tl_gset:ce { c_stex_mathhub_#1_archive } { {\l__stex_mathhub_id_str} {\l__stex_mathhub_b
  \stex_debug:nn{mathhub}{Result:~\use:c{c_stex_mathhub_#1_archive}}
  \str_if_eq:nnF {#1}{main}{
    \stex_persist:e {
      \__stex_mathhub_restore:nn{\l__stex_mathhub_id_str}{\l__stex_mathhub_base_str}
    }
  }
}
```

The `.sms` file persisting content should not store the file path of the archive, since that depends on the directory layout of the system that generated it. We therefore, when restoring archive macros, check if we find the archive on the local system, and if not, leave the directory path empty. This will throw an error iff some functionality at some point *requires* the archive to actually exist locally.

```
\cs_set_protected:Nn \__stex_mathhub_restore:nn {
  \__stex_mathhub_do_manifest:nn { #1 }{}
  \tl_if_exist:cF { c_stex_mathhub_#1_archive }{
    \tl_set:ce{ c_stex_mathhub_#1_archive }{
      {\tl_to_str:n{#1}}
      {\tl_to_str:n{#2}}
      {}
    }
  }
}

```

(*End of definition for* `\stex_in_archive:nn`. *This function is documented on page 132.*)

sfunction

`\c_stex_no_archive_str`
`\c_stex_no_archive`

```
\str_const:Nn \c_stex_no_archive_str { no/archive }

\tl_const:Ne \c_stex_no_archive_uri {
  {\c_stex_no_archive_str}
  {\tl_to_str:n{http://unknown.source}}
  {}
}

\tl_set_eq:cN {c_stex_mathhub_ no/archive _ archive}\c_stex_no_archive_uri
```

(*End of definition for* `\c_stex_no_archive_str` *and* `\c_stex_no_archive`. *These variables are documented on page 133.*)

`\c_stex_main_archive`
`\l_stex_current_archive`

```
\seq_map_inline:Nn \c_stex_mathhub_files {
  \seq_set_split:Nnn \l_tmpa_seq / {#1}
  \stex_if_file_starts_with:NNT \c_stex_pwd_file \l_tmpa_seq {
    \seq_set_eq:NN \l_tmpb_seq \c_stex_pwd_file
    \seq_map_inline:Nn \l_tmpa_seq { \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl }
    \exp_args:Nne \__stex_mathhub_find_manifest:nn {#1} { \stex_file_use:N \l_tmpb_seq }
    \str_if_empty:NTF \l__stex_mathhub_manifest_str {
      \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
    }{
      \__stex_mathhub_parse_manifest:n { main }
      \tl_set_eq:NN \c_stex_main_archive \c_stex_mathhub_main_archive
      \cs_undefine:N \c_stex_mathhub_main_archive
      \tl_set_eq:cN { c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _archive }
        \c_stex_main_archive
      \prop_set_eq:NN \l_stex_current_archive \c_stex_main_archive
      \stex_debug:nn{mathhub}{Current~archive:~
        \stex_archive_id:N \c_stex_main_archive
      }
      \stex_persist:e {
        \__stex_mathhub_restore:nn{\stex_archive_id:N \c_stex_main_archive}{\stex_archive_b
        \tl_gset_eq:Nc \exp_not:N \c_stex_main_archive {c_stex_mathhub_ \stex_archive_id:N
        \tl_gset_eq:NN \exp_not:N \l_stex_current_archive \exp_not:N \c_stex_main_archive
      }
      %\bool_if:NT \c_stex_persist_write_mode_bool {
      %  \tl_put_right:Ne \_stex_persist_read_now: {
      %    \stex_persist:n {{c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _archi
      %      \tl_gset_eq:cN{c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _manife
      %      \tl_gset_eq:NN \exp_not:N \l_stex_current_archive \exp_not:N\c_stex_main_archi
      %    }
      %  }
      %}
    }
    \seq_map_break:
  }
}
\tl_if_exist:NF \c_stex_main_archive {
  \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~directory}
}
```

(*End of definition for* `\c_stex_main_archive` *and* `\l_stex_current_archive`. *These variables are documented on page 133.*)

\stex_source_path:n
\stex_source_path:nn

```
\cs_new:Nn \stex_source_path:n {
  \tl_if_exist:NTF \l_stex_current_archive {
    \stex_archive_path:N \l_stex_current_archive / source \tl_if_empty:nF{#1}{/ #1}
  }{
    \stex_file_use:N \g_stex_current_file / .. \tl_if_empty:nF{#1}{/ #1}
  }
}
\cs_new:Nn \stex_source_path:nn {
  \tl_if_empty:nTF{#1}{
    \stex_source_path:n {#2}
  }{
    \tl_if_exist:cTF {c_stex_mathhub_ #1 _archive } {
      \stex_archive_path:n {#1} / source \tl_if_empty:nF{#2}{/ #2}
    }{
      \stex_file_use:N \g_stex_current_file / .. \tl_if_empty:nF{#2}{/ #2}
    }
  }
}
```

(*End of definition for* `\stex_source_path:n` *and* `\stex_source_path:nn`. *These functions are documented on page 133.*)

sfunction

# Chapter 29

# URIs

```
1116 ⟨@@=stex_uris⟩
```

\stex_use_archive_uri:n

```
1117 \cs_new:Ne \stex_use_archive_uri:n {
1118   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
1119 }
```

(*End of definition for* \stex_use_archive_uri:n. *This function is documented on page 134.*)

sfunction

## 29.1 Document URIs

\stex_use_document_uri:N
\stex_use_document_uri:n

```
1120 \cs_new:Nn \stex_use_document_uri:N {
1121   \exp_after:wN \__stex_uris_doc:nnnnn #1
1122 }
1123 \cs_new:Nn \stex_use_document_uri:n {
1124   \__stex_uris_doc:nnnnn #1
1125 }
1126 \cs_new:Ne \__stex_uris_doc:nnnnn {
1127   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
1128   \exp_not:N \tl_if_empty:nF{#2}{
1129     \c_ampersand_str\tl_to_str:n{p=}#2
1130   }
1131   \c_ampersand_str\tl_to_str:n{d=}#3
1132   \c_ampersand_str\tl_to_str:n{l=}#4
1133   \exp_not:N \tl_if_empty:nF{#5}{
1134     \c_ampersand_str\tl_to_str:n{e=}#5
1135   }
1136 }
```

(*End of definition for* \stex_use_document_uri:N *and* \stex_use_document_uri:n. *These functions are documented on page 134.*)

sfunction

\stex_document_uri_archive:N

```
1137 \cs_new:Nn \stex_document_uri_archive:N {
1138   \exp_after:wN \use_i:nnnnn #1
```

```
}
```

(*End of definition for* `\stex_document_uri_archive:N`. *This function is documented on page 134.*)

sfunction

`\stex_document_uri_path:N`

```
\cs_new:Nn \stex_document_uri_path:N {
  \exp_after:wN \use_ii:nnnnn #1
}
```

(*End of definition for* `\stex_document_uri_path:N`. *This function is documented on page 135.*)

sfunction

`\stex_document_uri_name:N`

```
\cs_new:Nn \stex_document_uri_name:N {
  \exp_after:wN \use_iii:nnnnn #1
}
```

(*End of definition for* `\stex_document_uri_name:N`. *This function is documented on page 135.*)

sfunction

`\stex_document_uri_language:N`

```
\cs_new:Nn \stex_document_uri_language:N {
  \exp_after:wN \use_iv:nnnnn #1
}
```

(*End of definition for* `\stex_document_uri_language:N`. *This function is documented on page 135.*)

sfunction

`\stex_document_uri_element:N`

```
\cs_new:Nn \stex_document_uri_element:N {
  \exp_after:wN \use_v:nnnnn #1
}
```

(*End of definition for* `\stex_document_uri_element:N`. *This function is documented on page 135.*)

sfunction

`\stex_document_uri_with_language:Nn`

```
\cs_new:Npn \stex_document_uri_with_language:Nn {
  \exp_after:wN \__stex_uris_with_language:nnnnnn
}
\cs_new:Nn \__stex_uris_with_language:nnnnnn {
  {#1}{#2}{#3}{#6}{}
}
```

(*End of definition for* `\stex_document_uri_with_language:Nn`. *This function is documented on page 135.*)

sfunction

`\stex_new_document_element_uri:n`

```
\cs_new:Npn \stex_new_document_element_uri:n {
  \exp_after:wN \__stex_uris_with_elem:nnnnnn \l_stex_current_document_uri
}
\cs_new:Nn \__stex_uris_with_elem:nnnnnn {
  {#1}{#2}{#3}{#4}{ \tl_if_empty:nTF{#5}{#6}{#5/#6}}
}
```

(*End of definition for* `\stex_new_document_element_uri:n`. *This function is documented on page 135.*)

sfunction

`\stex_document_uri_from_archive_file:Nn`

```
\cs_new_protected:Nn \stex_document_uri_from_archive_file:Nn {
  \stex_file_set:Nn \l__stex_uris_file {#2}
  \stex_debug:nn{URI}{relative~path:~\stex_file_use:N \l__stex_uris_file}
  \__stex_uris_doc_from_archive_file:NN #1 \l__stex_uris_file
}

\cs_new_protected:Nn \__stex_uris_doc_from_archive_file:NN {
  \__stex_uris_split_file:N #2
  \prop_if_exist:NTF \l_stex_current_archive {
    \str_set:Ne \l__stex_uris_archive {
      \stex_archive_id:N \l_stex_current_archive
    }
  }{
    \str_set_eq:NN \l__stex_uris_archive \c_stex_no_archive_str
  }
  \tl_set:Ne #1 {
    { \l__stex_uris_archive }
    { \stex_file_use:N \l__stex_uris_path_seq }
    { \l__stex_uris_name_str }
    { \l__stex_uris_lang_str }
    {}
  }
}

\cs_new_protected:Nn \__stex_uris_split_file:N {
  \seq_set_eq:NN \l__stex_uris_path_seq #1
  \seq_pop_right:NN \l__stex_uris_path_seq \l__stex_uris_name_str
  \seq_set_split:NnV \l_tmpa_seq . \l__stex_uris_name_str
  \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_lang_str % .tex?
  \seq_if_empty:NTF \l_tmpa_seq {
    \str_set_eq:NN \l__stex_uris_name_str \l__stex_uris_lang_str
    \str_set_eq:NN \l__stex_uris_lang_str \l_stex_current_language_str
  }\__stex_uris_split_file_i:
}

\cs_new_protected:Nn \__stex_uris_split_file_i: {
  \cs_if_eq:NNTF \l__stex_uris_lang_str \q_no_value {
    \str_set_eq:NN \l__stex_uris_lang_str \l_stex_current_language_str
  }{
    \prop_if_in:NoF \c_stex_languages_prop \l__stex_uris_lang_str {
      \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_lang_str % actual language maybe?
      \cs_if_eq:NNTF \l__stex_uris_lang_str \q_no_value {
          \str_set_eq:NN \l__stex_uris_lang_str \l_stex_current_language_str
      }{
        \prop_if_in:NoF \c_stex_languages_prop \l__stex_uris_lang_str {
          \seq_put_right:No \l_tmpa_seq \l__stex_uris_lang_str
          \str_set:Nn \l__stex_uris_lang_str {en}
        }
      }
    }
```

```
  }
  \str_set:Ne \l__stex_uris_name_str {\seq_use:Nn \l_tmpa_seq .}
}

\cs_new_protected:Nn \__stex_uris_split_file_ii: {

}
```

(*End of definition for* `\stex_document_uri_from_archive_file:Nn`. *This function is documented on page 135.*)

sfunction

`\c_stex_main_document_uri`
`\l_stex_current_document_uri`

```
\tl_if_exist:NTF \l_stex_current_archive {
  \str_set:Ne \l_tmpa_str { \stex_archive_path:N \l_stex_current_archive }
  \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
  \seq_set_eq:NN \l_tmpb_seq \c_stex_main_file
  \seq_map_inline:Nn \l_tmpa_seq {
    \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl
  }
  \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl
  \__stex_uris_doc_from_archive_file:NN \l_stex_current_document_uri \l_tmpb_seq
}{
  \__stex_uris_doc_from_archive_file:NN \l_stex_current_document_uri \c_stex_main_file
}

\tl_set_eq:NN \c_stex_main_document_uri \l_stex_current_document_uri
\str_set:Ne \l_stex_current_language_str { \stex_document_uri_language:N \l_stex_current_do

\stex_debug:nn{URIs}{Current~document:~ \stex_use_document_uri:N \c_stex_main_document_uri}
```

(*End of definition for* `\c_stex_main_document_uri` *and* `\l_stex_current_document_uri`. *These variables are documented on page 135.*)

## 29.2 Module URIs

`\stex_use_module_uri:N`
`\stex_use_module_uri:n`

```
\cs_new:Nn \stex_use_module_uri:N {
  \exp_after:wN \__stex_uris_module:nnn #1
}
\cs_new:Nn \stex_use_module_uri:n {
  \__stex_uris_module:nnn #1
}
\cs_new:Ne \__stex_uris_module:nnn {
  \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
  \exp_not:N \tl_if_empty:nF{#2}{
    \c_ampersand_str\tl_to_str:n{p=}#2
  }
  \c_ampersand_str\tl_to_str:n{m=}#3
}
```

(*End of definition for* `\stex_use_module_uri:N` *and* `\stex_use_module_uri:n`. *These functions are documented on page 135.*)

sfunction

\stex_module_uri_archive:N

```
\cs_new:Nn \stex_module_uri_archive:N {
  \exp_after:wN \use_i:nnn #1
}
```

(*End of definition for* `\stex_module_uri_archive:N`. *This function is documented on page 135.*)

sfunction

\stex_module_uri_path:N
\stex_module_uri_path:n

```
\cs_new:Nn \stex_module_uri_path:N {
  \exp_after:wN \use_ii:nnn #1
}
\cs_new:Nn \stex_module_uri_path:n {
  \use_ii:nnn #1
}
```

(*End of definition for* `\stex_module_uri_path:N` *and* `\stex_module_uri_path:n`. *These functions are documented on page 135.*)

sfunction

\stex_module_uri_name:N
\stex_module_uri_name:n

```
\cs_new:Nn \stex_module_uri_name:N {
  \exp_after:wN \use_iii:nnn #1
}

\cs_new:Nn \stex_module_uri_name:n {
  \use_iii:nnn #1
}
```

(*End of definition for* `\stex_module_uri_name:N` *and* `\stex_module_uri_name:n`. *These functions are documented on page 136.*)

sfunction

\stex_module_uri_split_name:NNN
\stex_module_uri_split_name:NNn

```
\cs_new_protected:Npn \stex_module_uri_split_name:NNN #1 {
  \exp_after:wN \__stex_uris_parent:NNnnn \exp_after:wN #1 \exp_after:wN
}

\cs_new_protected:Nn \stex_module_uri_split_name:NNn {
  \__stex_uris_parent:NNnnn #1 #2 #3
}

\cs_new_protected:Nn \__stex_uris_parent:NNnnn {
  \seq_set_split:Nnn #1 / {#5}
  \seq_pop_right:NN #1 #2
  \seq_if_empty:NTF #1 {
    \tl_set:Ne #1 { {#3} {#4} {#3}}
    \tl_clear:N #2
  }{
    \tl_set:Ne #1 { {#3} {#4} {\seq_use:Nn #1 /} }
  }
}
```

(*End of definition for* `\stex_module_uri_split_name:NNN` *and* `\stex_module_uri_split_name:NNn`. *These functions are documented on page 136.*)

sfunction

`\stex_module_uri_as_qm:n`

```
\cs_new:Nn \stex_module_uri_as_qm:n {
  \__stex_uris_as_qm:nnn #1
}
\cs_new:Nn \__stex_uris_as_qm:nnn {
  #1 / #2 ? #3
}
```

(*End of definition for* `\stex_module_uri_as_qm:n`. *This function is documented on page 136.*)

sfunction

`\stex_new_module_uri:n`

```
\cs_new:Npn \stex_new_module_uri:n {
  \stex_if_in_module:TF {
    \exp_after:wN \__stex_uris_new_nested_mod:nnnn \l_stex_current_module_uri
  }{
    \exp_after:wN \__stex_uris_new_mod:nnnnnn \l_stex_current_document_uri
  }
}

\cs_new:Nn \__stex_uris_new_mod:nnnnnn {
  {#1}
  \str_if_eq:nnTF{#3}{#6}{
    {#2}{#3}
  }{
    {#2 \tl_if_empty:nF{#2}/ #3}{ \tl_to_str:n{ #6 } }
  }
}
\cs_new:Nn \__stex_uris_new_nested_mod:nnnn {
  {#1}
  {#2}
  {#3 / \tl_to_str:n{#4}}
}
```

(*End of definition for* `\stex_new_module_uri:n`. *This function is documented on page 136.*)

sfunction

`\stex_uri_from_pair:Nnn`

```
\bool_new:N \l_stex_relative_import_bool
\cs_new_protected:Nn \stex_uri_from_pair:Nnn {
  \bool_set_false:N \l_stex_relative_import_bool
  \stex_debug:nn{uri}{resolving~<#2,#3>~in~\stex_use_document_uri:N \l_stex_current_documen
  \seq_set_split:Nne \l_tmpa_seq ? { \tl_to_str:n {#3} }
  \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_name_str
  \str_clear:N \l__stex_uris_path_str
  \seq_if_empty:NF \l_tmpa_seq {
    \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_path_str
  }
  \tl_if_empty:nTF { #2 }{
    \str_if_empty:NTF \l__stex_uris_path_str
```

```
      \__stex_uris_maybe_relative:N \__stex_uris_absolute:N
    #1
  }{
    \__stex_uris_pair_in_archive:Nn #1 { #2 }
  }
}
%\cs_generate_variant:Nn \stex_uri_from_pair:Nnn {Nee}

\cs_new_protected:Nn \__stex_uris_maybe_relative:N {
  \tl_set:Ne \l_tmpb_tl {
    \stex_document_uri_path:N \l_stex_current_document_uri
  }
  \tl_set:Ne \l_tmpa_tl {
    {
      \stex_document_uri_archive:N \l_stex_current_document_uri
    }
    {
      \l_tmpb_tl
      \exp_args:NNo \exp_args:Nne \str_if_eq:nnF \l__stex_uris_name_str {
        \stex_document_uri_name:N \l_stex_current_document_uri
      }{
        \str_if_empty:NF \l_tmpb_tl / \stex_document_uri_name:N \l_stex_current_document_ur
      }
    }
    { \l__stex_uris_name_str }
  }
  \stex_if_module_exists:NTF \l_tmpa_tl {
    \tl_set_eq:NN #1 \l_tmpa_tl
    \bool_set_true:N \l_stex_relative_import_bool
    \stex_debug:nn{uri}{returning~relative~\stex_use_module_uri:N #1}
  }{
    \__stex_uris_absolute:N #1
  }
}

\cs_new_protected:Nn \__stex_uris_absolute:N {
  \tl_if_exist:NTF \l_stex_current_archive {
    \tl_set:Ne #1 {
      { \stex_archive_id:N \l_stex_current_archive }
      { \l__stex_uris_path_str }
      { \l__stex_uris_name_str }
    }
    \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
  }{
    \__stex_uris_pair_no_archive:N #1
  }
}

\cs_new_protected:Nn \__stex_uris_pair_no_archive:N {
  \stex_file_resolve:Ne \l_tmpa_seq {
    \stex_file_use:N \g_stex_current_file / .. / \l__stex_uris_path_str
  }
  \tl_set:Ne #1 {
    { \c_stex_no_archive_str }
```

```
    { \stex_file_use:N \l_tmpa_seq }
    { \l__stex_uris_name_str }
  }
  \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
}

\cs_new_protected:Nn \__stex_uris_pair_in_archive:Nn {
 \tl_set:Ne #1 {
    { \tl_to_str:n{ #2 } }
    { \l__stex_uris_path_str }
    { \l__stex_uris_name_str }
 }
 \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
}
```

(*End of definition for* `\stex_uri_from_pair:Nnn`. *This function is documented on page 136.*)

`\stex_uri_from_pair:Nn` sfunction

```
\cs_new_protected:Nn \stex_uri_from_pair:Nn {
  \peek_charcode:NTF [ {
    \__stex_uris_parse_i:Nw #1
  }{
    \__stex_uris_parse_ii:Nw #1
  } #2 \__stex_uris_end:
}

\cs_new_protected:Npn \__stex_uris_parse_i:Nw #1 [#2] #3 \__stex_uris_end: {
  \stex_uri_from_pair:Nnn #1 {#2} {#3}
}

\cs_new_protected:Npn \__stex_uris_parse_ii:Nw #1 #2 \__stex_uris_end: {
  \stex_uri_from_pair:Nnn #1 {} {#2}
}
```

(*End of definition for* `\stex_uri_from_pair:Nn`. *This function is documented on page 136.*)

## 29.3 Symbol URIs

`\stex_use_symbol_uri:N` sfunction
`\stex_use_symbol_uri:n`

```
\cs_new:Nn \stex_use_symbol_uri:N {
  \exp_after:wN \__stex_uris_symbol:nnnn #1
}
\cs_new:Nn \stex_use_symbol_uri:n {
  \__stex_uris_symbol:nnnn #1
}
\cs_new:Ne \__stex_uris_symbol:nnnn {
  \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
  \exp_not:N \tl_if_empty:nF{#2}{
    \c_ampersand_str\tl_to_str:n{p=}#2
  }
  \c_ampersand_str\tl_to_str:n{m=}#3
```

```
    \c_ampersand_str\tl_to_str:n{s=}#4
  }
```

(*End of definition for* `\stex_use_symbol_uri:N` *and* `\stex_use_symbol_uri:n`. *These functions are documented on page 136.*)

sfunction

`\stex_new_symbol_uri:n`

```
\cs_new:Nn \stex_new_symbol_uri:n {
  \l_stex_current_module_uri { \tl_to_str:n{ #1 } }
}
```

(*End of definition for* `\stex_new_symbol_uri:n`. *This function is documented on page 136.*)

sfunction

`\stex_new_symbol_uri:nn`

```
\cs_new:Nn \stex_new_symbol_uri:nn {
  #1 { #2 }
}
```

(*End of definition for* `\stex_new_symbol_uri:nn`. *This function is documented on page 136.*)

sfunction

`\stex_symbol_uri_archive:N`

```
\cs_new:Nn \stex_symbol_uri_archive:N {
  \exp_after:wN \use_i:nnnn #1
}
```

(*End of definition for* `\stex_symbol_uri_archive:N`. *This function is documented on page 136.*)

sfunction

`\stex_symbol_uri_path:N`

```
\cs_new:Nn \stex_symbol_uri_path:N {
  \exp_after:wN \use_ii:nnnn #1
}
```

(*End of definition for* `\stex_symbol_uri_path:N`. *This function is documented on page 136.*)

sfunction

`\stex_symbol_uri_module:N`
`\stex_symbol_uri_module:n`

```
\cs_new:Nn \stex_symbol_uri_module:N {
  \exp_after:wN \__stex_uris_first_three:nnnn #1
}
\cs_new:Nn \stex_symbol_uri_module:n {
  \__stex_uris_first_three:nnnn #1
}

\cs_new:Nn \__stex_uris_first_three:nnnn {
  {#1}{#2}{#3}
}
```

(*End of definition for* `\stex_symbol_uri_module:N` *and* `\stex_symbol_uri_module:n`. *These functions are documented on page 137.*)

sfunction

\stex_symbol_uri_name:N
\stex_symbol_uri_name:n

```
\cs_new:Nn \stex_symbol_uri_name:N {
  \exp_after:wN \use_iv:nnnn #1
}
\cs_new:Nn \stex_symbol_uri_name:n {
  \use_iv:nnnn #1
}
```

(*End of definition for* `\stex_symbol_uri_name:N` *and* `\stex_symbol_uri_name:n`. *These functions are documented on page 137.*)

sfunction

# Chapter 30

# Documents

```
1449 ⟨@@=stex_doc⟩
```

\STEXftmllink
\STEXlinkftml
\STEXsetlinkftml

```
1450 \cs_new_protected:Npn \STEXftmllink {
1451   \exp_args:Ne\url{
1452     \stex_use_document_uri:N \l_stex_current_document_uri
1453   }
1454 }
1455
1456 \cs_new_protected:Npn \STEXsetlinkftml #1 {
1457   \tl_gset:Nn \g__stex_doc_ftml_link_text_tl { #1 }
1458 }
1459
1460 \tl_gset:Nn \g__stex_doc_ftml_link_text_tl{
1461   The~
1462   \stex_get_symbol:nF{FTML}{}
1463   \tl_if_empty:NTF \l_stex_get_symbol_uri {FTML}{\sn{FTML}}
1464   {}~ version ~ of ~ this ~ document ~ can ~ be ~ found ~ at\\
1465   \STEXftmllink
1466 }
1467
1468 \NewDocumentCommand \STEXlinkftml { O{} } {
1469   \ifstexhtml\else
1470     \begin{center}
1471       \emph{
1472         \tl_if_empty:nTF{#1}\g__stex_doc_ftml_link_text_tl{#1}
1473       }
1474     \end{center}
1475   \fi
1476 }
```

(*End of definition for* `\STEXftmllink`, `\STEXlinkftml`, *and* `\STEXsetlinkftml`. *These functions are documented on page 138.*)

sfunction

\stexdoctitle

Initial definition (before `\begin{document}`):

```
1477 \tl_new:N \g__stex_doc_title_tl
1478
1479 \str_new:N \g_stex_document_kind_str
```

```
\tl_new:N \g_stex_document_kind_parameters_tl
\str_set:Nn \g_stex_document_kind_str{article}
\stex_if_html_backend:T{
  \AtBeginDocument{
    \exp_args:Nne\stex_html_node:nnn{span}{
      data-ftml-document-kind=\g_stex_document_kind_str
      \g_stex_document_kind_parameters_tl
    }{}
  }
}

\cs_new_protected:Npn \stexdoctitle #1 {
  \tl_gset:Nn \g__stex_doc_title_tl { #1 }
  \global\def\stexdoctitle##1{}
}
```

At begin document, we switch to:

```
\cs_new_protected:Nn \__stex_doc_set_title:n {
  \stex_if_smsmode:F{
    \gdef\stexdoctitle##1{}
    \stex_debug:nn{title}{Setting~title~to:~\tl_to_str:n{#1}}
    \tl_gset:Nn \g__stex_doc_title_tl { #1 }
    \__stex_doc_title_html:
  }
}

\cs_new_protected:Nn \__stex_doc_title_html: {
  \stex_if_do_html:T{
    \stex_annotate_invisible:nn{data-ftml-doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
  }
}

\AtBeginDocument {
  \tl_if_empty:NTF \g__stex_doc_title_tl {
    \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
  }{
    \gdef\stexdoctitle#1{}
    \__stex_doc_title_html:
  }

  \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
  \protected\gdef\maketitle{
    \tl_if_empty:NF \@title {
      \exp_args:No \stexdoctitle \@title
    }
    \__stex_doc_maketitle:
  }
}
```

(*End of definition for* `\stexdoctitle`. *This function is documented on page 138.*)

sfunction

## 30.1 Sectioning

\l_stex_current_section_level_int
\l_stex_current_section_level_str

```
\int_new:N \l_stex_current_section_level_int
\str_set:Nn \l_stex_current_section_level_str {document}
```

(*End of definition for* `\l_stex_current_section_level_int` *and* `\l_stex_current_section_level_str`. *These variables are documented on page 139.*)

\currentsectionlevel
\Currentsectionlevel

```
\cs_set_protected:Npn \currentsectionlevel {
  \stex_if_do_html:TF{
    \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
    \stex_annotate:nn{data-ftml-currentsectionlevel={},data-ftml-capitalize=false}{}
  }{
   \l_stex_current_section_level_str
  }
  \tl_if_exist:NT\xspace\xspace
}

\cs_set_protected:Npn \Currentsectionlevel {
  \stex_if_do_html:TF{
    \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
    \stex_annotate:nn{data-ftml-currentsectionlevel={},data-ftml-capitalize=true}{}
  }{
    \exp_after:wN \str_uppercase:n \l_stex_current_section_level_str
  }
  \tl_if_exist:NT\xspace\xspace
}
```

(*End of definition for* `\currentsectionlevel` *and* `\Currentsectionlevel`. *These functions are documented on page 139.*)

sfunction

`sfragment` (*env.*)

```
\stex_keys_define:nnnn{ sfragment }{
  \tl_clear:N \l_stex_key_short_tl
}{
  short   .tl_set:N   = \l_stex_key_short_tl
}{id}

\NewDocumentEnvironment{sfragment}{ O{} m}{
  \stex_keys_set:nn{sfragment}{#1}
  \__stex_doc_do_section:n{#2}
  \_stex_do_id: % TODO
}{
  \_sfragment_end:
}

\cs_new_protected:Npn \__stex_doc_do_section:n {
  \int_case:nnF \l_stex_current_section_level_int {
    {0}{\cs_if_exist:NTF \thepart {\_sfragment_do_level:nn{part}}{
      \int_incr:N \l_stex_current_section_level_int
      \__stex_doc_do_section:n
```

```
    }}
    {1}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{chapter}}{
      \int_incr:N \l_stex_current_section_level_int
      \__stex_doc_do_section:n
    }}
    {2}{\_sfragment_do_level:nn{section}}
    {3}{\_sfragment_do_level:nn{subsection}}
    {4}{\_sfragment_do_level:nn{subsubsection}}
    {5}{\_sfragment_do_level:nn{paragraph}}
  }{\_sfragment_do_level:nn{subparagraph}}
}

\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_sfragment_do_level:nn {
    \stexdoctitle{#2}
    \par
    \begin{stex_env_node}{section}{data-ftml-section={\int_use:N \l_stex_current_section_le
      \noindent\stex_html_node:nnn{h1}{data-ftml-title={}}{
        \_stex_annotate_force_break:n{#2}
      }\par
  }
  \cs_new_protected:Nn \_sfragment_end: {
    \end{stex_env_node}
  }
}{
  \cs_new_protected:Nn \_sfragment_do_level:nn {
    \stexdoctitle{#2}
    \tl_if_empty:NTF \l_stex_key_short_tl {
      \use:c{#1}
    }{
      \exp_args:Nne \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
    }{#2}

    \int_compare:nNnF \l_stex_current_section_level_int = 6{
      \int_incr:N \l_stex_current_section_level_int
    }
    \tl_set:Nn\l_stex_current_section_level_str{#1}
  }
  \cs_new_protected:Nn \_sfragment_end: {}
}
```

`titlefragment` (*env.*)

```
\bool_new:N \l__stex_doc_titlefragment_bool

\NewDocumentEnvironment{titlefragment}{ O{} m}{
  \stex_keys_set:nn{sfragment}{#1}
  \cs_if_exist:NTF \maketitle {
    \bool_set_true:N \l__stex_doc_titlefragment_bool
    \title{#2}
    \tl_if_empty:NF \l_stex_key_short_tl {
      \cs_if_exist:NT \shorttitle {
        \exp_args:No \shorttitle \l_stex_key_short_tl
      }
    }
```

```
    \stexdoctitle{#2}
    \maketitle
  }{
    \bool_set_false:N \l__stex_doc_titlefragment_bool
    \__stex_doc_do_section:n{#2}
    \_stex_do_id:
  }
}{
    \bool_if:NF \l__stex_doc_titlefragment_bool \_sfragment_end:
}
```

blindfragment (*env.*)

```
\cs_new_protected:Nn \__stex_doc_skip_section_i: {
  \int_case:nn \l_stex_current_section_level_int {
    {0}{\cs_if_exist:NF \thepart {
      \int_incr:N \l_stex_current_section_level_int \__stex_doc_skip_section_i:
    }}
    {1}{\cs_if_exist:NF \thechapter {
      \int_incr:N \l_stex_current_section_level_int \__stex_doc_skip_section_i:
    }}
  }
  \int_compare:nNnF \l_stex_current_section_level_int = 6{
    \int_incr:N \l_stex_current_section_level_int
  }
}

\stex_if_html_backend:TF {
  \cs_new_protected:Nn \__stex_doc_skip_section: {
    \__stex_doc_skip_section_i:
    \begin{stex_annotate_env}{data-ftml-skipsection={\int_use:N \l_stex_current_section_lev
    \_stex_annotate_force_break:n{}
  }
}{
  \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
}

\NewDocumentEnvironment{blindfragment}{}{
  \__stex_doc_skip_section:
}{
  \stex_if_html_backend:T{
    \stex_annotate_invisible:n{~}
    \end{stex_annotate_env}
  }
}
```

\skipfragment

```
\cs_new_protected:Npn \skipfragment {
  \int_case:nnF \l_stex_current_section_level_int {
    {0}{\cs_if_exist:NTF \thepart {\stepcounter{part}}{
      \int_incr:N \l_stex_current_section_level_int
      \skipfragment
    }}
    {1}{\cs_if_exist:NTF \thechapter {\stepcounter{chapter}}{
      \int_incr:N \l_stex_current_section_level_int
```

```
      \skipfragment
    }}
    {2}{\stepcounter{section}}
    {3}{\stepcounter{subsection}}
    {4}{\stepcounter{subsubsection}}
    {5}{\stepcounter{paragraph}}
  }{\stepcounter{subparagraph}}
}
```

(*End of definition for* `\skipfragment`. *This function is documented on page 139.*)

sfunction

`\setsectionlevel`

```
\cs_new_protected:Npn \setsectionlevel #1 {
  \str_case:nnF{#1}{
    {part}{\int_set:Nn \l_stex_current_section_level_int 0}
    {chapter}{\int_set:Nn \l_stex_current_section_level_int 1}
    {section}{\int_set:Nn \l_stex_current_section_level_int 2}
    {subsection}{\int_set:Nn \l_stex_current_section_level_int 3}
    {subsubsection}{\int_set:Nn \l_stex_current_section_level_int 4}
    {paragraph}{\int_set:Nn \l_stex_current_section_level_int 5}
  }{
    \int_set:Nn \l_stex_current_section_level_int 6
  }
  \stex_if_do_html:T{
    \cs_if_eq:NNTF\@onlypreamble\@notprerr{
      \stex_html_literal:n{<span~ data-ftml-sectionlevel="\int_use:N\l_stex_current_section
    }{
      \AtBeginDocument{
        \stex_html_literal:n{<span~ data-ftml-sectionlevel="\int_use:N\l_stex_current_secti
      }
    }
  }
}
```

In HTML mode, we make sure to record the top section level:

```
\stex_if_html_backend:T{
  \cs_new_protected:Nn \__stex_doc_check_topsect: {
    \int_case:nnF \l_stex_current_section_level_int {
      {0}{\cs_if_exist:NTF \thepart {
        \stex_annotate_invisible:nn{data-ftml-sectionlevel=0}{}
      }{
        \int_incr:N \l_stex_current_section_level_int
        \__stex_doc_check_topsect:
      }}
      {1}{\cs_if_exist:NTF \thechapter {
        \stex_annotate_invisible:nn{data-ftml-sectionlevel=1}{}
      }{
        \int_incr:N \l_stex_current_section_level_int
        \__stex_doc_check_topsect:
      }}
    }{
      \stex_annotate_invisible:nn{data-ftml-sectionlevel={\int_use:N\l_stex_current_section
    }
  }
```

```
  \AtBeginDocument{\__stex_doc_check_topsect:
    \cs_if_exist:NT \thechapter {
      \int_case:nnT \l_stex_current_section_level_int{
        {0}{}
        {1}{}
      }{
        \stex_css_literal:n {
          .ftml-section {
            &~.ftml-title-section { &::before {
              content:~counter(ftml-chapter)~"."~counter(ftml-section)~"~";
            } }
          }
          .ftml-subsection {
            &~.ftml-title-subsection { &::before {
              content:~counter(ftml-chapter)~"."~counter(ftml-section)~"."~counter(ftml-sub
            } }
          }
          .ftml-subsubsection {
            &~.ftml-title-subsubsection { &::before {
              content:~counter(ftml-chapter)~"."~counter(ftml-section)~"."~counter(ftml-sub
            } }
          }
        }
      }
    }
  }
```

(*End of definition for* `\setsectionlevel`. *This function is documented on page 139.*)

sfunction

We make sure `\frontmatter` and `\backmatter` are always defined. TODO: this seems like a hack and probably shouldn't be here

```
\bool_if:NF \c_stex_no_frontmatter_bool {
  \cs_new_protected:Nn \__stex_doc_x_matter: {
    \cs_if_exist:NTF\frontmatter{
      \let\__stex_doc_orig_frontmatter\frontmatter
      \let\__stex_doc_orig_endfrontmatter\endfrontmatter
      \let\endfrontmatter\relax
      \let\frontmatter\relax
    }{
      \tl_set:Nn\__stex_doc_orig_frontmatter{
        \clearpage
        %\@mainmatterfalse
        \pagenumbering{roman}
      }
      \tl_set:Nn\__stex_doc_orig_endfrontmatter{}
    }
    \cs_if_exist:NTF\backmatter{
      \let\__stex_doc_orig_backmatter\backmatter
      \let\__stex_doc_orig_endbackmatter\endbackmatter
      \let\backmatter\relax
      \let\endbackmatter\relax
    }{
      \tl_set:Nn\__stex_doc_orig_backmatter{
```

```
        \clearpage
        %\@mainmatterfalse
        \pagenumbering{roman}
      }
    }
    \newenvironment{frontmatter}{
      \__stex_doc_orig_frontmatter
    }{
      \cs_if_exist:NTF\mainmatter{
        \mainmatter
      }{
        \clearpage
        %\@mainmattertrue
        \pagenumbering{arabic}
      }
    }
    \newenvironment{backmatter}{
      \__stex_doc_orig_backmatter
    }{
      \cs_if_exist:NTF\mainmatter{
        \mainmatter
      }{
        \clearpage
        %\@mainmattertrue
        \pagenumbering{arabic}
      }
    }
  }
   \AddToHook{begindocument/end}\__stex_doc_x_matter:
}
```

## 30.2 Inter-Document References

⟨@@=stex_refs⟩

The .sref-file:

```
\iow_new:N \c__stex_refs_iow
\AtBeginDocument{ \iow_open:Nn \c__stex_refs_iow {\jobname.sref} }
\AtEndDocument{\iow_close:N \c__stex_refs_iow}
\str_set:Ne \c__stex_refs_iow_str { \stex_file_use:N \c_stex_pwd_file /  }
```

Keys:

```
\stex_keys_define:nnnn{sref / 1}{
  \tl_clear:N \l_stex_key_pre_tl
  \tl_clear:N \l_stex_key_post_tl
  \tl_clear:N \l_stex_key_fallback_tl
}{
  % TODO get rid of this
  fallback  .tl_set:N = \l_stex_key_fallback_tl,
  pre       .code:n = {},
  post      .code:n = {}
}{archive file}
\stex_keys_define:nnnn{sref / 2}{}{}{archive file, title}
```

Target declarations:

\sreflabel

\stex_ref_new_doc_target:n

```
\cs_new_protected:Nn \stex_ref_new_doc_target:n {
  \stex_if_smsmode:F{
    \tl_set:Ne \l__stex_refs_new_tl { \stex_new_document_element_uri:n { #1 } }
    \exp_args:Ne \label { \stex_use_document_uri:N \l__stex_refs_new_tl }
    \exp_args:Nne \STeXInternalNewSRefLabel{#1}{ \stex_use_document_uri:N \l__stex_refs_new
    \iow_now:Ne \@auxout {
      \STeXInternalNewSRefLabel{#1}{ \stex_use_document_uri:N \l__stex_refs_new_tl }
    }
    \iow_now:Ne \c__stex_refs_iow {
      \exp_not:N \STeXInternalSRefLabel
        { #1 }
        { \stex_use_document_uri:N \l__stex_refs_new_tl }
        { \exp_not:o \@currentHref}
        { \exp_not:o \@currentlabelname }
    }
  }
}

\NewDocumentCommand \sreflabel {m} { \stex_ref_new_doc_target:n{#1} }
```

(*End of definition for* `\sreflabel` *and* `\stex_ref_new_doc_target:n`. *These functions are documented on page 86.*)

sfunction

\stex_ref_new_sym_target:n

```
\cs_new_protected:Nn \stex_ref_new_sym_target:n {
  \stex_if_smsmode:F{
    \cs_if_exist:cTF{r@\tl_to_str:n{#1}}{
      \cs_if_exist:cT{__stex_sref_aux_sym_ \tl_to_str:n{#1}}{
        \cs_undefine:c{__stex_sref_aux_sym_ \tl_to_str:n{#1}}
        \exp_args:Ne\label { \tl_to_str:n{#1} }
      }
    }{ \exp_args:Ne\label { \tl_to_str:n{#1} } }
  }
}
```

(*End of definition for* `\stex_ref_new_sym_target:n`. *This function is documented on page* **??**.)

sfunction

Inserting References:

\sref

```
\NewDocumentCommand \sref { O{} m O{}}{
  \stex_keys_set:nn { sref / 1 }{ #1 }
  \tl_clear:N \l__stex_refs_tmp
  \str_if_empty:NTF \l_stex_key_file_str {
    \__stex_refs_local:n
  }{
    \__stex_refs_remote:nn{#3}
  }{#2}
}

\cs_new_protected:Nn \__stex_refs_local:n {
  \stex_debug:nn{sref}{Checking~#1}
```

```
  \tl_if_exist:cTF{ g_stex_sref_label_ #1 }{
    %\exp_args:Ne \stex_annotate:nn{data-ftml-sref={\use:c{ g_stex_sref_label_ #1 }}}{
      \exp_args:Ne \__stex_refs_local_ref:n {\use:c{ g_stex_sref_label_ #1 }}
    %}
  }{
    \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
  }
}

\cs_new_protected:Nn \__stex_refs_local_ref:n {
  \cs_if_exist:cTF{autoref}{
    \l_stex_key_pre_tl \autoref{#1} \l_stex_key_post_tl
  }{
    \message{^^J^^J
      sTeX~Warning: \c_backslash_str sref ~ with ~ local ~ target ~ used ~ without ~ hyperr
    ^^J^^J}
    \l_stex_key_pre_tl \ref{#1} \l_stex_key_post_tl
  }
}

\cs_new_protected:Nn \__stex_refs_remote:nn {
  \str_set:Nn \l__stex_refs_key { #2 }
  \tl_set:Nn \l__stex_refs_in { #1 }
  \exp_args:No \stex_in_archive:nn \l_stex_key_archive_str {
    \exp_args:NNo \stex_document_uri_from_archive_file:Nn \l__stex_refs_uri {\l_stex_key_fi
    \stex_debug:nn{sref}{Checking~#2~from~\stex_use_document_uri:N \l__stex_refs_uri}
    \tl_if_eq:NNTF \l__stex_refs_uri \c_stex_main_document_uri {
      \stex_debug:nn{sref}{From~current~document;~delegating~to~\string\autoref}
      \__stex_refs_local:n{ #2 }
    }{
      \str_set:Ne \l__stex_refs_uri { \stex_use_document_uri:N \l__stex_refs_uri }
      \str_set:Ne \l__stex_refs_file { \exp_args:No \stex_source_path:n {\l_stex_key_file_s
      \exp_args:No \IfFileExists \l__stex_refs_file {
        \__stex_refs_find:
      }{}
      \tl_if_empty:NTF \l__stex_refs_tmp {
        \stex_debug:nn{sref}{Not~found}
        \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
      }{
        \str_set_eq:NN \l__stex_refs_uri \l__stex_refs_tmp
        \stex_debug:nn{sref}{Found~ \l__stex_refs_uri}
        \__stex_refs_maybe_in:
      }
    }
  }
}

\cs_new_protected:Nn \__stex_refs_find: {
  \setbox0\vbox\bgroup
    \cs_set_eq:NN \STeXInternalSRefLabel \__stex_refs_check_i:nnnn
    \use:c{@ @ input}{\l__stex_refs_file}
    \exp_args:NNe \use:nn
  \egroup {
    \tl_set:Nn \exp_not:N \l__stex_refs_tmp { \l__stex_refs_tmp }
```

```
  }
}

\cs_new_protected:Nn \__stex_refs_check_i:nnnn {
  \exp_args:No \str_if_eq:nnT{ \l__stex_refs_key }{#1}{
    \exp_args:Nno \stex_str_if_starts_with:nnT{#2}{\l__stex_refs_uri}{
      \str_set:Nn \l__stex_refs_tmp {#2}
      \endinput
    }
  }
}

\cs_new_protected:Nn \__stex_refs_maybe_in: {
  \tl_if_empty:NTF\l__stex_refs_in {
    \cs_if_exist:cTF{r@\l__stex_refs_uri}{
      \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
    }{
      \tl_if_empty:NTF \l__stex_refs_default_file {
        \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
      }{
        \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title
        \str_set_eq:NN \l__stex_refs_file \l__stex_refs_default_file
        \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive
        \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_relpath
        \str_if_eq:NNTF \l__stex_refs_default_file \c__stex_refs_iow_str {
          \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
        }{
          \stex_debug:nn{sref}{in~title=\exp_args:No\exp_not:n\l_stex_key_title_tl,file=\l_
          \__stex_refs_in:
        }
      }
    }
  }{
    \stex_debug:nn{sref}{in~\exp_args:No\exp_not:n\l__stex_refs_in,archive=\l_stex_key_arch
    \exp_args:No \stex_in_archive:nn \l_stex_key_archive_str {
      \str_set:Ne \l__stex_refs_file { \exp_args:No \stex_source_path:n {\l_stex_key_file_s
    }
    \exp_args:Nno \stex_keys_set:nn{ sref / 2 }{ \l__stex_refs_in }
    \__stex_refs_in:
  }
}

\cs_new_protected:Nn \__stex_refs_in: {
  \tl_clear:N \l__stex_refs_tmp
  \exp_args:No \IfFileExists \l__stex_refs_file {
      \__stex_refs_find_in:
  }{
    \stex_debug:nn{sref}{not~found:~\l__stex_refs_file}
  }
  \tl_if_empty:NTF\l__stex_refs_tmp{
    \stex_debug:nn{sref}{Not~found}
    \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
  }{
    \stex_debug:nn{sref}{found:~\meaning\l__stex_refs_tmp}
```

```
    \tl_set:Ne \l__stex_refs_tmp {
      \exp_after:wN \__stex_refs_in_text:nn \l__stex_refs_tmp
    }
    \stex_if_html_backend:TF{
      \exp_args:No \stex_in_archive:nn \l_stex_key_archive_str {
        \exp_args:NNo \stex_document_uri_from_archive_file:Nn \l__stex_refs_in \l_stex_key_
        \exp_args:Ne \stex_annotate:nn{data-ftml-sref={\l__stex_refs_uri},data-ftml-sref-in
          \l__stex_refs_tmp
        }
      }
    }{
      \cs_if_exist:NT \href {\exp_args:No \href \l__stex_refs_uri }\l__stex_refs_tmp
    }
  }
}

\cs_new:Nn \__stex_refs_in_text:nn {
  \__stex_refs_in_text:w #1 \__stex_refs_stop:
  \tl_if_empty:nF{#2}{~(\exp_not:n{#2})}
  \tl_if_empty:NF \l_stex_key_title_tl {
    {}~in~\exp_args:No \exp_not:n \l_stex_key_title_tl
  }
}

\cs_new:Npn \__stex_refs_in_text:w #1.#2 \__stex_refs_stop: {
  \__stex_refs_uppercase:n #1~#2~
}

\cs_new:Nn \__stex_refs_uppercase:n { \uppercase{#1}}

\cs_new_protected:Nn \__stex_refs_find_in: {
  \stex_debug:nn{sref}{finding~\l__stex_refs_uri;~in~\l__stex_refs_file...?}
  \setbox0\vbox\bgroup
    \catcode`^^J=9\relax
    \cs_set_eq:NN \STeXInternalSRefLabel \__stex_refs_check_in:nnnn
    \use:c{@ @ input}{\l__stex_refs_file}
    \exp_args:NNe \use:nn
  \egroup {
    \tl_set:Nn \exp_not:N \l__stex_refs_tmp { \exp_args:No \exp_not:n \l__stex_refs_tmp }
  }
}

\cs_new_protected:Nn \__stex_refs_check_in:nnnn {
  \exp_args:No \str_if_eq:nnT{ \l__stex_refs_uri }{#2}{
    \tl_set:Nn \l__stex_refs_tmp { {#3}{#4} }
    \endinput
  }
}
```

(*End of definition for* `\sref`. *This function is documented on page 86.*)

sfunction

TODO

```
%\int_new:N \l__stex_refs_unnamed_counter_int

```

```
\cs_new:Npn \STeXInternalSRefLabel #1 #2 #3 #4 {}

\cs_new_protected:Npn \STeXInternalNewSRefLabel #1 #2 {
  \tl_gset:ce{ g_stex_sref_label_ #1 }{ \tl_to_str:n{ #2 } }
}

\tl_new:N \l__stex_refs_default_title
\str_new:N \l__stex_refs_default_file
\str_new:N \l__stex_refs_default_archive
\str_new:N \l__stex_refs_default_relpath

\newcommand\srefsetin[3][]{
  \str_set:Ne \l__stex_refs_default_relpath { #2 }
  \tl_if_empty:nTF{#1}{
    \str_set:Ne \l__stex_refs_default_archive { \stex_archive_id:N \c_stex_main_archive }
  }{
    \str_set:Nn \l__stex_refs_default_archive { #1 }
  }
  \exp_args:No \stex_in_archive:nn \l__stex_refs_default_archive {
    \str_set:Ne \l__stex_refs_default_file { \exp_args:No \stex_source_path:n {\l__stex_ref
  }
  \str_set:Nn \l__stex_refs_default_title { #3 }
}
\NewDocumentCommand \extref { O{} m m}{???}

\cs_new_protected:Nn \stex_ref_new_symbol:n {}


\NewDocumentCommand \srefsym { m m }{
  \stex_get_symbol:n { #1 }
  \exp_args:Ne \srefsymuri { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }{ #2 }
}

\cs_new_protected:Npn \srefsymuri #1 {
  \cs_if_exist:cTF{r@\tl_to_str:n{#1}}{
    \cs_if_exist:NT \hyperlink { \hyperlink{#1} }
  }{
    \cs_if_exist:NT \href {\href{#1} }
  }
}
```

## 30.3 Inputting From MathHub Resources

```
⟨@@=stex_inputs⟩
```

\inputref

```
\NewDocumentCommand \inputref { O{} m}{
  \stex_in_archive:nn{#1}{
    \stex_document_uri_from_archive_file:Nn \l__stex_inputs_uri {#2}
    \__stex_inputs_ref:n{#2}
```

```
  }
}

\stex_if_html_backend:TF{
  \str_set:Nn \c__stex_inputs_ref_pre_str {<span~data-ftml-inputref="}
  \str_set:Nn \c__stex_inputs_ref_post_str {"~><!----></span>}
  \cs_new_protected:Nn \__stex_inputs_ref_i:n {
    \IfFileExists{\stex_source_path:n {#1}}{
      %\relax
      %\noindent\vbox{
        \exp_args:Ne\stex_html_literal:n{
          \c__stex_inputs_ref_pre_str
          \stex_use_document_uri:N\l__stex_inputs_uri
          \c__stex_inputs_ref_post_str
        }
      %}
    }{
      \stex_input_with_hooks:Ne\l__stex_inputs_uri {\stex_source_path:n {#1}}
    }
  }
  \cs_new_protected:Nn \__stex_inputs_ref:n {
    \ifnum\@listdepth>0
      \noindent\vbox{\vskip-26pt\relax\__stex_inputs_ref_i:n{#1}}\par
      %\begin{stex_env_node}{div}{style:foo=bar}\vskip-700pt\end{stex_env_node}
    \else
      \__stex_inputs_ref_i:n{#1}
    \fi
  }

}{
  \cs_new_protected:Nn \__stex_inputs_ref:n {
    \begingroup
      \inputreftrue
      \let \l_stex_metatheory_uri \c_stex_default_metatheory
      %\seq_clear:N \l_stex_all_modules_seq
      \stex_undefine:N \l_stex_current_module_uri
      \stex_debug:nn{inputref}{Inputrefing~document~\stex_use_document_uri:N \l__stex_input
      \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#1}}
    \endgroup\medskip
  }
}
```

(*End of definition for* `\inputref`. *This function is documented on page 140.*)

sfunction

`\mhinput`

```
\NewDocumentCommand \mhinput { O{} m}{
  \stex_in_archive:nn {#1} {
    \stex_document_uri_from_archive_file:Nn \l__stex_inputs_uri {#2}
    \ifinputref
      \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#2}}
    \else
      \inputreftrue
      \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#2}}
```

```
      \inputreffalse
    \fi
  }
}
```

(*End of definition for* `\mhinput`. *This function is documented on page 140.*)

sfunction

`\ifinputref`

```
\newif \ifinputref \inputreffalse
```

(*End of definition for* `\ifinputref`. *This function is documented on page 140.*)

sfunction

`\IfInputref`

```
\stex_if_html_backend:TF{
  \newcommand \IfInputref[2]{
    \stex_annotate:nn{data-ftml-ifinputref=true}{#1}
    \stex_annotate:nn{data-ftml-ifinputref=false}{#2}
  }
}{
  \newcommand \IfInputref[2]{
    \ifinputref #1 \else #2 \fi
  }
}
```

(*End of definition for* `\IfInputref`. *This function is documented on page 140.*)

sfunction

`\libinput`

```
\newcommand \libinput [2][] {
  \stex_in_archive:nn{#1}{
    \__stex_inputs_up_archive:nn{#2}{tex}
    \stex_debug:nn{lib}{Result:~\seq_use:Nn \l__stex_inputs_libinput_files_seq ,}
    \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
      \msg_error:nnnn{stex}{error/nofile}{\libinput}{#2.tex}
    }{
      \seq_map_function:NN \l__stex_inputs_libinput_files_seq \input
    }
  }
}
```

(*End of definition for* `\libinput`. *This function is documented on page 140.*)

sfunction

`\libusepackage`

```
\newcommand\libusepackage[2][]{
  \__stex_inputs_up_archive:nn{#2}{sty}
  \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
    \str_set:Ne \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
    \exp_args:Nne \use:n {\usepackage[#1]} {
      \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
    }
  }{
    \msg_fatal:nnnn{stex}{error/nofile}{\libusepackage}{#2.sty}
```

```
  }
}
```

(*End of definition for* `\libusepackage`. *This function is documented on page 140.*)

sfunction

`\libusetikzlibrary`

```
\newcommand \libusetikzlibrary [2][] {
  \cs_if_exist:NF \usetikzlibrary {
    \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
  }
  \tl_if_empty:nTF{#1}{
    \__stex_inputs_usetikzlibrary:n{#2}
  }{
    \stex_in_archive:nn{#1}{\__stex_inputs_usetikzlibrary:n{#2}}
  }
}

\cs_new_protected:Nn \__stex_inputs_usetikzlibrary:n{
  \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
  \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
    \exp_args:Nne \__stex_inputs_usetikzlibrary_i:nn{#1}{ \seq_item:Nn \l__stex_inputs_libi
  }{
    \msg_fatal:nnnn{stex}{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
  }
}

\cs_new_protected:Nn \__stex_inputs_usetikzlibrary_i:nn {
  \pgfkeys@spdef\pgf@temp{#1}
  \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
  \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfu
  \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode`\@}
  \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode`\|}
  \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode`\$}
  \catcode`\@=11
  \catcode`\|=12
  \catcode`\$=3
  \pgfutil@InputIfFileExists{#2}{}{}
  \catcode`\@=\csname tikz@library@#1@atcode\endcsname
  \catcode`\|=\csname tikz@library@#1@barcode\endcsname
  \catcode`\$=\csname tikz@library@#1@dollarcode\endcsname
}
```

(*End of definition for* `\libusetikzlibrary`. *This function is documented on page 140.*)

sfunction

`\addmhbibresource`

```
\newcommand \addmhbibresource [2][] {
  \stex_in_archive:nn{#1}{
    \__stex_inputs_up_archive:nn{#2}{bib}
    \stex_debug:nn{lib}{Result:~\seq_use:Nn \l__stex_inputs_libinput_files_seq ,}
    \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
      \msg_error:nnnn{stex}{error/nofile}{\addmhbibresource}{#2.bib}
    }{
```

```
      \seq_map_function:NN \l__stex_inputs_libinput_files_seq \addbibresource
    }
  }
}
```

(*End of definition for* `\addmhbibresource`. *This function is documented on page 141.*)

sfunction

`\__stex_inputs_up_archive:nn` Iterates up the current archive's directory in search for `lib` files with name `#1.#2`

```
\cs_new_protected:Nn \__stex_inputs_up_archive:nn {
  \tl_if_exist:NF \l_stex_current_archive {
    \msg_error:nnn{stex}{error/notinarchive}\libinput
  }
  \seq_clear:N \l__stex_inputs_libinput_files_seq
  \seq_set_split:Nne \l__stex_inputs_path_seq / {\stex_archive_path:N \l_stex_current_archi
  \seq_set_split:Nne \l__stex_inputs_id_seq / {\stex_archive_id:N \l_stex_current_archive}
  \seq_map_inline:Nn \l__stex_inputs_id_seq {
    \seq_pop_right:NN \l__stex_inputs_path_seq \l_tmpa_tl
  }
  \stex_debug:nn{lib}{Checking~#1~in~\seq_use:Nn \l__stex_inputs_path_seq /}

  \bool_while_do:nn { ! \seq_if_empty_p:N \l__stex_inputs_id_seq }{
    \str_set:Ne \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-
    \stex_debug:nn{lib}{Checking~\l__stex_inputs_path_str}
    \IfFileExists{ \l__stex_inputs_path_str }{
      \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_
        \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
      }
    }{}
    \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str
    \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str
  }

  \str_set:Ne \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #
  \stex_debug:nn{lib}{Checking~\l__stex_inputs_path_str}
  \IfFileExists{ \l__stex_inputs_path_str }{
    \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_st
      \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
    }
  }{}
}
```

(*End of definition for* `\__stex_inputs_up_archive:nn`.)

`\mhgraphics`
`\cmhgraphics`

```
\ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
  \define@key{Gin}{archive}{
    \stex_in_archive:nn{#1}{}
    \str_set:Ne \Gin@mhrepos {
      \stex_source_path:nn{#1}{}
    }
  }
  \providecommand\mhgraphics[2][]{
    \str_set:Ne \Gin@mhrepos {
```

```
      \stex_source_path:n{}
    }
    \setkeys{Gin}{#1}
    \includegraphics[#1]{ \Gin@mhrepos / #2 }
  }
  \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
}
```

(*End of definition for* `\mhgraphics` *and* `\cmhgraphics`. *These functions are documented on page 141.*)

sfunction

`\lstinputmhlisting`
`\clstinputmhlisting`

```
\ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
  \define@key{lst}{archive}{
    \stex_in_archive:nn{#1}{}
    \str_set:Ne \lst@mhrepos{
      \stex_source_path:nn{#1}{}
    }
  }
  \newcommand\lstinputmhlisting[2][]{%
    \str_set:Ne \lst@mhrepos{
      \stex_source_path:n{}
    }
    \setkeys{lst}{#1}%
    \lstinputlisting[#1]{\lst@mhrepos / #2}}
  \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
}
```

(*End of definition for* `\lstinputmhlisting` *and* `\clstinputmhlisting`. *These functions are documented on page 141.*)

sfunction

`\mhtikzinput`
`\cmhtikzinput`

```
\ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
  \define@key{Gin}{archive}{
    \stex_in_archive:nn{#1}{}
    \str_set:Ne \Gin@mhrepos{
      \stex_source_path:nn{#1}{}
    }
    \str_set:Ne \l__stex_inputs_gin_repo_str {#1}
  }
  \newcommand\mhtikzinput[2][]{%
    \str_clear:N \l__stex_inputs_gin_repo_str
    \str_set:Ne \Gin@mhrepos{
      \stex_source_path:n{}
    }
    \setkeys{Gin}{#1}%
    \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
      \tikzinput[#1]{\Gin@mhrepos / #2}
    }
  }
  \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
}
```

(*End of definition for* `\mhtikzinput` *and* `\cmhtikzinput`. *These functions are documented on page 141.*)

sfunction

# Chapter 31

# Modules

⟨@@=stex_modules⟩

A module consists of four separate macros, storing its *symbols*, *incoming morphisms* (e.g. \importmodule), *notations*, and "initialization code", respectively:

```
\cs_new:Nn \__stex_modules_macro_short:nnn {
  #1 ? #2 ? #3
}
\cs_new:Nn \_stex_symbol_macro:N {
  c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _symbols_prop
}
\cs_new:Nn \_stex_symbol_macro:n {
  c_stex_module_ \__stex_modules_macro_short:nnn #1 _symbols_prop
}
\cs_new:Nn \_stex_morphisms_macro:N {
  c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _morphisms_prop
}
\cs_new:Nn \_stex_morphisms_macro:n {
  c_stex_module_ \__stex_modules_macro_short:nnn #1 _morphisms_prop
}
\cs_new:Nn \_stex_notations_macro:N {
  c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _notations_prop
}
\cs_new:Nn \_stex_notations_macro:n {
  c_stex_module_ \__stex_modules_macro_short:nnn #1 _notations_prop
}
\cs_new:Nn \_stex_module_code_macro:N {
  c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _code
}
\cs_new:Nn \_stex_module_code_macro:n {
  c_stex_module_ \__stex_modules_macro_short:nnn #1 _code
}
```

\l_stex_current_module_uri initially undefined.

(*End of definition for* \l_stex_current_module_uri*. This variable is documented on page 142.*)

\l_stex_all_modules_seq

```
\seq_new:N \l_stex_all_modules_seq
```

(*End of definition for* \l_stex_all_modules_seq*. This variable is documented on page 142.*)

smodule (*env.*)

```
\tl_new:N \l_stex_metatheory_uri

\stex_keys_define:nnnn{smodule}{
  \str_clear:N \l_stex_key_sig_str
}{
  meta          .code:n       = {
    \tl_if_empty:nTF {#1}{
      \tl_clear:N \l_stex_metatheory_uri
    }{
      \stex_uri_from_pair:Nn \l_stex_metatheory_uri { #1 }
    }
  },
  %ns            .code:n       = {
  %  \stex_uri_resolve:Ne \l_stex_current_ns_uri { #1 }
  %} ,
  %lang          .code:n       = {
  %  \stex_set_language:n { #1 }
  %} ,
  sig           .str_set_x:N  = \l_stex_key_sig_str ,
  %creators      .code:n       = {} , % todo ?
  %contributors  .code:n       = {} , % todo ?
}{id, title, style, deprecate}

\stex_if_html_backend:T {
  \cs_new_protected:Nn \__stex_modules_html_annots: {
    \exp_args:Ne \stex_annotate_env {
      data-ftml-module={\stex_module_uri_name:N \l_stex_current_module_uri}
      %data-ftml-language={ \l_stex_current_language_str},
      \str_if_empty:NF\l_stex_key_sig_str {, data-ftml-signature={\l_stex_key_sig_str}}
      \tl_if_empty:NF \l_stex_metatheory_uri {,
        data-ftml-metatheory={\stex_use_module_uri:N \l_stex_metatheory_uri}
      }
    }
    \stex_annotate_invisible:n{}
  }
}

\stex_new_stylable_env:nnnnnnn {module} {O{} m}{
  \stex_keys_set:nn { smodule }{ #1 }
  \tl_set_eq:NN \thistitle \l_stex_key_title_tl
  \tl_if_empty:NF \thistitle {
    \exp_args:No \stexdoctitle \thistitle
  }
  \stex_module_setup:n { #2 }

  \stex_if_do_html:T \__stex_modules_html_annots:
  \stex_if_smsmode:F {
    \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_current_module_uri }
    \str_set:Nn \thismodulename {#2}
    \stex_style_apply:
  }
  \stex_smsmode_do:
}{
```

```
  \stex_close_module:
  \stex_if_smsmode:F \stex_style_apply:
  \stex_if_do_html:T{ \endstex_annotate_env }
}{}{}{s}
```

\stex_module_setup:n

```
\cs_new_protected:Npn \stex_module_setup:n {
  \stex_if_in_module:TF \__stex_modules_nested:n \__stex_modules_top:n
}

\cs_new_protected:Nn \__stex_modules_top:n {
  \str_if_empty:NTF \l_stex_key_sig_str
    \_stex_module_setup_top_nosig:n \__stex_modules_top_sig:n {#1}
  \g_stex_every_module_tl
  \tl_if_empty:NF \l_stex_metatheory_uri {
    \stex_debug:nn{modules}{loading~metatheory~\stex_use_module_uri:N \l_stex_metatheory_ur
    \__stex_modules_load_meta:
  }
}
```

(*End of definition for* \stex_module_setup:n*. This function is documented on page 142.*)

sfunction

\stex_module_setup_top_nosig:n

```
\cs_new_protected:Nn \_stex_module_setup_top_nosig:n {
  \stex_metagroup_new:
  \tl_set:Ne \l__stex_modules_uri { \stex_new_module_uri:n {#1} }
  \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
  \exp_args:No \stex_if_module_exists:NTF \l__stex_modules_uri {
    \stex_debug:nn{modules}{(already exists)}
  }{
    \tl_gclear:c{ \_stex_module_code_macro:N \l__stex_modules_uri }
    \prop_gclear:c{ \_stex_morphisms_macro:N \l__stex_modules_uri }
    \prop_gclear:c{ \_stex_symbol_macro:N \l__stex_modules_uri }
    \prop_gclear:c{ \_stex_notations_macro:N \l__stex_modules_uri }
  }
  \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
  \seq_put_right:No \l_stex_all_modules_seq \l__stex_modules_uri
}
```

(*End of definition for* \stex_module_setup_top_nosig:n*. This function is documented on page 142.*)

sfunction

__stex_modules_load_meta: loads the metatheory:

```
\bool_new:N \l_stex_in_meta_bool

\cs_new_protected:Nn \__stex_modules_load_meta: {
  \exp_after:wN \stex_execute_in_module:n \exp_after:wN {
    \exp_after:wN \__stex_modules_load_meta_i:n \exp_after:wN { \l_stex_metatheory_uri }
  }
}

\cs_new_protected:Nn \__stex_modules_load_meta_i:n {
  \bool_if:NTF \l_stex_in_meta_bool {
```

```
    \stex_activate_module:n {#1}
  }{
    \bool_set_true:N \l_stex_in_meta_bool
    \stex_activate_module:n {#1}
    \bool_set_false:N \l_stex_in_meta_bool
  }
}
```

(*End of definition for* `\__stex_modules_load_meta:`.)

`\__stex_modules_top_sig:n` sets up a new module with a signature:

```
\cs_new_protected:Nn \__stex_modules_top_sig:n {
  \stex_debug:nn{modules}{Signature:\l_stex_key_sig_str}
  \stex_metagroup_new:
  \tl_set:Ne \l__stex_modules_uri { \stex_new_module_uri:n {#1} }
  \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
  \stex_if_module_exists:NTF \l__stex_modules_uri {
    \stex_debug:nn{modules}{(already exists)}
    \stex_activate_module:N \l__stex_modules_uri
  }{
    \stex_debug:nn{modules}{(needs loading)}
    \__stex_modules_load_sig:
  }
  \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
}

\cs_new_protected:Nn \__stex_modules_load_sig: {
  \stex_file_split_off_lang:NN \l__stex_modules_sigfile \g_stex_current_file
  \tl_set:Ne \l__stex_modules_sig {
    \exp_args:NNo \stex_document_uri_with_language:Nn
      \l_stex_current_document_uri \l_stex_key_sig_str
  }
  \stex_debug:nn{modules}{Loading~signature~\stex_use_document_uri:N \l__stex_modules_sig;~
  \exp_args:NNe \stex_file_in_smsmode:Nn \l__stex_modules_sig {
    \stex_file_use:N \l__stex_modules_sigfile . \l_stex_key_sig_str . tex
  }
  \stex_activate_module:N \l__stex_modules_uri
}
```

(*End of definition for* `\__stex_modules_top_sig:n`.)

`\__stex_modules_nested:n` sets up a new nested module:

```
\cs_new_protected:Nn \__stex_modules_nested:n {
  \stex_metagroup_new:
  \stex_debug:nn{module}{Nested~module~#1~in~\stex_use_module_uri:N \l_stex_current_module_
  \tl_set:Ne \l__stex_modules_uri { \stex_new_module_uri:n{#1} }
  \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
  \exp_args:No \stex_if_module_exists:NTF \l__stex_modules_uri {
    \stex_debug:nn{modules}{(already exists)}
  }{
    \tl_gclear:c{ \_stex_module_code_macro:N \l__stex_modules_uri }
    \prop_gclear:c{ \_stex_morphisms_macro:N \l__stex_modules_uri }
    \prop_gclear:c{ \_stex_symbol_macro:N \l__stex_modules_uri }
    \prop_gclear:c{ \_stex_notations_macro:N \l__stex_modules_uri }
  }
```

```
  \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
  \seq_put_left:No \l_stex_all_modules_seq \l__stex_modules_uri
}
```

(*End of definition for* `\__stex_modules_nested:n`.)

`\stex_close_module:`

```
\cs_new_protected:Nn \stex_close_module: {
  \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
  \stex_debug:nn{module}{
    Closing~module~\stex_use_module_uri:N \l_stex_current_module_uri^^J
    Code:~\cs_meaning:c{ \_stex_module_code_macro:N \l_stex_current_module_uri }^^J
    Imports:~ \exp_after:wN \prop_to_keyval:N \cs:w \_stex_morphisms_macro:N \l_stex_curren
    Declarations:~ \exp_after:wN \prop_to_keyval:N \cs:w \_stex_symbol_macro:N \l_stex_curr
    Notations:~ \exp_after:wN \prop_to_keyval:N \cs:w \_stex_notations_macro:N \l_stex_curr
  }
}
```

Persist and restore modules in the `.sms`-file:

```
\cs_new_protected:Nn \__stex_modules_persist_module: {
  \stex_persist:e {
    \__stex_modules_restore_module:nnnn {\l_stex_current_module_uri}{
      \exp_after:wN \prop_to_keyval:N \cs:w
        \_stex_morphisms_macro:N \l_stex_current_module_uri
      \cs_end:
    }{
      \exp_after:wN \prop_to_keyval:N \cs:w
        \_stex_symbol_macro:N \l_stex_current_module_uri
      \cs_end:
    }{
      \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
      \exp_after:wN \exp_after:wN \exp_after:wN
      { \cs:w \_stex_module_code_macro:N \l_stex_current_module_uri \cs_end: }
    }{}
    \prop_map_function:cN{\_stex_notations_macro:N \l_stex_current_module_uri}
      \__stex_modules_persist_nots_i:nn
    \exp_not:N \STEXRestoreNotsEnd {}
  }
}

\cs_new:Nn \__stex_modules_persist_nots_i:nn {
  \exp_not:n{#2}
}


\cs_new:Nn \__stex_modules_stringify_uri:nnn {
  {\tl_to_str:n{#1}}
  {\tl_to_str:n{#2}}
  {\tl_to_str:n{#3}}
}
\cs_new:Nn \__stex_modules_stringify_uri:nnnn {
  {\tl_to_str:n{#1}}
  {\tl_to_str:n{#2}}
  {\tl_to_str:n{#3}}
  {\tl_to_str:n{#4}}
```

```
}

\cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
  \tl_set:Ne \l__stex_modules_uri { \__stex_modules_stringify_uri:nnn #1 }
  \stex_debug:nn{persist}{restoring~\stex_use_module_uri:N \l__stex_modules_uri}
  \prop_gset_from_keyval:cn{ \_stex_morphisms_macro:N \l__stex_modules_uri }{#2}
  \prop_gset_from_keyval:cn{\_stex_symbol_macro:N \l__stex_modules_uri}{#3}
  \prop_map_inline:cn{\_stex_symbol_macro:N \l__stex_modules_uri}{
    \stex_ref_new_symbol:n{#1?##1}
  }
  \def \l_tmpa_tl { #4 }
  \exp_args:Nno \tl_gset:cn{\_stex_module_code_macro:N \l__stex_modules_uri}\l_tmpa_tl
  \prop_gclear:c{\_stex_notations_macro:N \l__stex_modules_uri}
  \group_begin:
    \catcode`_=8\relax
    \catcode`:=12\relax
    \__stex_modules_restore_nots:n
}

\quark_new:N \STEXRestoreNotsEnd

\cs_new_protected:Nn \__stex_modules_restore_nots:n {
  \__stex_modules_restore_nots_i:n
}

\cs_new_protected:Nn \__stex_modules_restore_nots_i:n {
  \tl_if_eq:nnTF{#1}{\STEXRestoreNotsEnd}{
    \group_end:
  }{
    \__stex_modules_restore_nots_ii:nnnnn {#1}
  }
}

\cs_new_protected:Nn \__stex_modules_restore_nots_ii:nnnnn {
  \tl_set:Nn \l__stex_modules_tl {{#4}{#5}}
  % eliminate ## => #
  \exp_after:wN \def \exp_after:wN \l__stex_modules_tl \exp_after:wN {\l__stex_modules_tl}
  \exp_args:NNe\use:nn\prop_gput:cnn{
    {\_stex_notations_macro:N \l__stex_modules_uri}
    {\stex_use_symbol_uri:n{#1}\tl_to_str:n{?#2}}{
      {\__stex_modules_stringify_uri:nnnn #1}{\tl_to_str:n{#2}}{#3}
      \exp_args:No \exp_not:n \l__stex_modules_tl
    }
  }
  \__stex_modules_restore_nots_i:n
}
```

(*End of definition for* `\stex_close_module:`. *This function is documented on page 142.*)

sfunction

`\stex_every_module:n`

```
\tl_new:N \g_stex_every_module_tl
\cs_new_protected:Nn \stex_every_module:n {
  \tl_gput_right:Nn \g_stex_every_module_tl { #1 }
}
```

(*End of definition for* `\stex_every_module:n`. *This function is documented on page 142.*)

sfunction

`\stex_do_up_to_module:n`
`\stex_do_up_to_module:e`

```
\cs_new_protected:Nn \stex_do_up_to_module:n {
  \stex_metagroup_do_in:n {#1}
}
\cs_generate_variant:Nn \stex_do_up_to_module:n {e}
```

(*End of definition for* `\stex_do_up_to_module:n`. *This function is documented on page 142.*)

sfunction

`\stex_execute_in_module:n`
`\stex_execute_in_module:e`

```
\cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
  \tl_gput_right:cn { \_stex_module_code_macro:N \l_stex_current_module_uri } { #1 }
  \stex_debug:nn{modules}{extending~activation~for~\stex_use_module_uri:N \l_stex_current_m
  \stex_do_up_to_module:n { #1 }
}{ #1 }}
\cs_generate_variant:Nn \stex_execute_in_module:n {e}
```

(*End of definition for* `\stex_execute_in_module:n`. *This function is documented on page 142.*)

sfunction

`\stex_if_in_module_p:`
`\stex_if_in_module:`*TF*

```
\prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
  \tl_if_exist:NTF \l_stex_current_module_uri
    \prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_if_in_module:TF`. *This function is documented on page 143.*)

sfunction

`\stex_if_module_exists_p:N`
`\stex_if_module_exists:N`*TF*
`\stex_if_module_exists_p:n`
`\stex_if_module_exists:n`*TF*

```
\prg_new_conditional:Nnn \stex_if_module_exists:N {p, T, F, TF} {
  \tl_if_exist:cTF { \_stex_module_code_macro:N #1 }
    \prg_return_true: \prg_return_false:
}
\prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
  \tl_if_exist:cTF { \_stex_module_code_macro:n {#1} }
    \prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_if_module_exists:NTF` *and* `\stex_if_module_exists:nTF`. *These functions are documented on page 143.*)

sfunction

`\stex_activate_module:N`
`\stex_activate_module:n`

```
\cs_new_protected:Nn \stex_activate_module:N {
  \exp_args:No \stex_activate_module:n #1
}

\cs_new_protected:Nn \stex_activate_module:n {
  \seq_if_in:NnF \l_stex_all_modules_seq {#1} {
    \stex_debug:nn{modules}{Activating~module~\stex_use_module_uri:n{#1}}
```

```
    \seq_put_right:Nn \l_stex_all_modules_seq {#1}
    \stex_pseudogroup:nn{
      \tl_set:Nn \l_stex_current_module_uri {#1}
      \tl_if_exist:cF { \_stex_module_code_macro:N \l_stex_current_module_uri }{
        \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:n {#1} }
      }
      \_stex_activate_symbols:
      \_stex_activate_notations:
      \stex_debug:nn{modules}{Executing:~\expandafter\meaning\csname\_stex_module_code_macr
      \use:c{ \_stex_module_code_macro:N \l_stex_current_module_uri }
    }{
      \stex_pseudogroup_restore:N \l_stex_current_module_uri
    }
    \stex_debug:nn{modules}{Activated~module~\stex_use_module_uri:n {#1} }
  }
}
```

(*End of definition for* `\stex_activate_module:N` *and* `\stex_activate_module:n`*. These functions are documented on page 143.*)

sfunction

`\setmetatheory`

```
\NewDocumentCommand \setmetatheory {O{} m}{
  \stex_debug:nn{metatheory}{Setting~metatheory~[#1]#2}
  \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
  \stex_debug:nn{metatheory}{Here:\stex_use_module_uri:N \l_stex_import_uri
  }
  \tl_set_eq:NN \l_stex_metatheory_uri \l_stex_import_uri
  \begingroup
    \stex_require_module:N \l_stex_metatheory_uri
  \endgroup
  \stex_smsmode_do:
}
```

(*End of definition for* `\setmetatheory`*. This function is documented on page 143.*)

sfunction

`\STEXexport`

```
\cs_new_protected:Npn \STEXexport {
  \ExplSyntaxOn
  \__stex_modules_export:n
}
\cs_new_protected:Nn \__stex_modules_export:n {
  \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
  \tl_gput_right:cn { \_stex_module_code_macro:N \l_stex_current_module_uri } {
    \stex_ignore_spaces_and_pars: #1
  }
  \stex_smsmode_do:
}

\stex_deactivate_macro:Nn \STEXexport {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \STEXexport}
```

(*End of definition for* `\STEXexport`*. This function is documented on page 143.*)

sfunction

\stex_structural_feature_module:nn
\stex_structural_feature_module_end:

```
\cs_new_protected:Nn \stex_structural_feature_module:nn {
  \stex_if_do_html:TF {
    \exp_args:Nne \begin{stex_annotate_env} {
      data-ftml-feature-#2={#1}
      \str_if_empty:NF \l_stex_macroname_str {,
        data-ftml-macroname={\l_stex_macroname_str}
      }
    }
    \stex_annotate_invisible:n{}
  }\group_begin:
  \stex_module_setup:n {#1}
}

\cs_new_protected:Nn \stex_structural_feature_module_end: {
  \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
  \stex_close_module:
  \stex_if_do_html:TF{
    \end{stex_annotate_env}
  }\group_end:
}
```

(*End of definition for* `\stex_structural_feature_module:nn` *and* `\stex_structural_feature_module_end:`. *These functions are documented on page 143.*)

sfunction

## 31.1 SMS Mode

⟨@@=stex_smsmode⟩

\stex_sms_allow:N

```
\tl_new:N \g__stex_smsmode_allowed_tl

\cs_new_protected:Nn \stex_sms_allow:N {
  \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
}
```

(*End of definition for* `\stex_sms_allow:N`. *This function is documented on page 144.*)

sfunction

\stex_sms_allow_escape:N

```
\tl_new:N \g__stex_smsmode_allowed_escape_tl

\cs_new_protected:Nn \stex_sms_allow_escape:N {
  \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
}
```

(*End of definition for* `\stex_sms_allow_escape:N`. *This function is documented on page 144.*)

sfunction

\stex_sms_allow_env:n

```
\seq_new:N \g__stex_smsmode_allowedenvs_seq

\cs_new_protected:Nn \stex_sms_allow_env:n {
```

```
  \seq_gput_right:Ne \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
}
```

(*End of definition for* `\stex_sms_allow_env:n`. *This function is documented on page 144.*)

sfunction

Some initial allowed macros:

```
\stex_sms_allow:N \makeatletter
\stex_sms_allow:N \makeatother
\stex_sms_allow:N \ExplSyntaxOn
\stex_sms_allow:N \ExplSyntaxOff
\stex_sms_allow:N \rustexBREAK
\stex_sms_allow_env:n{smodule}
\stex_sms_allow_escape:N \setmetatheory
\stex_sms_allow_escape:N \STEXexport
```

`\stex_sms_allow_import:Nn`

```
\tl_new:N \g__stex_smsmode_allowed_import_tl
\tl_new:N \g__stex_smsmode_import_setup_tl

\cs_new_protected:Nn \stex_sms_allow_import:Nn {
  \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
  \tl_gput_right:Nn \g__stex_smsmode_import_setup_tl {#2}
}
```

(*End of definition for* `\stex_sms_allow_import:Nn`. *This function is documented on page 144.*)

sfunction

`\stex_sms_allow_import_env:nn`

```
\seq_new:N \g__stex_smsmode_allowed_import_env_seq

\cs_new_protected:Nn \stex_sms_allow_import_env:nn {
  \seq_gput_right:Ne \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}}
  \tl_gput_right:Nn \g__stex_smsmode_import_setup_tl {#2}
}

%\stex_sms_allow_import_env:nn{smodule}{}
```

(*End of definition for* `\stex_sms_allow_import_env:nn`. *This function is documented on page 144.*)

sfunction

`\g_stex_sms_import_code`

```
\tl_new:N \g_stex_sms_import_code
```

(*End of definition for* `\g_stex_sms_import_code`. *This variable is documented on page 144.*)

`\stex_if_smsmode_p:`
`\stex_if_smsmode:`*TF*

```
\bool_new:N \g__stex_smsmode_bool

\prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
  \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_if_smsmode:TF`. *This function is documented on page 144.*)

sfunction

\stex_file_in_smsmode:Nn

```
\seq_new:N \l__stex_smsmode_cycles_seq

\cs_new_protected:Nn \stex_file_in_smsmode:Nn {
  \seq_gclear:N \l__stex_smsmode_importmodules_seq
  \seq_gclear:N \l__stex_smsmode_sigmodules_seq
  \tl_clear:N \g_stex_sms_import_code
  \seq_if_in:NnF \l__stex_smsmode_cycles_seq {#2} {
  \group_begin:
    \seq_push:Nn \l__stex_smsmode_cycles_seq {#2}
    \let \l_stex_metatheory_uri \c_stex_default_metatheory
    \seq_clear:N \l_stex_all_modules_seq
    \stex_undefine:N \l_stex_current_module_uri
    \cs_set:Npn \stex_check_term:nn ##1 ##2 {}
    \stex_debug:nn{sms}{Loading~#2~in~\stex_use_document_uri:N #1}
    \stex_with_file_hooks:Nnn #1 {#2}{
      \__stex_smsmode_in_smsmode:n {
        \group_begin:
          \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
          \g__stex_smsmode_import_setup_tl
          \__stex_smsmode_start_smsmode:n{#2}
        \group_end:
        %{
          \g_stex_sms_import_code
        %}
        %\group_begin:
          \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
          \__stex_smsmode_start_smsmode:n{#2}
        %\group_end:
      }
    }
  \group_end:
  }
}

\cs_new_protected:Nn \__stex_smsmode_in_smsmode:n {
  \stex_suppress_html:n {
    \vbox_set:Nn \l_tmpa_box {
      \bool_set_true:N \g__stex_smsmode_bool
      #1
    }
  }
}

\quark_new:N \q__stex_smsmode_break

\cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
  \everyeof{\q__stex_smsmode_break\exp_not:N}
  \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
  \exp_after:wN \exp_after:wN \exp_after:wN
  \stex_smsmode_do:
  \cs:w @ @ input\cs_end: "#1" \relax
}
```

(*End of definition for* `\stex_file_in_smsmode:Nn`. *This function is documented on page 144.*)

sfunction

`\stex_smsmode_do:`

```
\let\stex_smsmode_do:\relax

\cs_new_protected:Nn \__stex_smsmode_smsmode_do: { \__stex_smsmode_do:w }

\cs_new_protected:Npn \__stex_smsmode_do:w #1 {
  \exp_args:Ne \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
    %\__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
    \__stex_smsmode_check_cs:N {#1}
  }{
    \__stex_smsmode_do:w #1
  }
}

\cs_new_protected:Nn \__stex_smsmode_check_cs:N {
  %\stex_debug:nn{temp}{Checking \exp_not:N#1 \relax}
  \exp_after:wN\if\exp_after:wN\relax\noexpand #1 ~
  \exp_after:wN \__stex_smsmode_do_aux:N \exp_after:wN#1\else
  \exp_after:wN \__stex_smsmode_do:w \fi
}

%\cs_new_protected:Nn \__stex_smsmode_check_cs:NNn {
%  \message{^^JHERE: \tl_to_str:n{#1~and~#2~and~#3}}
%  \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
%  \exp_after:wN#1\exp_after:wN#3\else
%  \exp_after:wN#2\fi
%}

\cs_new_protected:Nn \__stex_smsmode_do_aux:N {
  \cs_if_eq:NNF #1 \q__stex_smsmode_break {
    \__stex_smsmode_do_aux_curr:N #1
  }
}

\cs_new_protected:Nn \__stex_smsmode_do_aux_normal:N {
  \tl_if_in:NnTF \g__stex_smsmode_allowed_tl {#1} {
    \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
    #1\__stex_smsmode_do:w
  }{
    \tl_if_in:NnTF \g__stex_smsmode_allowed_escape_tl {#1} {
      \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
      #1
    }{
      \cs_if_eq:NNTF \begin #1 {
        \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowedenvs_seq
      }{
        \cs_if_eq:NNTF \end #1 {
          \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowedenvs_seq
        }{
          \__stex_smsmode_do:w
        }
```

```
      }
    }
  }
}

\cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
  \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
    \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
    #1
  }{
    \cs_if_eq:NNTF \begin #1 {
      \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowed_import_env_seq
    }{
      \cs_if_eq:NNTF \end #1 {
        \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowed_import_env_seq
      }{
        \__stex_smsmode_do:w
      }
    }
  }
}

\cs_new_protected:Nn \__stex_smsmode_check_begin:Nn {
  \seq_if_in:NeTF #1 { \tl_to_str:n{#2} }{
    \stex_debug:nn{sms}{Environment~#2}
    \begin{#2}
  }{
    \__stex_smsmode_do:w
  }
}
\cs_new_protected:Nn \__stex_smsmode_check_end:Nn {
  \seq_if_in:NeTF #1 { \tl_to_str:n{#2} }{
    \stex_debug:nn{sms}{End~Environment~#2}
    \end{#2}\__stex_smsmode_do:w
  }{
    \__stex_smsmode_do:w
  }
}
```

(*End of definition for* `\stex_smsmode_do:`. *This function is documented on page 144.*)

sfunction

## 31.2 Inheritance and Morphisms

```
⟨@@=stex_import⟩
```

`\stex_require_module:N`
`\stex_require_module_noerr:N`
`\stex_require_module_noerr:n`

```
\cs_new_protected:Nn \stex_require_module:N {
  \stex_if_module_exists:NF #1 {
    \__stex_import_load_module:NF #1 {
      \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:N #1}
    }
  }
  \stex_activate_module:N #1
```

```
}

\cs_new_protected:Nn \stex_require_module_noerr:N {
  \stex_if_module_exists:NF #1 {
    \__stex_import_load_module:NF #1 {}
    \str_if_empty:NF \l__stex_import_file_str {
      \stex_activate_module:N #1
    }
  }
}

\cs_new_protected:Nn \stex_require_module_noerr:n {
  \tl_set:Nn \l__stex_import_uri { #1 }
  \stex_require_module_noerr:N \l__stex_import_uri
}

\cs_new_protected:Npn \__stex_import_load_module:NF #1 #2 {
    \tl_set_eq:NN \l__stex_import_uri #1
    \tl_set:Ne \l__stex_import_archive_str { \stex_module_uri_archive:N #1 }
    \tl_set:Ne \l__stex_import_path_str { \stex_module_uri_path:N #1 }
    \tl_set:Ne \l__stex_import_name_str { \stex_module_uri_name:N #1 }
    \str_if_eq:NNTF \l__stex_import_archive_str \c_stex_no_archive_str {
      \str_set_eq:NN \l__stex_import_pre_str \l__stex_import_path_str
      \__stex_import_load_check:n {#2}
    }{
      \exp_args:No \stex_in_archive:nn \l__stex_import_archive_str {
        \str_set:Ne \l__stex_import_pre_str {
          \exp_args:Ne \stex_source_path:n{ \l__stex_import_path_str }
        }
        \exp_args:No \__stex_import_load_check:n {#2}
      }
    }
}

\cs_new_protected:Nn \__stex_import_load_check:n {
  \str_clear:N \l__stex_import_file_str
  \str_set_eq:NN \l__stex_import_language_str \l_stex_current_language_str
  \__stex_import_check_file:nnn{ / \l__stex_import_name_str .tex }{}{
    \__stex_import_check_file:nnn{/ \l__stex_import_name_str . \l_stex_current_language_str
      \__stex_import_check_file:nnn{/ \l__stex_import_name_str .en.tex}{
        \str_set:Nn \l__stex_import_language_str {en}
      }{
        \__stex_import_load_check_i:n{#1}
      }
    }
  }
  \__stex_import_load_check_ii:
}

\cs_new_protected:Nn \__stex_import_load_check_i:n {
  \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_import_path_str
  \seq_pop_right:NN \l_tmpa_seq \l__stex_import_name_str
  \str_set:Ne \l__stex_import_path_str { \seq_use:Nn \l_tmpa_seq /}
  \__stex_import_check_file:nnn{.tex}{}{
```

```
    \__stex_import_check_file:nnn{. \l_stex_current_language_str .tex}{}{
      \__stex_import_check_file:nnn{.en.tex}{
        \str_set:Nn \l__stex_import_language_str {en}
      }{
        #1
      }
    }
  }
}

\cs_new_protected:Nn \__stex_import_load_check_ii: {
  \str_if_empty:NF \l__stex_import_file_str {
    \stex_if_smsmode:TF{
      \exp_args:NNo \exp_args:Nne \str_if_eq:nnTF{\l__stex_import_file_str}{\stex_file_use:
        \stex_debug:nn{imports}{Skipping~current~file}
      }{
        \IfFileExists{ \l__stex_import_file_str }{
          \__stex_import_load_file:
        }{}
      }
    }{
      \IfFileExists{ \l__stex_import_file_str }{
        \__stex_import_load_file:
      }{}
    }
  }
}

\cs_new_protected:Npn \__stex_import_check_file:nnn #1 #2 {
  \stex_debug:nn{imports}{Checking~ \l__stex_import_pre_str #1}
  \IfFileExists{ \l__stex_import_pre_str #1 }{
    \stex_debug:nn{imports}{Success}
    \str_set:Ne \l__stex_import_file_str { \l__stex_import_pre_str #1 }
    #2
  }
}

\cs_new_protected:Nn \__stex_import_load_file: {
  \tl_set:Ne \l__stex_import_doc_uri {
    { \l__stex_import_archive_str }
    { \l__stex_import_path_str }
    { \l__stex_import_name_str }
    { \l__stex_import_language_str }
    {}
  }
  \exp_args:NNo \stex_file_in_smsmode:Nn \l__stex_import_doc_uri \l__stex_import_file_str
  \stex_if_module_exists:NF \l__stex_import_uri {
    \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:N \l__stex_import_uri}
  }
}
```

(*End of definition for* `\stex_require_module:N`, `\stex_require_module_noerr:N`, *and* `\stex_require_module_noerr:n`. *These functions are documented on page 145.*)

sfunction

\usemodule

```
\stex_new_stylable_cmd:nnnn {usemodule} { O{} m } {
  \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
  \stex_require_module:N \l_stex_import_uri
  \__stex_import_usemodule:
}{}%{\aftergroup\ignorespaces}

\cs_new_protected:Nn \__stex_import_usemodule: {
  \stex_debug:nn{usemodule}{Done.}
  \stex_if_do_html:T {
    \hbox{\stex_annotate_invisible:nn
      {data-ftml-usemodule=\stex_use_module_uri:N \l_stex_import_uri } {}}
  }
  \stex_if_smsmode:F{
    \group_begin:
    \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_import_uri }
    \str_set:Ne \thismodulename {\stex_module_uri_name:N \l_stex_import_uri}
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
  }
}
\stex_deactivate_macro:Nn \usemodule {document~environment}
\AtBeginDocument{\stex_reactivate_macro:N \usemodule}
```

(*End of definition for* `\usemodule`. *This function is documented on page 145.*)

sfunction

\importmodule

```
\stex_new_stylable_cmd:nnnn{importmodule} { O{} m } {
  \__stex_import_import_module:nn {#1}{#2}
  \stex_smsmode_do:
}{\aftergroup\ignorespaces}

\stex_deactivate_macro:Nn \importmodule {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \importmodule}
\stex_sms_allow_escape:N \importmodule

\cs_new_protected:Nn \__stex_import_import_module:nn {
  \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
  \stex_require_module:N \l_stex_import_uri
  \__stex_import_import_module_i:n {#1}
}

\cs_new_protected:Nn \__stex_import_import_module_i:n {
  \stex_execute_in_module:e{
    \stex_activate_module:n { \l_stex_import_uri }
  }
  \stex_add_morphism:nonn
    {}{\l_stex_import_uri}{import}{}
  \stex_if_do_html:T {
    \stex_annotate_invisible:nn
      {data-ftml-import=\stex_use_module_uri:N \l_stex_import_uri } {}
  }
```

```
  \stex_if_smsmode:F{
    \group_begin:
    \tl_set:Nn \thisarchive {#1}
    \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_import_uri }
    \str_set:Ne \thismodulename {\stex_module_uri_name:N \l_stex_import_uri}
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
  }
}
```

In sms mode, we change the definition:

```
\cs_new_protected:Nn \__stex_import_import_module_presms:nn {
  \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
  \bool_if:NF \l_stex_relative_import_bool {
    \tl_gput_right:Ne \g_stex_sms_import_code {
      \stex_require_module_noerr:n { \l_stex_import_uri }
    }
  }
}

\stex_sms_allow_import:Nn \importmodule {
  \stex_reactivate_macro:N \importmodule
  \let \__stex_import_import_module:nn \__stex_import_import_module_presms:nn
}
```

(*End of definition for* `\importmodule`. *This function is documented on page 145.*)

sfunction

`\requiremodule`

```
\stex_new_stylable_cmd:nnnn {requiremodule} { O{} m } {
  \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
  \stex_require_module:N \l_stex_import_uri
  \__stex_import_requiremodule:
}{}%{\aftergroup\ignorespaces}
\stex_deactivate_macro:Nn \requiremodule {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \requiremodule}

\cs_new_protected:Nn \__stex_import_requiremodule: {
  \stex_debug:nn{requiremodule}{Done.}
  \stex_if_do_html:T {
    \hbox{\stex_annotate_invisible:nn
      {data-ftml-import=\stex_use_module_uri:N \l_stex_import_uri } {}}
  }
  \stex_if_smsmode:F{
    \group_begin:
    \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_import_uri }
    \str_set:Ne \thismodulename {\stex_module_uri_name:N \l_stex_import_uri}
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
  }
}
```

(*End of definition for* `\requiremodule`. *This function is documented on page 145.*)

sfunction

`\stex_add_morphism:nnnn`

```
\cs_new_protected:Nn \stex_add_morphism:nnnn {
  \exp_args:Nne \prop_gput:cnn{ \_stex_morphisms_macro:N \l_stex_current_module_uri }{
    \tl_if_empty:nTF{#1}{[\stex_use_module_uri:n {#2}]}{#1}
  }{{#1}{#2}{#3}{#4}}
}
\cs_generate_variant:Nn \stex_add_morphism:nnnn {nonn,oooe}
```

(*End of definition for* `\stex_add_morphism:nnnn`. *This function is documented on page 145.*)

sfunction

## 31.3 Theory Morphisms

⟨@@=stex_morphisms⟩

`\stex_structural_feature_morphism:nnnnn`
`ructural_feature_morphism_with_macros:nnnnn`
`\stex_structural_feature_morphism_end:`

```
\tl_new:N \l_stex_current_domain_uri
\bool_new:N \l__stex_morphisms_with_macros_bool

\cs_new_protected:Npn \stex_structural_feature_morphism_with_macros:nnnnn {
  \bool_set_true:N \l__stex_morphisms_with_macros_bool
  \__stex_morphisms_morphism:nnnnn
}
\cs_new_protected:Npn \stex_structural_feature_morphism:nnnnn {
  \bool_set_false:N \l__stex_morphisms_with_macros_bool
  \__stex_morphisms_morphism:nnnnn
}

\cs_new_protected:Nn \__stex_morphisms_morphism:nnnnn {
  \tl_clear:N \l_stex_current_domain_uri
  \stex_debug:nn{morphisms}{new~morphism:{#1}{#2}{#3}{#4}{#5}}
  \tl_if_empty:nT{#3}{
    \stex_get_mathstructure:n{#4}
    \tl_set_eq:NN \l_stex_current_domain_uri \l_stex_get_structure_module_uri
  }
  \tl_if_empty:NT \l_stex_current_domain_uri {
    \stex_uri_from_pair:Nnn \l_stex_import_uri { #3 }{ #4 }
    \group_begin:
    \stex_require_module:N \l_stex_import_uri
    \exp_args:Nne \use:nn \group_end: {
      \tl_set:Nn \exp_not:N\l_stex_current_domain_uri {\l_stex_import_uri}
    }
  }
  \tl_if_empty:nT{#1}{
    \msg_error:nn{stex}{error/morphism-needs-name}
  }
  \str_set:Nn \l_tmpa_str {#1}

  \stex_debug:nn{morphisms}{#2:~\l_tmpa_str;~for~\stex_use_module_uri:N \l_stex_current_dom
```

```
  \str_set:Nn \l__stex_morphisms_feature_str {#2}
  \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str

  \stex_if_do_html:TF {
    \begin{stex_annotate_env} {
      data-ftml-feature-#2={\l_tmpa_str},
      data-ftml-domain={\stex_use_module_uri:N \l_stex_current_domain_uri}
      #5
    }
    \stex_annotate_invisible:n{}
  }\group_begin:
  \__stex_morphisms_setup:
  \__stex_morphisms_reactivate:
  \stex_metagroup_new:
}

\cs_new_protected:Nn \__stex_morphisms_setup: {
  \seq_clear:N \l_stex_morphism_symbols_seq
  \seq_clear:N \l_stex_morphism_renames_seq
  \seq_clear:N \l_stex_morphism_morphisms_seq
  \seq_clear:N \l_stex_morphism_assigns_seq
  \__stex_morphisms_do_decls:
  \exp_args:No \__stex_morphisms_do:n \l_stex_current_domain_uri
}

\cs_new_protected:Nn \__stex_morphisms_do_decls: {
  \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_uri {
    \exp_args:NNe \seq_put_right:Nn \l_stex_morphism_symbols_seq {
      { \stex_new_symbol_uri:nn{##1}{##3} }
      { {##2}{##4}{##5}{##6}{##7}{##8}##9 }
    }
  }
}


\cs_new_protected:Nn \__stex_morphisms_do:n {
  %\seq_clear:N \l__stex_morphisms_dones_seq
    \prop_map_inline:cn {\_stex_morphisms_macro:n{#1}}{
      %\seq_if_in:NnF \l__stex_morphisms_dones_seq {##2}{
      %  \seq_push:Nn \l__stex_morphisms_dones_seq {##2}
        \stex_debug:nn{morphisms}{Doing ~ \tl_to_str:n{##2}}
        \__stex_morphisms_do_morph:nnnn ##2
     % }
    }
}

\cs_new_protected:Nn \__stex_morphisms_do_morph:nnnn {
    \tl_if_empty:nF{#3}{
      \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
    }
    \__stex_morphisms_do:n{#2}
}
```

```
\cs_new_protected:Nn \stex_structural_feature_morphism_end: {
  \tl_gset_eq:NN \l_stex_current_domain_uri \l_stex_current_domain_uri
  \seq_gset_eq:NN \l_stex_morphism_symbols_seq \l_stex_morphism_symbols_seq
  \seq_gset_eq:NN \l_stex_morphism_renames_seq \l_stex_morphism_renames_seq
  \seq_gset_eq:NN \l_stex_morphism_assigns_seq \l_stex_morphism_assigns_seq
  \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
  \bool_gset_eq:NN \l__stex_morphisms_with_macros_bool \l__stex_morphisms_with_macros_bool
  \stex_if_do_html:TF{
    \end{stex_annotate_env}
  }\group_end:
  \__stex_morphisms_do_elaboration:
}

\cs_new_protected:Nn \__stex_morphisms_do_elaboration: {
  \stex_debug:nn{morphisms}{
    Elaborating:^^J\meaning \l_stex_morphism_symbols_seq
    ^^J^^J
    Renamings:^^J
    \meaning \l_stex_morphism_renames_seq
    ^^J^^J
    Assignments:^^J
    \meaning \l_stex_morphism_assigns_seq
  }

  \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
    \__stex_morphisms_elab:nn ##1
  }

  \stex_add_morphism:oooe
    \l_stex_feature_name_str
    \l_stex_current_domain_uri
    \l__stex_morphisms_feature_str
    {\seq_map_function:NN \l_stex_morphism_renames_seq \__stex_morphisms_rename:n}
}

\cs_new:Nn \__stex_morphisms_rename:n{
  \__stex_morphisms_rename:nn #1
}

\cs_new:Nn \__stex_morphisms_rename:nn{
  {#1}{\use_ii:nn#2}
}

\cs_new_protected:Nn \__stex_morphisms_elab:nn {
  \tl_clear:N \l__stex_morphisms_tmp
  \seq_map_inline:Nn \l_stex_morphism_renames_seq {
    \__stex_morphisms_elab_check_rename:nnn{#1}##1
  }
  \tl_if_empty:NTF \l__stex_morphisms_tmp {
    \exp_args:Ne \__stex_morphisms_elab_i:nnn {\stex_symbol_uri_name:n {#1}}{#1}
  }{
    \stex_debug:nn{morphisms}{Generating~1~\l__stex_morphisms_tmp}
    \use_i:nn{ \exp_args:Nno \use:nn {\__stex_morphisms_add_symbol:nnnnnnnnN {#1}} \l__stex
  }#2
```

```
3174
3175   \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_uri {
3176     \str_if_eq:nnT{#1}{##1}{
3177       \exp_args:Ne \stex_add_notation:nnnnn {
3178         \exp_args:Ne \stex_new_symbol_uri:n {\exp_after:wN \use_ii:nn \l__stex_morphisms_tm
3179       }{##2}{##3}{##4}{##5}
3180     }
3181   }
3182 }
3183
3184 \cs_new_protected:Nn \__stex_morphisms_elab_i:nnn {
3185   \tl_set:Ne \l__stex_morphisms_tmp {{}{\l_stex_feature_name_str / #1}}
3186   \stex_debug:nn{morphisms}{Generating~2~\l_stex_feature_name_str / #1}
3187   \bool_if:NTF \l__stex_morphisms_with_macros_bool {
3188     \exp_args:Nnno \__stex_morphisms_add_symbol:nnnnnnnnN {#2} {#3}
3189   }{
3190     \exp_args:Nnno \__stex_morphisms_add_symbol:nnnnnnnnN {#2} {}
3191   }
3192   {\l_stex_feature_name_str / #1}
3193 }
3194
3195 \cs_new_protected:Npn \__stex_morphisms_add_symbol:nnnnnnnnN #1 {
3196   \tl_clear:N \l__stex_morphisms_tmp_b
3197   \seq_map_inline:Nn \l_stex_morphism_assigns_seq {
3198     \__stex_morphisms_elab_check_assign:nnnnn{#1}##1
3199   }
3200   \tl_if_empty:NTF \l__stex_morphisms_tmp_b
3201   \stex_add_symbol:nnnnnnnN
3202   \__stex_morphisms_add_symbol_i:nnnnnnnN
3203 }
3204
3205 \cs_new_protected:Npn \__stex_morphisms_add_symbol_i:nnnnnnnN #1 #2 #3 #4 #5 {
3206   \stex_add_symbol:nnnnnnnN {#1}{#2}{#3}{#4}{assigned}
3207 }
3208
3209 \cs_new_protected:Nn \__stex_morphisms_elab_check_assign:nnnnn {
3210   \tl_if_eq:nnT{#1}{{#2}{#3}{#4}{#5}}{
3211     \seq_map_break:n{
3212       \tl_set:Nn \l__stex_morphisms_tmp_b {assigned}
3213     }
3214   }
3215 }
3216
3217 \cs_new_protected:Nn \__stex_morphisms_elab_check_rename:nnn {
3218   \tl_if_eq:nnT{#1}{#2}{
3219     \seq_map_break:n{
3220       \tl_set:Nn \l__stex_morphisms_tmp {#3}
3221     }
3222   }
3223 }
3224
3225 \cs_new_protected:Nn \__stex_morphisms_do_for_list: {
3226   \seq_clear:N \l_stex_fors_seq
3227   \clist_map_inline:Nn \l_stex_key_for_clist {
```

```
    \exp_args:Ne\stex_get_in_morphism:n{\tl_to_str:n{##1}}
    \seq_put_right:No \l_stex_fors_seq
      {\l_stex_get_symbol_uri}
  }
}


\cs_new_protected:Nn \__stex_morphisms_definiens_impl:nn {
  \tl_if_empty:nF {#1} {
    \stex_get_in_morphism:n { #1 }
    \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
  }
  \tl_if_empty:NT \l_stex_current_def_uri {
    \msg_error:nn{stex}{error/definiensfor}
  }
  \stex_debug:nn{definiens}{Checking~\stex_use_symbol_uri:N \l_stex_current_def_uri }
  \stex_if_do_html:TF{
    \exp_after:wN \stex_metagroup_do_in:n \exp_after:wN {
      \exp_after:wN \seq_put_right:Nn \exp_after:wN  \l_stex_morphism_assigns_seq
      \exp_after:wN { \l_stex_current_def_uri }
    }
    \mode_leave_vertical: \stex_annotate:nn{data-ftml-assign={\stex_use_symbol_uri:N  \l_st
      \_stex_annotate_force_break:n{
        \stex_annotate:nn{data-ftml-definiens={}}{#2}
      }
    }
  }{
    \exp_args:No \__stex_morphisms_add_definiens:nn \l_stex_current_def_uri {#2}
    \stex_if_smsmode:F{#2}
  }
}

\cs_new_protected:Nn \__stex_morphisms_add_definiens:nn {
  \tl_set:Nn \l_stex_get_symbol_uri {#1}
  \stex_debug:nn{morphisms}{Adding~definiens~\tl_to_str:n{#1~#2}}
  %\__stex_morphisms_set_definiens_macros: #1\__stex_morphisms_break:
  \_stex_assign_do:n{#2}
}

%\cs_new_protected:Npn \__stex_morphisms_set_definiens_macros: #1?#2?#3\__stex_morphisms_br
%  \str_set:Nn \l_stex_get_symbol_uri { #1?#2} % TODO
%  \str_set:Nn \l_stex_get_symbol_name_str {#3}
%  \exp_args:Nne\use:nn{\__stex_morphisms_set_definiens_macros_i:nnnnnnn}{
%    \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}
%  }
%}

%\cs_new_protected:Nn \__stex_morphisms_set_definiens_macros_i:nnnnnnn {
%  \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
%}


\cs_new_protected:Nn \__stex_morphisms_rename_all: {
    % TODO
```

```
}


\cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
  \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
    \__stex_morphisms_total_check:nn ##1
  }
}

\cs_new_protected:Nn \__stex_morphisms_total_check:nn {
  \__stex_morphisms_total_check:nnnnnnnN {#1} #2
}

\cs_new_protected:Nn \__stex_morphisms_total_check:nnnnnnnN {
  \tl_if_empty:nT{#5}{
    \seq_if_in:NnF \l_stex_morphism_assigns_seq {#1} {
      \msg_error:nnxx{stex}{error/needsdefiniens}{\stex_use_symbol_uri:n {#1}}{total~morphi
    }
  }
}

\cs_new:Npn \__stex_morphisms_split_qm:w #1 ? #2 ? #3 { #3 }



\cs_new_protected:Npn \__stex_morphisms_reactivate: {
  \stex_deactivate_macro:Nn \symdecl {module~environments}
  \stex_deactivate_macro:Nn \textsymdecl {module~environments}
  \stex_deactivate_macro:Nn \symdef {module~environments}
  \stex_deactivate_macro:Nn \notation {module~environments}
  \stex_deactivate_macro:Nn \importmodule {module~environments}
  \stex_deactivate_macro:Nn \requiremodule {module~environments}
  \stex_deactivate_macro:Nn \smodule {outside~of~morphisms}
  \stex_reactivate_macro:N \assign
  \stex_reactivate_macro:N \assignMorphism
  \stex_reactivate_macro:N \renamedecl
  \cs_set_eq:NN \_stex_do_for_list: \__stex_morphisms_do_for_list:
  \cs_set_eq:NN \_stex_add_definiens:nn \__stex_morphisms_add_definiens:nn
  \cs_set_eq:NN \_stex_definiens_impl:nn \__stex_morphisms_definiens_impl:nn
}
```

(*End of definition for* `\stex_structural_feature_morphism:nnnnn`, `\stex_structural_feature_morphism_with_macros:nnnnn`, *and* `\stex_structural_feature_morphism_end:`. *These functions are documented on page 145.*)

sfunction

`\stex_iterate_morphisms:nn`

```
\cs_new_protected:Nn \stex_iterate_morphisms:nn {
  \seq_clear:N \l__stex_morphisms_mods_seq
  \bool_set_true:N \l__stex_morphisms_continue_bool
  \cs_set:Npn \__stex_morphisms_cs:nnnn ##1 ##2 ##3 ##4 ##5 {
    #2
    \bool_if:NT \l__stex_morphisms_continue_bool {
      \str_if_eq:nnTF{##1}{[##2]}{
        \tl_put_right:Nn \l__stex_morphisms_todo_tl {
```

```
          \__stex_morphisms_iterate:nn{##5}{##2}
        }
      }{
        \tl_put_right:Nn \l__stex_morphisms_todo_tl {
          \__stex_morphisms_iterate:nn{##5 / ##1}{##2}
        }
      }
    }
  }
  \cs_set:Npn \stex_iterate_break:n ##1 {
    \bool_set_false:N \l__stex_morphisms_continue_bool
    \prop_map_break:n{##1}
  }
  \__stex_morphisms_iterate:nn{}{#1}
}
\cs_new_protected:Nn \__stex_morphisms_iterate:nn {
  \tl_clear:N \l__stex_morphisms_todo_tl
  \seq_if_in:NnF \l__stex_morphisms_mods_seq {#1 #2}{
    \seq_put_right:Nn \l__stex_morphisms_mods_seq {#1 #2}
    \prop_map_inline:cn{\_stex_morphisms_macro:n{#2}}{
      \__stex_morphisms_cs:nnnn ##2 {#1}
      % TODO
      % ##1: name or [mpath]
      % ##2 = {####1}{####2}{####3}{####4}
      % ####1 = name
      % ####2 = mpath
      % ####3 = type
      % ####4 = {origname}{newname}*
    }
    \bool_if:NT \l__stex_morphisms_continue_bool \l__stex_morphisms_todo_tl
  }
}
```

(*End of definition for* `\stex_iterate_morphisms:nn`. *This function is documented on page 145.*)

sfunction

`\stex_get_in_morphism:n`

```
\cs_new_protected:Nn \stex_get_in_morphism:n {
  \stex_debug:nn{morphisms}{Getting~in~morphism:~#1}
  \tl_clear:N \l_stex_get_symbol_uri
  \str_set:Nn \l__stex_morphisms_get_str {#1}
  \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
    \__stex_morphisms_get_check:nn ##1
  }
  \tl_if_empty:NT \l_stex_get_symbol_uri {
    \seq_map_inline:Nn \l_stex_morphism_renames_seq {
      \__stex_morphisms_renamed_check:nn##1
    }
    \tl_if_empty:NT \l_stex_get_symbol_uri {
      \msg_error:nnxx{stex}{error/unknownsymbolin}{#1}{
        morphism~\l_stex_feature_name_str
      }
    }
  }
```

```
}


\cs_new_protected:Nn \__stex_morphisms_get_check:nn {
  \__stex_morphisms_check_name:nnnn #1 #2
}

\cs_new_protected:Nn \__stex_morphisms_check_name:nnnn {
  % TODO module name, path, archive, macroname ?
  \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str {#4} {
    \tl_set:Nn \l_stex_get_symbol_uri {{#1}{#2}{#3}{#4}}
    \__stex_morphisms_set:nnnnnnN
  }{
    \__stex_morphisms_macro:nnnnnnnN{{#1}{#2}{#3}{#4}}
  }
}

\cs_new_protected:Nn \__stex_morphisms_set:nnnnnnN {
  \seq_map_break:n{
    \str_set:Nn \l_stex_get_symbol_macro_str{#1}
    \int_set:Nn \l_stex_get_symbol_arity_int {#2}
    \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
    \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
    \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
    \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
    \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
  }
}

\cs_new_protected:Npn \__stex_morphisms_macro:nnnnnnnN #1 #2 {
  \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str {#2}{
    \tl_set:Nn \l_stex_get_symbol_uri{#1}
    \__stex_morphisms_set:nnnnnnN
  }\__stex_morphisms_gobble:nnnnnnN
  {#2}
}

\cs_new_protected:Nn \__stex_morphisms_gobble:nnnnnnN {}

\cs_new_protected:Nn \__stex_morphisms_renamed_check:nn {
  \stex_debug:nn{here}{\tl_to_str:n{{#1}{#2}}:~\l__stex_morphisms_get_str}
  \__stex_morphisms_renamed_check:nnnnnn #1 #2
}

\cs_new_protected:Nn \__stex_morphisms_renamed_check:nnnnnn {
  \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str{#5}{
    \seq_map_break:n{
      \str_set:Nn \l__stex_morphisms_get_str {#4}
      \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
        \__stex_morphisms_get_check:nn ##1
      }
    }
  }{
    \exp_args:No \str_if_eq:nnT \l__stex_morphisms_get_str{#6}{
```

```
      \seq_map_break:n{
        \str_set:Nn \l__stex_morphisms_get_str {#4}
        \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
          \__stex_morphisms_get_check:nn ##1
        }
      }
    }
  }
}
```

(*End of definition for* `\stex_get_in_morphism:n`. *This function is documented on page 145.*)

sfunction

`copymodule` (*env.*)

```
\stex_new_stylable_env:nnnnnnn {copymodule}{m O{} m}{
  \__stex_morphisms_begin_copy:Nnnn\stex_structural_feature_morphism:nnnnn {#1}{#2}{#3}
}{
  \stex_if_smsmode:F {
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_end:
}{}{}{}

\stex_new_stylable_env:nnnnnnn {copymodule*}{m O{} m}{
  \cs_set:Npn \stex_style_apply: {
    \_stex_apply_patch_begin:n{copymodule}
  }
  \__stex_morphisms_begin_copy:Nnnn\stex_structural_feature_morphism_with_macros:nnnnn {#1}
}{
  \cs_set:Npn \stex_style_apply: {
    \_stex_apply_patch_end:n{copymodule}
  }
  \stex_if_smsmode:F {
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_end:
}{}{}{}

\cs_new_protected:Nn \__stex_morphisms_begin_copy:Nnnn {
  #1{#2}{morphism}{#3}{#4}{,data-ftml-total=false}
  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #2 }
    \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
    \stex_style_apply:
  }
  \stex_smsmode_do:
}


\stex_deactivate_macro:Nn \copymodule {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \copymodule
}
\stex_sms_allow_env:n{copymodule}
```

```
3482
3483 \exp_after:wN \stex_deactivate_macro:Nn \csname copymodule*\endcsname {module~environments}
3484 \stex_every_module:n {
3485   \exp_after:wN\stex_reactivate_macro:N \csname copymodule*\endcsname
3486 }
3487 \stex_sms_allow_env:n{copymodule*}
3488
3489 \stex_new_stylable_cmd:nnnn{copymod}{s m O{} m m}{
3490   \IfBooleanTF#1\stex_structural_feature_morphism_with_macros:nnnnn
3491   \stex_structural_feature_morphism:nnnnn{#2}{morphism}{#3}{#4}{,data-ftml-total={false}}
3492   \clist_map_function:nN{#5}\__stex_morphisms_parse_assign:n
3493   \stex_if_smsmode:F {
3494     \tl_set:Nn \thiscopyname { #2 }
3495     \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3496     \stex_style_apply:
3497   }
3498   \stex_structural_feature_morphism_end:
3499   \stex_smsmode_do:
3500 }{}
3501
3502 \stex_deactivate_macro:Nn \copymod {module~environments}
3503 \stex_every_module:n {
3504   \stex_reactivate_macro:N \copymod
3505 }
3506 \stex_sms_allow_escape:N\copymod
```

`interpretmodule` (*env.*)

```
3507 \stex_new_stylable_env:nnnnnnn {interpretmodule}{m O{} m}{
3508   \__stex_morphisms_begin_interpret:Nnnn\stex_structural_feature_morphism:nnnnn {#1}{#2}{#3
3509 }{
3510   \stex_structural_feature_morphism_check_total:
3511   \stex_if_smsmode:F {
3512     \stex_style_apply:
3513   }
3514   \stex_structural_feature_morphism_end:
3515 }{}{}{}
3516
3517 \stex_new_stylable_env:nnnnnnn {interpretmodule*}{m O{} m}{
3518   \cs_set:Npn \stex_style_apply: {
3519     \_stex_apply_patch_begin:n{interpretmodule}
3520   }
3521   \__stex_morphisms_begin_interpret:Nnnn\stex_structural_feature_morphism_with_macros:nnnnn
3522 }{
3523   \cs_set:Npn \stex_style_apply: {
3524     \_stex_apply_patch_end:n{interpretmodule}
3525   }
3526   \stex_structural_feature_morphism_check_total:
3527   \stex_if_smsmode:F {
3528     \stex_style_apply:
3529   }
3530   \stex_structural_feature_morphism_end:
3531 }{}{}{}
3532
3533 \cs_new_protected:Nn \__stex_morphisms_begin_interpret:Nnnn {
```

```
  #1{#2}{morphism}{#3}{#4}{,data-ftml-total=true}
  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #2 }
    \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
    \stex_style_apply:
  }
  \stex_smsmode_do:
}

\stex_deactivate_macro:Nn \interpretmodule {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \interpretmodule
}
\stex_sms_allow_env:n{interpretmodule}

\exp_after:wN \stex_deactivate_macro:Nn \csname interpretmodule*\endcsname {module~environm
\stex_every_module:n {
  \exp_after:wN \stex_reactivate_macro:N \csname interpretmodule*\endcsname
}
\stex_sms_allow_env:n{interpretmodule*}

\stex_new_stylable_cmd:nnnn{interpretmod}{s m O{} m m}{
  \IfBooleanTF#1\stex_structural_feature_morphism_with_macros:nnnnn
  \stex_structural_feature_morphism:nnnnn{#2}{morphism}{#3}{#4}{,data-ftml-total=true}
  \clist_map_function:nN{#5}\__stex_morphisms_parse_assign:n
  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #2 }
    \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_check_total:
  \stex_structural_feature_morphism_end:
  \stex_smsmode_do:
}{}

\stex_deactivate_macro:Nn \interpretmod {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \interpretmod
}
\stex_sms_allow_escape:N\interpretmod
```

\assign

```
\stex_new_stylable_cmd:nnnn {assign} { m m }{
  \stex_get_in_morphism:n{#1}
  \stex_if_do_html:TF{
    \stex_annotate_invisible:n{\hbox{$\_stex_assign_do:n{#2}$}}
  }{
    \_stex_assign_do:n{#2}
  }
  \stex_smsmode_do:
}{}
\stex_sms_allow_escape:N\assign
\stex_deactivate_macro:Nn \assign {morphism~environments}

```

```
\cs_new_protected:Nn \_stex_assign_do:n{
  \stex_debug:nn{assign}{Assigning~\stex_use_symbol_uri:N \l_stex_get_symbol_uri~to~\tl_to_
  \stex_check_term:nn{assignment}{#1}
  \stex_if_do_html:T{
    \stex_annotate_invisible:nn{data-ftml-assign={\stex_use_symbol_uri:N \l_stex_get_symbol
      \_stex_annotate_force_break:n{
        \mode_if_math:TF{\stex_annotate:nn{data-ftml-definiens={}}{#1}}{\hbox{$\stex_annota
      }
    }
  }
  \exp_after:wN \stex_metagroup_do_in:n \exp_after:wN {
    \exp_after:wN \seq_put_right:Nn \exp_after:wN  \l_stex_morphism_assigns_seq
    \exp_after:wN { \l_stex_get_symbol_uri }
  }
}

\cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
  \str_clear:N \l__stex_morphisms_name_str
  \str_clear:N \l__stex_morphisms_newname_str
  \tl_clear:N \l__stex_morphisms_ass_tl
  \stex_debug:nn{morphisms}{Parsing~#1}
  \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{@}} {#1}
  \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
    \stex_debug:nn{morphisms}{No~@}
    \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
  }{
    \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
    \stex_debug:nn{morphisms}{Name:~\l__stex_morphisms_name_str}
    \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
    \tl_set:Ne \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq @}
  }
  \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
  \str_if_empty:NTF \l__stex_morphisms_name_str {
    \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
    \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
    \tl_set:Ne \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
  }{
    \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
    \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
    \tl_set:Ne \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
  }
  \__stex_morphisms_do_parsed_assign:
}

\cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
  \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str
  \str_if_empty:NF \l__stex_morphisms_newname_str {
    \stex_debug:nn{morphisms}{renaming~\l__stex_morphisms_name_str;~to~\l__stex_morphisms_n
    \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__st
  }
  \tl_if_empty:NF \l__stex_morphisms_ass_tl {
    \stex_debug:nn{morphisms}{assigning~\l__stex_morphisms_name_str;~to~\exp_args:No \tl_to
    \exp_args:No \_stex_assign_do:n \l__stex_morphisms_ass_tl
  }
```

```
}

\cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
  \peek_charcode:NTF [ {
    \__stex_morphisms_do_parsed_newname:w
  }{
    \__stex_morphisms_do_parsed_newname:w []
  }
}

\cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end:
  \_stex_renamedecl_do:nn{#1}{#2}
}
```

(*End of definition for* `\assign`. *This function is documented on page 146.*)

sfunction

`\renamedecl`

```
\stex_new_stylable_cmd:nnnn {renamedecl} { m O{} m }{
  \stex_get_in_morphism:n{#1}
  \_stex_renamedecl_do:nn{#2}{#3}
  \stex_smsmode_do:
}{}
\stex_sms_allow_escape:N\renamedecl
\stex_deactivate_macro:Nn \renamedecl {morphism~environments}

\cs_new_protected:Nn \_stex_renamedecl_do:nn {
  \stex_debug:nn{renamedecl}{Renaming~\stex_use_symbol_uri:N \l_stex_get_symbol_uri~to~[#1]
  \stex_if_do_html:T{
    \exp_args:Ne \stex_annotate_invisible:nn{
      data-ftml-rename={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
      data-ftml-macroname={#2}
      \str_if_empty:nF{#1}{ ,data-ftml-to={#1} }
    }{}
  }
  \stex_metagroup_do_in:e {
    \seq_push:Nn \exp_not:N \l_stex_morphism_renames_seq
    {{\l_stex_get_symbol_uri}{{#2}{
      \tl_if_empty:nTF{#1}{
        \l_stex_feature_name_str/\stex_symbol_uri_name:N \l_stex_get_symbol_uri
      }{#1}
    }}}
  }
}
```

(*End of definition for* `\renamedecl`. *This function is documented on page 146.*)

sfunction

`\assignMorphism`

```
\stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
  \str_clear:N \l__stex_morphisms_morphism_dom_str
  \exp_args:No \stex_iterate_morphisms:nn\l_stex_current_domain_uri{
    \stex_debug:nn{assignMorphism}{
      Checking:~#1~vs:^^J##1^^J##2^^J##3^^J##4
```

```
    }
    \str_if_eq:nnTF{#1}{##1}{
      \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
    }{
      \stex_str_if_ends_with:nnT{##2}{#1}{
        \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
      }
    }
  }
  \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
    \msg_error:nnn{stex}{error/nomorphism}{#1}
  }
  \bool_set_false:N \l_tmpa_bool
  \stex_iterate_morphisms:nn \l_stex_current_module_str {
    \stex_debug:nn{assignMorphism}{
      Checking:~#2~vs:^^J##1^^J##2^^J##3^^J##4
    }
    \str_if_eq:nnTF{#2}{##1}{
      \stex_debug:nn{assignMorphism}{match!}
      \stex_iterate_break:n{
        \stex_annotate_invisible:nn{
          data-ftml-assignmorphismfrom={\l__stex_morphisms_morphism_dom_str}
          data-ftml-assignmorphismto={\l_stex_current_module_str?##1}
        }{}
        \bool_set_true:N \l_tmpa_bool
      }
    }{
      \stex_str_if_ends_with:nnT{##2}{#2}{
        \stex_debug:nn{assignMorphism}{match!}
        \stex_iterate_break:n{
          \stex_annotate_invisible:nn{
            data-ftml-assignmorphismfrom={\l__stex_morphisms_morphism_dom_str},
            data-ftml-assignmorphismto={\l_stex_current_module_str?##1}
          }{}
          \bool_set_true:N \l_tmpa_bool
        }
      }
    }
  }
  \bool_if:NF \l_tmpa_bool {
    \msg_error:nnn{stex}{error/nomorphism}{#2}
  }
}{}
\stex_sms_allow_escape:N \assignMorphism
\stex_deactivate_macro:Nn \assignMorphism {morphism~environments}

\cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
  \stex_iterate_break:n{
    \str_set:Ne \l__stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
    \stex_debug:nn{assignMorphism}{match!}
    \stex_iterate_symbols:nn{#2}{
      \stex_debug:nn{assignMorphism}{removing~##1?##3}
      % TODO: non-trivial assignments
      \prop_remove:Nn \l_stex_morphism_symbols_prop {
```

```
3738            [##1]/[##3]
3739          }
3740        }
3741      }
3742    }
```

(*End of definition for* `\assignMorphism`. *This function is documented on page 146.*)

sfunction

# Chapter 32

# Symbols

```
⟨@@=stex_syms⟩
```

## 32.1 Declarations

Keys used in various symbol declarations:

```
\stex_keys_define:nnnn{symargs}{
  \str_clear:N \l_stex_key_args_str
  \str_clear:N \l_stex_key_role_str
  \str_clear:N \l_stex_key_reorder_str
  \str_clear:N \l_stex_key_assoc_str
}{
  args      .str_set:N  = \l_stex_key_args_str ,
  reorder   .str_set:N  = \l_stex_key_reorder_str ,
  assoc     .choices:nn = {bin,binl,binr,pre,conj,pwconj}
    {\str_set:Ne \l_stex_key_assoc_str \l_keys_choice_tl},
  role      .str_set:N  = \l_stex_key_role_str
}{}

\stex_keys_define:nnnn{decl}{
  \str_clear:N \l_stex_key_name_str
  \str_clear:N \l_stex_key_args_str
  \tl_clear:N \l_stex_key_type_tl
  \tl_clear:N \l_stex_key_def_tl
  \tl_clear:N \l_stex_key_return_tl
  \str_clear:N \l_stex_key_wikidata_str
  \clist_clear:N \l_stex_key_argtypes_clist
}{
  name      .str_set:N    = \l_stex_key_name_str ,
  return    .tl_set:N     = \l_stex_key_return_tl ,
  argtypes  .clist_set:N  = \l_stex_key_argtypes_clist ,
  wikidata  .str_set:N    = \l_stex_key_wikidata_str ,
  type      .tl_set:N     = \l_stex_key_type_tl  ,
  def       .tl_set:N     = \l_stex_key_def_tl   ,
  align     .code:n       = {},
  gf        .code:n       = {}
}{style,deprecate,symargs}
```

\symdecl

```
\str_new:N \l_stex_macroname_str

\stex_new_stylable_cmd:nnnn {symdecl} { s m O{}} {
  \stex_keys_set:nn{decl}{#3}
  \IfBooleanTF #1 {
    \str_clear:N \l_stex_macroname_str
  }{
    \str_set:Ne \l_stex_macroname_str { #2 }
  }
  \__stex_syms_top:n{#2}
  \stex_if_smsmode:F \__stex_syms_style:
  \stex_smsmode_do:
}{}

\stex_deactivate_macro:Nn \symdecl {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \symdecl}
\stex_sms_allow_escape:N \symdecl

\cs_new_protected:Nn \__stex_syms_style: {
  \group_begin:
    \tl_set:Ne \thisdecluri {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
    \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
    \tl_set_eq:NN \thistype \l_stex_key_type_tl
    \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
    \tl_set_eq:NN \thisargs \l_stex_key_args_str
    \tl_clear:N \thisstyle
    \stex_style_apply:
  \group_end:
}
```

(*End of definition for* `\symdecl`. *This function is documented on page 147.*)

sfunction

`\symdef`

```
\stex_new_stylable_cmd:nnnn {symdef} { m O{} m} {
  \stex_keys_set:nn{symdef}{#2}
  \str_set:Ne \l_stex_macroname_str { #1 }
  \__stex_syms_top:n{#1}
  \stex_debug:nn{symdef}{Doing~\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
  \str_set_eq:NN \l_stex_get_symbol_uri \l_stex_current_symbol_uri
  \stex_notation_parse:n{#3}
  \stex_if_check_terms:T{ \_stex_notation_check: }
  \_stex_notation_add:
  \stex_if_do_html:T{
    \_stex_notation_do_html:n{\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
  }
  \stex_if_smsmode:F \__stex_syms_def_style:
  \stex_smsmode_do:
}{}

\stex_deactivate_macro:Nn \symdef {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \symdef}
\stex_sms_allow_escape:N \symdef

```

```
\cs_new_protected:Nn \__stex_syms_def_style: {
  \group_begin:
  \tl_set:Nx \thisdecluri {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
  \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
  \tl_set_eq:NN \thistype \l_stex_key_type_tl
  \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
  \tl_set_eq:NN \thisargs \l_stex_key_args_str
  \tl_clear:N \thisstyle
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \def\thisnotation{
    \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
      \_stex_notation_make_args:
    }
  }
  \stex_style_apply:
  \group_end:
}
```

(the symdef keyset is defined after notations as `\stex_keys_define:nnnn{symdef}{}{}{decl,notation}` )

(*End of definition for* `\symdef`. *This function is documented on page 147.*)

sfunction

`\textsymdecl`

```
\stex_keys_define:nnnn{textsymdecl}{
  \str_clear:N \l_stex_key_name_str
  \tl_clear:N \l_stex_key_type_tl
  \tl_clear:N \l_stex_key_def_tl
}{
  name      .str_set:N  = \l_stex_key_name_str ,
  type      .tl_set:N     = \l_stex_key_type_tl ,
  def       .tl_set:N     = \l_stex_key_def_tl,
  gf        .code:n       = {}
}{style,deprecate}

\stex_new_stylable_cmd:nnnn {textsymdecl} {m O{} m} {
  \stex_keys_set:nn{symdef}{}
  \stex_keys_set:nn{textsymdecl}{#2}
  \str_set:Ne \l_stex_macroname_str { #1 }
  \str_if_empty:NT \l_stex_key_name_str {
    \str_set:Nn \l_stex_key_name_str {#1}
  }%{
  %  \str_set:Ne \l_stex_key_name_str {\l_stex_key_name_str-sym}
  %}
  \str_set:Nn \l_stex_key_role_str {textsymdecl}
  \tl_set:Ne \l_stex_current_symbol_uri {
    \exp_args:No \stex_new_symbol_uri:n \l_stex_key_name_str
  }
  \stex_symdecl_do:
  \_stex_check_terms:
  \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnnN}{
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {0}{}
```

```
    {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
    {}% type
    {\use:c{#1name_nospace}}% return
    \stex_invoke_text_symbol:
  }
  \exp_args:Ne \stex_ref_new_symbol:n
    { \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
  \stex_if_do_html:T {
    \_stex_symdecl_html:
  }

  \int_set:Nn \l_stex_get_symbol_arity_int 0
  \tl_clear:N \l_stex_key_op_tl
  \str_clear:N \l_stex_key_intent_str
  \str_clear:N \l_stex_key_prec_str
  \tl_set_eq:NN \l_stex_get_symbol_uri \l_stex_current_symbol_uri
  \stex_notation_parse:n{\hbox{#3}}
  \_stex_notation_add:
  \stex_if_do_html:T {
    \def\comp{\_comp}
    \_stex_notation_do_html:n{ \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
  }
  \stex_execute_in_module:e{
    \__stex_syms_set_textsymdecl_macro:nnn{#1}{ \stex_use_symbol_uri:N \l_stex_current_symb
    \exp_not:n{{#3}}
  }

  \stex_if_smsmode:F{
    \group_begin:
    \tl_set:Ne \thisdecluri {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
    \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
  }
  \stex_smsmode_do:
}{}

\stex_deactivate_macro:Nn \textsymdecl {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \textsymdecl}
\stex_sms_allow_escape:N \textsymdecl

\cs_new_protected:Nn \__stex_syms_set_textsymdecl_macro:nnn {
  \cs_set_protected:cpn{#1name_nospace}{#3}
  \cs_set_protected:cpn{#1name}{
    \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
    \mode_if_math:T\hbox{\let\xspace\relax #3}
    \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
  }
}
```

(*End of definition for* `\textsymdecl`. *This function is documented on page 147.*)

sfunction

`\__stex_syms_top:n` shared behaviour for `\symdecl` and `\symdef`; calls `\stex_symdecl_do:`, optionally checks

the term components, adds the symbol to the current module and produces the FTML for the declaration.

```
\cs_new_protected:Nn \__stex_syms_top:n {
  \str_if_empty:NT \l_stex_key_name_str {
    \str_set:Ne \l_stex_key_name_str { #1 }
  }
  \tl_set:Ne \l_stex_current_symbol_uri {
    \exp_args:No \stex_new_symbol_uri:n \l_stex_key_name_str
  }

  \stex_symdecl_do:
  \_stex_check_terms:
  \__stex_syms_add_decl:
  \stex_if_do_html:T \_stex_symdecl_html:
}
```

(*End of definition for* `\__stex_syms_top:n`.)

Adds the symbol to the current module:

```
\cs_new_protected:Nn \__stex_syms_add_decl: {
  \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnnN}{
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {\int_use:N \l_stex_get_symbol_arity_int}
    {\l_stex_get_symbol_args_tl}
    {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
    {}
    {\exp_args:No \exp_not:n \l_stex_key_return_tl}
    \stex_invoke_symbol:
  }
  \exp_args:Ne \stex_ref_new_symbol:n
    {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
}
```

`\stex_symdecl_do:`

```
\cs_new_protected:Nn \stex_symdecl_do: {
  \_stex_do_deprecation:n \l_stex_key_name_str
  \__stex_syms_parse_arity:
  \__stex_syms_do_args:
}

\cs_new:Nn \_stex_return_args:nn {
  {\svar{ARGUMENT_#1}\_stex_eat_exclamation_point:}
}

\cs_new_protected:Nn \__stex_syms_do_args: {
  \tl_clear:N \l_stex_get_symbol_args_tl
  \int_step_inline:nn \l_stex_get_symbol_arity_int {
    \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
    \tl_put_right:Ne \l_stex_get_symbol_args_tl {
      \str_item:Nn \l_stex_key_args_str {##1}
    }
  }
}
```

Constructs the arity string of the symbol:

```
\int_new:N \l_stex_assoc_args_count

\cs_new_protected:Nn \__stex_syms_parse_arity: {
  \int_zero:N \l_stex_get_symbol_arity_int
  \int_zero:N \l_stex_assoc_args_count
  \str_map_inline:Nn \l_stex_key_args_str {
    \str_case:nnF ##1 {
      0 { \str_map_break: }
      1 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {1}
        \str_set:Nn \l_stex_key_args_str {i}
      } }
      2 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {2}
        \str_set:Nn \l_stex_key_args_str {ii}
      } }
      3 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {3}
        \str_set:Nn \l_stex_key_args_str {iii}
      } }
      4 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {4}
        \str_set:Nn \l_stex_key_args_str {iiii}
      } }
      5 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {5}
        \str_set:Nn \l_stex_key_args_str {iiiii}
      } }
      6 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {6}
        \str_set:Nn \l_stex_key_args_str {iiiiii}
      } }
      7 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {7}
        \str_set:Nn \l_stex_key_args_str {iiiiiii}
      } }
      8 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {8}
        \str_set:Nn \l_stex_key_args_str {iiiiiiii}
      } }
      9 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {9}
        \str_set:Nn \l_stex_key_args_str {iiiiiiiii}
      } }
      i {\int_incr:N \l_stex_get_symbol_arity_int}
      b {\int_incr:N \l_stex_get_symbol_arity_int}
      a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
      B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
    }{
      \msg_error:nnxx{stex}{error/wrongargs}{}{##1}
    }
  }
}
```

(*End of definition for* `\stex_symdecl_do:`. *This function is documented on page 147.*)

sfunction

`\_stex_symdecl_html:`

```
\cs_new_protected:Nn \_stex_symdecl_html: {
  \stex_annotate_invisible:n{
    \hbox{\exp_args:Ne \stex_annotate:nn {
      data-ftml-symdecl = { \l_stex_key_name_str},
      data-ftml-args = {\l_stex_key_args_str}
      \str_if_empty:NF \l_stex_macroname_str {,
        data-ftml-macroname={\l_stex_macroname_str}
      }
      \str_if_empty:NF \l_stex_key_wikidata_str {,
        data-ftml-wikidata={\l_stex_key_wikidata_str}
      }
      \str_if_empty:NF \l_stex_key_assoc_str {,
        data-ftml-assoctype={\l_stex_key_assoc_str}
      }
      \str_if_empty:NF \l_stex_key_reorder_str {,
        data-ftml-reorderargs={\l_stex_key_reorder_str}
      }
      \str_if_empty:NF \l_stex_key_role_str {,
        data-ftml-role={\l_stex_key_role_str}
      }
    }{\_stex_annotate_force_break:n{
      \bool_set_true:N \stex_in_invisible_html_bool
      \tl_if_empty:NF \l_stex_key_type_tl {
        $\stex_annotate:nn{data-ftml-type={}}{\l_stex_key_type_tl}$
      }
      \tl_if_empty:NF \l_stex_key_def_tl {
        $\stex_annotate:nn{data-ftml-definiens={}}{\l_stex_key_def_tl}$
      }
      \tl_if_empty:NF \l_stex_key_return_tl{
        \exp_args:Nno \use:n{
        \cs_generate_from_arg_count:NNnn \l__stex_syms_cs
        \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
        \tl_set:Ne \l__stex_syms_args_tl {\_stex_map_args:N \_stex_return_args:nn}
        $\stex_annotate:nn{data-ftml-returntype={}}{
          \exp_after:wN \l__stex_syms_cs \l__stex_syms_args_tl!
        }$
      }
      \clist_if_empty:NF \l_stex_key_argtypes_clist {
        \stex_annotate:nn{data-ftml-argtypes={}}{\_stex_annotate_force_break:n{
          \clist_map_inline:Nn \l_stex_key_argtypes_clist {
              $\stex_annotate:nn{data-ftml-type={}}{##1}$
          }
        }}
      }
    }}
  }}
}
```

(*End of definition for* `\_stex_symdecl_html:`. *This function is documented on page 147.*)

sfunction

\stex_add_symbol:nnnnnnN

```
\cs_new_protected:Nn \stex_add_symbol:nnnnnnnN {
  \stex_debug:nn{declaration}{New~declaration:~\stex_use_module_uri:N \l_stex_current_modul
    Macro:#1^^JArity:#3~(#4)^^J
    Def:~\tl_to_str:n{#5}^^J
    Type:~\tl_to_str:n{#6}^^J
    Returns:~\tl_to_str:n{#7}^^J
    Invokes:~\tl_to_str:n{#8}
  }
  \prop_gput:cnn{ \_stex_symbol_macro:N \l_stex_current_module_uri }
  {#2}{{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
  \tl_if_empty:nF{#1}{
    \stex_do_up_to_module:n {
      \__stex_syms_activate_i:nnnnnnnn{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}
    }
  }
}
```

Generate semantic macros etc.:

```
\cs_new_protected:Nn \_stex_activate_symbols: {
  \stex_debug:nn{activating}{All~symbols:~\cs_meaning:c{ \_stex_symbol_macro:N \l_stex_curr
  \prop_map_inline:cn{ \_stex_symbol_macro:N \l_stex_current_module_uri }{
    \__stex_syms_maybe_activate:nnnnnnnn ##2
  }
}

\cs_new_protected:Nn \__stex_syms_maybe_activate:nnnnnnnn {
  \str_if_empty:nF{#1}{
    \__stex_syms_activate_i:nnnnnnnn {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}
  }
}

\cs_new_protected:Nn \__stex_syms_activate_i:nnnnnnnn {
  \stex_debug:nn{activating}{macro~#1:\stex_use_module_uri:N \l_stex_current_module_uri ? #
    \tl_to_str:n{{#3}{#4}{#5}{#6}{#7}#8}
  }
  \cs_set:cpe{#1} {
    \_stex_invoke_symbol:nnnnnnN
    {\exp_args:No \stex_new_symbol_uri:nn {\l_stex_current_module_uri} {#2} }
    \exp_not:n{{#3}{#4}{#5}{#6}{#7}{#8}}
  }
}
```

(*End of definition for* `\stex_add_symbol:nnnnnnN`. *This function is documented on page 147.*)

sfunction

\stex_has_definiens_p:N
\stex_has_definiens:N*TF*
\stex_has_definiens_p:n
\stex_has_definiens:n*TF*

```
\prg_new_conditional:Nnn \stex_has_definiens:N { p, T, F, TF } {
  \exp_args:No \stex_has_definiens:nTF #1 \prg_return_true: \prg_return_false:
}
\prg_new_conditional:Nnn \stex_has_definiens:n { p, T, F, TF } {
  \exp_args:Nne \use:nn{\__stex_syms_has_definiens:nnnnnnnN}{\exp_args:Nne \prop_item:cn{
    \exp_args:Ne \_stex_symbol_macro:n {\stex_symbol_uri_module:n{#1}}
  }{
```

```
    \stex_symbol_uri_name:n {#1}
  }}
}

\cs_new:Nn \__stex_syms_has_definiens:nnnnnnnN {
  \tl_if_empty:nTF{#5}\prg_return_false:\prg_return_true:
}
```

(*End of definition for* `\stex_has_definiens:NTF` *and* `\stex_has_definiens:nTF`. *These functions are documented on page 147.*)

sfunction

## 32.2 Retrieval

`\stex_iterate_symbols:n`
`\stex_iterate_break:`
`\stex_iterate_break:n`

```
\cs_new_protected:Nn \stex_iterate_symbols:n {
  \stex_pseudogroup_with:nn{\__stex_syms_sym_cs:nnnnnnnnN\stex_iterate_break:\stex_iterate_
    \cs_set:Npn \__stex_syms_sym_cs:nnnnnnnnN
    ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
    \cs_set:Npn \stex_iterate_break: {
      \prop_map_break:n{\seq_map_break:}
    }
    \cs_set:Npn \stex_iterate_break:n ##1 {
      \prop_map_break:n{\seq_map_break:n{##1}}
    }
    \seq_map_inline:Nn \l_stex_all_modules_seq {
      \prop_map_inline:cn{\_stex_symbol_macro:n {##1}}{
        \__stex_syms_sym_cs:nnnnnnnnN {##1} ####2
      }
    }
  }
}
```

(*End of definition for* `\stex_iterate_symbols:n`, `\stex_iterate_break:`, *and* `\stex_iterate_break:n`. *These functions are documented on page 148.*)

sfunction

`\stex_iterate_symbols:nn`

```
\cs_new_protected:Nn \stex_iterate_symbols:nn {
  \seq_clear:N \l__stex_syms_mods_seq
  \stex_pseudogroup_with:nn{\__stex_syms_sym_cs:nnnnnnnnN}{
    \cs_set:Npn \__stex_syms_sym_cs:nnnnnnnnN
    ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
    \clist_map_function:nN {#1} \__stex_syms_it_decl_i:n
  }
}

\cs_new_protected:Nn \__stex_syms_it_decl_i:n {
  \seq_if_in:NnF \l__stex_syms_mods_seq {#1} {
    \seq_put_left:Nn \l__stex_syms_mods_seq {#1}
    \prop_map_inline:cn{\_stex_morphisms_macro:n {#1} }{
      \__stex_syms_it_decl_check:nnnn ##2
    }
```

```
    \prop_map_inline:cn{\_stex_symbol_macro:n {#1} }{
      \__stex_syms_sym_cs:nnnnnnnnN {#1} ##2
    }
  }
}

\cs_new_protected:Nn \__stex_syms_it_decl_check:nnnn {
  \tl_if_empty:nT{#1}{
    \__stex_syms_it_decl_i:n {#2}
  }
}
```

(*End of definition for* `\stex_iterate_symbols:nn`. *This function is documented on page 148.*)

sfunction

`\stex_get_symbol:n`
`\stex_get_symbol:nTF`
`\stex_get_symbol:nF`

```
\int_new:N \l_stex_get_symbol_arity_int

\cs_new_protected:Nn \stex_get_symbol:n {
  \stex_get_symbol:nF{ #1 }{
    \msg_error:nnn{stex}{error/unknownsymbol}{#1}
  }
}

\cs_new_protected:Npn \stex_get_symbol:nTF #1 #2 #3 {
  \tl_clear:N \l_stex_get_symbol_uri
  \cs_if_exist:cTF { #1 }{
    \cs_set_eq:Nc \l__stex_syms_cs { #1 }
    % command name
    \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l__stex_syms_cs }{
      % ...that takes no arguments
      \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l__stex_syms_cs}
        \_stex_invoke_symbol:nnnnnnN
        \__stex_syms_get_symbol_from_cs:
        {\__stex_syms_get_symbol_from_string:n { #1 }}
    }{
      \__stex_syms_get_symbol_from_string:n { #1 }
    }
  }{
    \__stex_syms_get_symbol_from_string:n { #1 }
  }
  \tl_if_empty:NTF \l_stex_get_symbol_uri { #3 } { #2 }
}

\cs_new_protected:Npn \stex_get_symbol:nF #1 #2 {
  \stex_get_symbol:nTF{ #1 }{}{ #2 }
}

\cs_new_protected:Nn \__stex_syms_get_symbol_from_cs: {
  \stex_pseudogroup_with:nn{\_stex_invoke_symbol:nnnnnnN}{
    \cs_set:Npn \_stex_invoke_symbol:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 {
      \tl_set:Nn \l_stex_get_symbol_uri { ##1 }
      \int_set:Nn \l_stex_get_symbol_arity_int {##2}
      \tl_set:Nn \l_stex_get_symbol_args_tl {##3}
```

```
      \tl_set:Nn \l_stex_get_symbol_def_tl {##4}
      \tl_set:Nn \l_stex_get_symbol_type_tl {##5}
      \tl_set:Nn \l_stex_get_symbol_return_tl {##6}
      \tl_set:Nn \l_stex_get_symbol_invoke_cs {##7}
    }
    \l__stex_syms_cs
  }
}

\cs_new_protected:Nn \__stex_syms_get_symbol_from_string:n {
  \stex_debug:nn{symbols}{Getting~from~string~#1...}
  \seq_set_split:Nnn \l__stex_syms_seq ? {#1}
  \seq_pop_right:NN \l__stex_syms_seq \l__stex_syms_name
  \seq_if_empty:NTF \l__stex_syms_seq {
    \exp_args:No \__stex_syms_get_from_one_string:n {#1}
  }{
    \exp_args:NNe \exp_args:Nno \__stex_syms_get_symbol_from_modules:nn {
      \seq_use:Nn \l__stex_syms_seq ?
    } \l__stex_syms_name
  }
}

\cs_new_protected:Nn \__stex_syms_sym_from_str_i:nnnn {
  \bool_lazy_any:nTF{
    {\str_if_eq_p:nn{#2}{#3}}
    {\str_if_eq_p:nn{#2}{#4}}
    {\stex_str_if_ends_with_p:nn{#4}{/#2}}
    }{
      \__stex_syms_sym_i_finish:nnnnnnnN{#1}{#4}
    }{
      \__stex_syms_sym_i_gobble:nnnnnn
    }
}
\cs_new_protected:Nn \__stex_syms_sym_i_gobble:nnnnnn {}

\cs_new_protected:Nn \__stex_syms_sym_i_finish:nnnnnnnN {
  \prop_map_break:n{\seq_map_break:n{
    \tl_set:Ne \l_stex_get_symbol_uri { \stex_new_symbol_uri:nn {#1} {#2}}
    \int_set:Nn \l_stex_get_symbol_arity_int {#3}
    \tl_set:Nn \l_stex_get_symbol_args_tl {#4}
    \tl_set:Nn \l_stex_get_symbol_def_tl {#5}
    \tl_set:Nn \l_stex_get_symbol_type_tl {#6}
    \tl_set:Nn \l_stex_get_symbol_return_tl {#7}
    \tl_set:Nn \l_stex_get_symbol_invoke_cs {#8}
  }}
}

\cs_new_protected:Nn \__stex_syms_get_symbol_from_modules:nn {
  %\seq_set_split:Nnn \l_tmpa_seq ? {#1}
  %\seq_pop_right:NN \l_tmpa_seq \l__stex_syms_name_str
  %\str_set:Ne \l__stex_syms_path_str {\seq_use:Nn \l_tmpa_seq ?}
  \stex_debug:nn{symbols}{Getting~#2~in~#1...}
  \seq_map_inline:Nn \l_stex_all_modules_seq {
    %\stex_debug:nn{symbols}{comparing~\stex_module_uri_as_qm:n{##1}~and~#1}
```

```
    \exp_args:Ne \stex_str_if_ends_with:nnT{\stex_module_uri_as_qm:n{##1}}{#1}{
      \prop_map_inline:cn{\_stex_symbol_macro:n {##1} }{
        \__stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
      }
    }

    %\str_if_empty:NTF \l__stex_syms_path_str {
    %  \exp_args:NNe \exp_args:Nno \stex_str_if_ends_with:nnT {\stex_module_uri_name:n{##1}
    %}{
    %  \exp_args:NNNe \exp_args:No \str_if_eq:nnT\l__stex_syms_name_str{
    %    \stex_module_uri_name:n {##1}
    %  }
    %}{
    %  \str_if_empty:NTF \l__stex_syms_path_str {
    %    \prop_map_inline:cn{\_stex_symbol_macro:n {##1} }{
    %      \__stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
    %    }
    %  }{
    %    \stex_debug:nn{symbols}{Comparing~\l__stex_syms_path_str;~with~\tl_to_str:n{##1}}
    %    \exp_args:NNe \exp_args:Nno \stex_str_if_ends_with:nnT{\stex_module_uri_path:n {##
    %      \prop_map_inline:cn{\_stex_symbol_macro:n {##1} }{
    %        \__stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
    %      }
    %    }
    %  }
    %}
  }
}

\prg_new_conditional:Nnn \__stex_syms_uri_match:n {T,F,TF} {
  \str_if_eq:nnT
}


\cs_new_protected:Nn \__stex_syms_get_from_one_string:n {
  \stex_debug:nn{symbols}{Getting~#1~anywhere...}
  \stex_iterate_symbols:n{
    %\stex_debug:nn{symbols}{>#1==##2~|~#1==##3<...}
    \bool_lazy_any:nT{
      {\str_if_eq_p:nn{#1}{##2}}
      {\str_if_eq_p:nn{#1}{##3}}
      {\stex_str_if_ends_with_p:nn{##3}{/#1}}
      }{
        \stex_iterate_break:n{
          \tl_set:Ne \l_stex_get_symbol_uri { \stex_new_symbol_uri:nn {##1} {##3}}
          \int_set:Nn \l_stex_get_symbol_arity_int {##4}
          \tl_set:Nn \l_stex_get_symbol_args_tl {##5}
          \tl_set:Nn \l_stex_get_symbol_def_tl {##6}
          \tl_set:Nn \l_stex_get_symbol_type_tl {##7}
          \tl_set:Nn \l_stex_get_symbol_return_tl {##8}
          \tl_set:Nn \l_stex_get_symbol_invoke_cs {##9}
        }
    }
  }
```

```
}
```

(*End of definition for* `\stex_get_symbol:n`, `\stex_get_symbol:nTF`, *and* `\stex_get_symbol:nF`. *These functions are documented on page 148.*)

sfunction

## 32.3 Variables

```
⟨@@=stex_vars⟩
```

`\l_stex_variables_prop`

```
\prop_new:N \l_stex_variables_prop
```

(*End of definition for* `\l_stex_variables_prop`. *This variable is documented on page 148.*)

`\vardef`

```
\bool_new:N \l__stex_vars_bind_bool
\cs_new_protected:Nn \_stex_variable:nnnnnnnN {}

\stex_new_stylable_cmd:nnnn {vardef} { m O{} m} {
  \stex_keys_set:nn{vardef}{#2}
  \str_set:Ne \l_stex_macroname_str { #1 }
  \str_if_empty:NT \l_stex_key_name_str {
    \str_set:Ne \l_stex_key_name_str { #1 }
  }

  \stex_symdecl_do:
  \stex_if_check_terms:T \_stex_check_terms:
  \__stex_vars_add:
  \__stex_vars_macro:
  \stex_if_do_html:T \__stex_vars_html:

  \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
  \stex_debug:nn{vardef}{Doing~\l_stex_key_name_str}
  \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
  \stex_notation_parse:n{#3}
  \stex_if_check_terms:T{ \_stex_notation_check: }
  \_stex_var_notation_macro:
  \stex_if_do_html:T {
    \def\comp{\_varcomp}
    \_stex_notation_do_html:n \l_stex_key_name_str
  }
  \group_begin:
  \tl_set_eq:NN \thisvarname \l_stex_key_name_str
  \tl_clear:N \thisstyle
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \def\thisnotation{
    \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
      \_stex_notation_make_args:
    }
  }
  \stex_style_apply:
  \group_end:\ignorespaces
}{}
```

(*End of definition for* `\vardef`. *This function is documented on page 148.*)

sfunction

Adds a new variable to the current scope:

```
\cs_new_protected:Nn \__stex_vars_add: {
  \exp_args:NNNo \exp_args:NNne
  \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {\int_use:N \l_stex_get_symbol_arity_int}
    {\l_stex_get_symbol_args_tl}
    {\exp_args:No \exp_not:n \l_stex_key_def_tl}
    {\exp_args:No \exp_not:n \l_stex_key_type_tl}
    {\exp_args:No \exp_not:n \l_stex_key_return_tl}
    \stex_invoke_symbol:
  }
}
```

Defines the macro for a variable:

```
\cs_new_protected:Nn \__stex_vars_macro: {
  \tl_set:ce{\l_stex_macroname_str}{
    \_stex_invoke_variable:nnnnnnN
      {\l_stex_key_name_str}
      {\int_use:N \l_stex_get_symbol_arity_int}
      {\l_stex_get_symbol_args_tl}
      {\exp_args:No \exp_not:n \l_stex_key_def_tl}
      {\exp_args:No \exp_not:n \l_stex_key_type_tl}
      {\exp_args:No \exp_not:n \l_stex_key_return_tl}
      \stex_invoke_symbol:
  }
}
```

Insert the HTML for a variable declaration:

```

\cs_new_protected:Nn \__stex_vars_html: {
  \stex_if_do_html:T {
    \hbox\bgroup\exp_args:Ne \stex_annotate_invisible:nn {
      data-ftml-vardef = {\l_stex_key_name_str},
      data-ftml-args = {\l_stex_key_args_str}
      \str_if_empty:NF \l_stex_macroname_str {,
        data-ftml-macroname={\l_stex_macroname_str}
      }
      \str_if_empty:NF \l_stex_key_assoc_str {,
        data-ftml-assoctype={\l_stex_key_assoc_str}
      }
      \str_if_empty:NF \l_stex_key_role_str {,
        data-ftml-role={\l_stex_key_role_str}
      }
      \str_if_empty:NF \l_stex_key_reorder_str {,
        data-ftml-reorderargs={\l_stex_key_reorder_str}
      }
      \bool_if:NT \l__stex_vars_bind_bool {,
        data-ftml-bind={true}
      }
    }{
```

```
      \_stex_annotate_force_break:n{
        \bool_set_true:N \stex_in_invisible_html_bool
        \tl_if_empty:NF \l_stex_key_type_tl {
          \stex_annotate:nn{data-ftml-type={}}{$\l_stex_key_type_tl$}
        }
        \tl_if_empty:NF \l_stex_key_def_tl {
          \stex_annotate:nn{data-ftml-definiens={}}{$\l_stex_key_def_tl$}
        }
        \tl_if_empty:NF \l_stex_key_return_tl{
          \exp_args:Nno \use:n{
          \cs_generate_from_arg_count:NNnn \l__stex_vars_cs
          \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
          \tl_set:Ne \l__stex_vars_args_tl {\_stex_map_args:N \_stex_return_args:nn}
          $\stex_annotate:nn{data-ftml-returntype={}}{
            \exp_after:wN \l__stex_vars_cs \l__stex_vars_args_tl!}$
        }
        \tl_if_empty:NF \l_stex_key_argtypes_clist {
          \stex_annotate:nn{data-ftml-argtypes={}}{
            \_stex_annotate_force_break:n{
              \clist_map_inline:Nn \l_stex_key_argtypes_clist {
                $\stex_annotate:nn{data-ftml-type={}}{##1}$
              }
            }
          }
        }
      }\egroup
    }
}
```

\newvar

```
\NewDocumentCommand\newvar{ m O{}}{
  \vardef{v#1}[name=#1,#2]{#1}
}
```

(*End of definition for* \newvar. *This function is documented on page 148.*)

sfunction

\newseq

```
\NewDocumentCommand\newseq{ m O{} m}{
  \varseq{v#1}[name=#1,#2]{#3}{{\maincomp{#1}}\c_math_subscript_token{##1}}
}
```

(*End of definition for* \newseq. *This function is documented on page 148.*)

sfunction

\stex_get_var:n

```
\cs_new_protected:Nn \stex_get_var:n {
  \str_clear:N \l_stex_get_variable_str
  \__stex_vars_get_var:n{#1}
  \str_if_empty:NT \l_stex_get_variable_str {
    \msg_error:nnn{stex}{error/unknownsymbol}{#1}
  }
}
```

```
4439
4440 \cs_new_protected:Nn \__stex_vars_get_var:n {
4441   \prop_map_inline:Nn \l_stex_variables_prop {
4442     \__stex_vars_check_var:nnnnnnnnN {#1} ##2
4443   }
4444 }
4445
4446 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
4447   \str_if_eq:nnTF{#1}{#2}{
4448     \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}#9}
4449   }{
4450     \str_if_eq:nnT{#1}{#3}{
4451       \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}#9}
4452     }
4453   }
4454 }
4455
4456 \cs_new_protected:Nn \__stex_vars_set_vars:nnnnnnN {
4457   \stex_debug:nn{symbols}{Variable~#1~found}
4458   \cs_set:Npn \_stex_variable:nnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
4459   \str_set:Nn \l_stex_get_variable_str {#1}
4460   \int_set:Nn \l_stex_get_symbol_arity_int {#2}
4461   \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
4462   \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
4463   \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
4464   \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
4465   \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
4466 }
```

(*End of definition for* `\stex_get_var:n`. *This function is documented on page 149.*)

sfunction

### 32.3.1 Variable Sequences

4467 ⟨@@=stex_seqs⟩

`\varseq`

```
4468 \stex_new_stylable_cmd:nnnn {varseq}{m O{} m m} {
4469   \stex_keys_set:nn{symdef}{#2}
4470   \str_set:Ne \l_stex_macroname_str { #1 }
4471   \str_if_empty:NT \l_stex_key_name_str {
4472     \str_set:Ne \l_stex_key_name_str { #1 }
4473   }
4474   \str_if_empty:NT \l_stex_key_args_str {
4475     \str_set:Nn \l_stex_key_args_str {1}
4476   }
4477   \stex_symdecl_do:
4478
4479   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4480   \clist_set:Nn \l__stex_seqs_range_clist {#3}
4481   \tl_if_empty:NTF \l_stex_key_op_tl {
4482     \stex_notation_parse:n{#4}
4483     \tl_clear:N \l_stex_key_op_tl
4484   }{
```

```
    \stex_notation_parse:n{#4}
  }
  \stex_if_do_html:T \__stex_seqs_html:
  \stex_if_check_terms:T \_stex_check_terms:
  \__stex_seqs_add:
  \__stex_seqs_macro:
  \stex_if_check_terms:T \_stex_notation_check:
  \_stex_var_notation_macro:
   \stex_if_do_html:T {
    \def\comp{\_varcomp}
    \_stex_notation_do_html:n \l_stex_key_name_str
  }
  \group_begin:
  \tl_set_eq:NN \thisvarname \l_stex_key_name_str
  \tl_clear:N \thisstyle
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \def\thisnotation{
    \l_stex_notation_macrocode_cs{}!
  }
  \stex_style_apply:
  \group_end:\ignorespaces
}{}

\cs_new_protected:Nn \__stex_seqs_add: {
  \exp_args:NNNo \exp_args:NNne
  \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {\int_use:N \l_stex_get_symbol_arity_int}
    {\l_stex_get_symbol_args_tl}
    {\exp_args:No \exp_not:n \l_stex_key_def_tl}
    {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
    {\exp_args:No \exp_not:n \l_stex_key_return_tl}
    \stex_invoke_sequence:
  }
}

\cs_new_protected:Nn \__stex_seqs_macro: {
  \tl_set:ce{\l_stex_macroname_str}{
    \_stex_invoke_variable:nnnnnnN
      {\l_stex_key_name_str}
      {\int_use:N \l_stex_get_symbol_arity_int}
      {\l_stex_get_symbol_args_tl}
      {\exp_args:No \exp_not:n \l_stex_key_def_tl}
      {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
      {\exp_args:No \exp_not:n \l_stex_key_return_tl}
      \stex_invoke_sequence:
  }
}

\cs_new_protected:Nn \__stex_seqs_html: {
    \exp_args:Ne \stex_annotate_invisible:nn {
      data-ftml-varseq = {\l_stex_key_name_str},
      data-ftml-args = {\l_stex_key_args_str}
```

```
      \str_if_empty:NF \l_stex_macroname_str {,
        data-ftml-macroname={\l_stex_macroname_str}
      }
      \str_if_empty:NF \l_stex_key_assoc_str {,
        data-ftml-assoctype={\l_stex_key_assoc_str}
      }
      \str_if_empty:NF \l_stex_key_role_str {,
        data-ftml-role={\l_stex_key_role_str}
      }
      \str_if_empty:NF \l_stex_key_reorder_str {,
        data-ftml-reorderargs={\l_stex_key_reorder_str}
      }
    }{\hbox\bgroup
      \_stex_annotate_force_break:n{

        $\stex_annotate:nn{data-ftml-seqrange={}}{
          \exp_args:Nno\symuse{Metatheory?sequence~range}\l__stex_seqs_range_clist
        }$

        \tl_if_empty:NF \l_stex_key_type_tl {
          \stex_annotate:nn{data-ftml-type={}}{$\l_stex_key_type_tl$}
        }
        \tl_if_empty:NF \l_stex_key_def_tl {
          \stex_annotate:nn{data-ftml-definiens={}}{$\l_stex_key_def_tl$}
        }
        \tl_if_empty:NF \l_stex_key_return_tl{
          \exp_args:Nno \use:n{
          \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
          \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
          \tl_set:Nx \l__stex_seqs_args_tl {\_stex_map_args:N \_stex_return_args:nn}
          \stex_annotate:nn{data-ftml-returntype={}}{
            $\exp_after:wN \l__stex_seqs_cs \l__stex_seqs_args_tl!$}
        }
        \tl_if_empty:NF \l_stex_key_argtypes_clist {
          \stex_annotate:nn{data-ftml-argtypes={}}{
            \_stex_annotate_force_break:n{
              \clist_map_inline:Nn \l_stex_key_argtypes_clist {
                \stex_annotate:nn{data-ftml-type={}}{$##1$}
              }
            }
          }
        }
      }
    \egroup}
}
```

(*End of definition for* `\varseq`. *This function is documented on page 149.*)

sfunction

## 32.4 Expressions

⟨`@@=stex_expr`⟩

`\symuse`

```
\cs_new_protected:Npn \symuse #1 {
  \stex_get_symbol:n{#1}
  \exp_args:Nno \use:n {\_stex_invoke_symbol:NeooooN
  \l_stex_get_symbol_uri
  {\int_use:N \l_stex_get_symbol_arity_int}
  \l_stex_get_symbol_args_tl
  \l_stex_get_symbol_def_tl
  \l_stex_get_symbol_type_tl
  \l_stex_get_symbol_return_tl}
  \l_stex_get_symbol_invoke_cs
}
```

(*End of definition for* `\symuse`. *This function is documented on page 149.*)

sfunction

`\_stex_next_symbol:n`

```
\tl_new:N \l__stex_expr_reset_tl

\cs_new_protected:Nn \_stex_next_symbol:n {
  \tl_set:Ne \l_stex_every_symbol_tl {
    \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
      \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl  {
        \exp_args:No \exp_not:n \l_stex_every_symbol_tl
      }
      \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
        \exp_args:No \exp_not:n \l__stex_expr_reset_tl
      }
    }
    \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl  {
      \exp_args:No \exp_not:n \l_stex_every_symbol_tl
    }
    \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
    \exp_not:N \l_stex_every_symbol_tl
    \exp_not:n{ #1 }
  }
}
\cs_generate_variant:Nn \_stex_next_symbol:n {e}
```

(*End of definition for* `\_stex_next_symbol:n`. *This function is documented on page 149.*)

sfunction

`\_stex_invoke_symbol:NnnnnnN`
`\_stex_invoke_symbol:NeooooN`
`\_stex_invoke_symbol:nnnnnnN`

- `\l_stex_current_symbol_uri`,
- `\l_stex_current_symbol_arity_int`,
- `\l_stex_current_symbol_args_tl`,
- definiens (currently not used),
- `\l_stex_current_symbol_type_tl`,
- `\l_stex_current_symbol_return_tl`,

- the invokation macro (is called after setup).

```
\cs_new_protected:Nn \_stex_invoke_symbol:nnnnnnN {
  \bool_if:NTF \l_stex_allow_semantic_bool{
    \group_begin:
    \tl_set:Nn \l_stex_current_symbol_uri {#1}
    \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_current_symbol_uri
    \tl_set:Nn \l_stex_current_display_tl {
      \stex_symbol_uri_name:N \l_stex_current_symbol_uri
    }
    \__stex_expr_invoke:nnnnN{#2}{#3}{#5}{#6}{#7}
  }{
    \msg_error:nnxx{stex}{error/notallowed}{
      \stex_use_symbol_uri:n{#1}
    }{
      \l_stex_current_full_tl
    }
  }
}

\cs_new_protected:Npn \_stex_invoke_symbol:NnnnnnN {
  \exp_args:No \_stex_invoke_symbol:nnnnnnN
}
\cs_generate_variant:Nn \_stex_invoke_symbol:NnnnnnN {NeooooN}
```

Whether semantic macros are allowed currently. This is false e.g. in the body of a semantic macro outside of one of its arguments:

```
\bool_new:N \l_stex_allow_semantic_bool
\bool_set_true:N \l_stex_allow_semantic_bool
```

Code to execute every time a semantic macro is invoked:

```
\tl_set:Nn \l_stex_every_symbol_tl {
  \bool_set_false:N \l_stex_allow_semantic_bool
}
```

The current top-level term; may be undefined:

```
\tl_new:N \l_stex_current_term_tl
```

Keeps track of all set up so it can be reexecuted; this may be necessary to e.g. typeset `\this` in a struct field:

```
\tl_new:N \l_stex_current_redo_tl
```

Setup for symbols; takes: arity, args, type, return, invoke. Calls invoke at the end.

In HTML mode, we make sure we enter horizontal mode *before* any annotations are inserted

```
\cs_new_protected:Nn \__stex_expr_invoke:nnnnN {
  \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
  \__stex_expr_setup:nnnnn{\_comp}{#1}{#2}{#4}{#3}
  \cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_oms:nn
  \tl_put_right:Nn \l_stex_current_redo_tl{
    \cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_oms:nn
  }
  #5
}
```

general setup; takes: comp, arity, args, return, type

```
\cs_new_protected:Nn \__stex_expr_setup:nnnnn {
  \tl_clear:N \l_stex_return_notation_tl
  \tl_set:Nn \l_stex_current_redo_tl {
    \let \this \stex_current_this:
    \def\comp{#1}
    \def\maincomp{\comp}
    \str_set:Nn \l_stex_current_arity_str{ #2 }
    \tl_set:Nn \l_stex_current_args_tl{ #3 }
    \tl_set:Nn \l_stex_current_return_tl{ #4 }
    \tl_set:Nn \l_stex_current_type_tl{ #5 }
    \tl_clear:N \l_stex_current_term_tl
  }
  \tl_put_right:No \l_stex_current_redo_tl \l_stex_every_symbol_tl
  \tl_put_left:Ne \l_stex_current_redo_tl {
    \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
  }
  \l_stex_current_redo_tl
}
```

(*End of definition for* `\_stex_invoke_symbol:NnnnnnN` *and* `\_stex_invoke_symbol:nnnnnnN`. *These functions are documented on page 149.*)

sfunction

`\stex_invoke_symbol:` Math or text mode?

```
\cs_new_protected:Nn \stex_invoke_symbol: {
  \stex_debug:nn{expressions}{Invoking~ \l_stex_current_full_tl}
  \mode_if_math:TF \__stex_expr_math: \__stex_expr_text:
}
```

(*End of definition for* `\stex_invoke_symbol:`. *This function is documented on page 149.*)

sfunction

`\stex_invoke_text_symbol:`

```
\cs_new_protected:Nn \stex_invoke_text_symbol: {
  \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
  \_stex_term_oms_or_omv:nn{}{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
  \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
}
```

(*End of definition for* `\stex_invoke_text_symbol:`. *This function is documented on page 149.*)

sfunction

`\stex_invoke_sequence:`

```
\cs_new_protected:Nn \stex_invoke_sequence: {
  \peek_charcode_remove:NTF ! {
    \peek_charcode:NTF [ \__stex_expr_seq_op:w { \__stex_expr_seq_op:w [] }
  }\__stex_expr_seq_first:
}

\cs_new_protected:Npn \__stex_expr_seq_op:w [#1] {
  \stex_use_op_notation:nnTF \l_stex_current_full_tl{#1}{
    %\stex_debug:nn{vars}{Here: \meaning\l_stex_notation_cs}
    \_stex_maybe_brackets:nn{\neginfprec}{
```

```
      \_stex_term_oms_or_omv:nn{#1}
      {\l_stex_notation_cs}\group_end:
    }
  }{
    \__stex_expr_get_index_notation:n{#1}
    \peek_charcode:NTF [ \__stex_expr_range:w { \__stex_expr_range:w[] }
  }
}

\cs_new_protected:Nn \__stex_expr_get_index_notation:n {
  \stex_use_notation:nnTF \l_stex_current_full_tl{#1}{}{
    \stex_do_default_notation:
    \cs_set_eq:NN \l_stex_notation_cs \l_stex_default_notation
  }
}

\cs_new_protected:Npn \__stex_expr_range:w [#1] {
  \bool_set_true:N \l_stex_allow_semantic_bool
  \clist_clear:N \l__stex_expr_clist
  \clist_map_function:NN \l_stex_current_type_tl \__stex_expr_seq_arg:n
    \stex_annotate:nn{
      data-ftml-term=OMV,
      data-ftml-head={\l_stex_current_full_tl},
      data-ftml-notationid={}
    }{
      \l__stex_expr_clist
    }
    \group_end:
}

\cs_new_protected:Nn \__stex_expr_seq_arg:n {
  \tl_if_eq:nnTF{#1}{\ellipses}{
    \clist_put_right:Nn \l__stex_expr_clist {
        \ellipses
    }
  }{
    \clist_put_right:Nn \l__stex_expr_clist {
      \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1}{
        \group_begin:
          \l_stex_notation_cs \group_end: {#1}

      }{
        \group_begin:
          \l_stex_notation_cs \group_end: #1
      }
    }
  }
}

\cs_new_protected:Nn \__stex_expr_seq_first: {
  \exp_args:Nne \use:nn{
  \cs_generate_from_arg_count:NNnn \l_stex_notation_cs \cs_set:Npn
  \l_stex_current_arity_str} {{
    \tl_set:Nn \exp_not:N \l__stex_expr_first_args_tl {
```

```
      \int_step_function:nN \l_stex_current_arity_str \__stex_expr_do_first_arg:n
    }
    \exp_not:N \__stex_expr_do_first_next:
  }}
  \l_stex_notation_cs
}

\cs_new:Nn \__stex_expr_do_first_arg:n {{\exp_not:n{## #1}}}

\cs_new_protected:Nn \__stex_expr_do_first_next: {
  \peek_charcode_remove:NTF ! {
    \peek_charcode:NTF [ \__stex_expr_do_one:w {\__stex_expr_do_one:w []}
  }{
    \peek_charcode:NTF [ \__stex_expr_do_all:w {\__stex_expr_do_all:w []}
  }
}

\cs_new_protected:Npn \__stex_expr_do_one:w [#1] {
  \__stex_expr_get_index_notation:n{#1}
  \exp_args:Nno\use:nn{\l_stex_notation_cs\group_end:}\l__stex_expr_first_args_tl
}

\cs_new_protected:Npn \__stex_expr_do_all:w [#1] {
  \exp_args:Nno\use:nn{\__stex_expr_notation:w [#1]}\l__stex_expr_first_args_tl
}
```

(*End of definition for* `\stex_invoke_sequence:`. *This function is documented on page 150.*)

sfunction

`\stex_invoke_structure:`

```
\cs_new_protected:Nn \stex_invoke_structure: {
  \tl_set:Nn \l__stex_expr_set_comp_tl {\__stex_expr_set_thiscomp:}
  \__stex_expr_struct_top:n {}
}

\cs_new_protected:Nn \__stex_expr_set_thiscomp: {
  \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
    \edef\maincomp {\_thiscomp{\comp}}
  }
}

\cs_new_protected:Nn \__stex_expr_struct_top:n {
  \stex_debug:nn{structure}{
    invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
  }
  \peek_charcode:NTF [ {
    \__stex_expr_merge:nw{#1}
  }{
    \__stex_expr_struct_type:n {#1}
    \tl_set:Nn \l_stex_struct_this_tl {}
    \peek_charcode_remove:NTF ! {
      \peek_charcode:NTF [ {
        \__stex_expr_maybe_notation:w
      }{
```

```
        \__stex_expr_maybe_notation:w []
      }
    }{
      \__stex_expr_invoke_this:n
    }
  }
}

\cs_new_protected:Npn \__stex_expr_merge:nw #1 [ #2 ] {
  \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{comp=}{
    \__stex_expr_set_customcomp: #2 \__stex_expr_end:
    \__stex_expr_struct_top:n{#1}
  }{
    \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{this=}{
      \__stex_expr_set_thisnotation: #2 \__stex_expr_end:
      \__stex_expr_struct_top:n{#1}
    }{
      \tl_if_empty:nTF{#1}{
        \__stex_expr_struct_top:n{#2}
      }{
        \tl_if_empty:nTF{#2}{
          \__stex_expr_struct_top:n{#1}
        }{
          \__stex_expr_struct_top:n{#1,#2}
        }
      }
    }
  }
}

\cs_new_protected:Npn \__stex_expr_set_thisnotation: this= #1 \__stex_expr_end: {
  \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
  \tl_set:Nn \l__stex_expr_set_comp_tl {}
}

\cs_new_protected:Npn \__stex_expr_set_customcomp: comp= #1 \__stex_expr_end: {
  \tl_set:Nn \l__stex_expr_set_comp_tl {
    \__stex_expr_set_custom_comp:n{#1}
  }
  \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
}
```

The structure type:

```
\cs_new_protected:Nn \__stex_expr_struct_type:n {
  \__stex_expr_do_assign_list:n{#1}
  \clist_if_empty:NTF \l__stex_expr_fields_clist {
    \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
      = 1 {
        \tl_set:Ne \l__stex_expr_current_type_tl {
          \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl }
          \exp_args:No \exp_not:n \l_stex_current_redo_tl
          \_stex_term_oms_or_omv:nn{}{}
        }
      }{
        \exp_args:No \__stex_expr_make_type:n \l_stex_current_type_tl
```

```
    }
  }{
    \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
      = 1 {
        \__stex_expr_make_type:n {}
      }{
        \exp_args:No \__stex_expr_make_type:n \l_stex_current_type_tl
    }
  }
}

\cs_new_protected:Nn \__stex_expr_do_assign_list:n {
  \clist_clear:N \l__stex_expr_fields_clist
  \tl_if_empty:nF {#1} {
    \keyval_parse:NNn\TODO\__stex_expr_do_assign:nn{#1}
  }
}

\cs_new_protected:Nn \__stex_expr_do_assign:nn {
  \clist_put_right:Nn \l__stex_expr_fields_clist {{#1}{#2}}
}

\cs_new_protected:Nn \__stex_expr_make_type:n {
  \tl_if_empty:nTF{#1}{
    \seq_clear:N \l_tmpa_seq
  }{
    \seq_set_split:Nnn \l_tmpa_seq ,{#1}
    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
    \seq_reverse:N \l_tmpa_seq
  }
  \tl_set:Ne \l__stex_expr_current_type_tl {
    \symuse{Metatheory?module~type~merge}{
      {
        \exp_args:No \exp_not:n \l_stex_current_redo_tl
        \_stex_term_oms_or_omv:nn{}{}
      }
      \seq_map_function:NN \l_tmpa_seq \__stex_expr_make_mod:n
      \clist_if_empty:NF \l__stex_expr_fields_clist {
        ,\symuse{Metatheory?anonymous~record}{
          \exp_args:Ne \tl_tail:n{
            \clist_map_function:NN \l__stex_expr_fields_clist \__stex_expr_make_oml:n
          }
        }
      }
    }
  }
}

\cs_new:Nn \__stex_expr_make_mod:n {
  ,\symuse{Metatheory?module~type}{
    \exp_args:Ne \stex_annotate:nn{data-ftml-term=OMMOD,data-ftml-head={\stex_use_module_ur
  }
}
```

```
\cs_new:Nn \__stex_expr_make_oml:n {
  \__stex_expr_make_oml:nn #1
}
\cs_new:Nn \__stex_expr_make_oml:nn {
  ,\stex_annotate:nn{
    data-ftml-term=OML,
    data-ftml-head={#1}
  }{
    \_stex_annotate_force_break:n{
      \stex_annotate:nn{data-ftml-definiens={}}{\exp_not:n{#2!}}
    }
  }
}
```

Insert the structure type as a term:

```
\cs_new:Nn \__stex_expr_current_type: {
  %\exp_args:No \exp_not:n \l_stex_current_redo_tl
  \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
    \exp_args:No\exp_not:n\l__stex_expr_current_type_tl
  }
  \_stex_term_oms_or_omv:nn{}{}
}
```

The structure type itself:

```
\cs_new_protected:Npn \__stex_expr_maybe_notation:w [ #1 ] {
  \tl_set_eq:NN \l_stex_current_term_tl \l__stex_expr_current_type_tl
  \stex_use_notation:nnTF \l_stex_current_full_tl {#1}{
    \l_stex_notation_cs\group_end:
  }{
    \__stex_expr_make_prop:
    \__stex_expr_make_prop_assign:
    \__stex_expr_present_i:w [#1]
  }
}

\cs_new_protected:Nn \__stex_expr_present: {
  \peek_charcode:NTF [ {
    \__stex_expr_present_i:w
  }{
    \__stex_expr_present:nn{}{}
  }
}

\cs_new_protected:Npn \__stex_expr_present_i:w [#1] {
  \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
    \__stex_expr_present:nn{}{#1}
  }{
    \peek_charcode:NTF [ {
      \__stex_expr_present_ii:nw{#1}
    }{
      \__stex_expr_present:nn{#1}{}
    }
  }
}
```

```
%First: clist, second:notation-id
\cs_new_protected:Npn \__stex_expr_present_ii:nw #1 [#2] {
  \__stex_expr_present:nn{#1}{#2}
}

\cs_new_protected:Nn \__stex_expr_present:nn {
  \clist_clear:N \l__stex_expr_clist
  \tl_if_empty:nTF{#1}{
    \cs_set:Npn \l__stex_expr_cs ##1 ##2 ##3 {
      \tl_if_empty:nF{##2}{
        \__stex_expr_present_entry:nn {##1}{##3}
      }
    }
  }{
    \cs_set:Npn \l__stex_expr_cs ##1 ##2 ##3 {
      \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1}{
        \__stex_expr_present_entry:nn {##1}{##3}
      }
    }
  }
  \prop_map_inline:Nn \l__stex_expr_prop {
    \l__stex_expr_cs {##1} ##2
  }
  \_stex_term_oms_or_omv:nn{}{
    \exp_args:Nno \use:n{
      \bool_set_true:N \l_stex_allow_semantic_bool
      \symuse{Metatheory?mathematical~structure}[#2]
    }{\l__stex_expr_clist}
  }\group_end:
}

\cs_new_protected:Nn \__stex_expr_present_entry:nn {
  \seq_if_in:NnTF \l__stex_expr_assigned_seq {#1}{
    \clist_put_right:Nn \l__stex_expr_clist {#2!}
  }{
    \exp_args:NNe \clist_put_right:Nn \l__stex_expr_clist {
      \_stex_next_symbol:n {
        \exp_args:No \exp_not:n \l__stex_expr_set_comp_tl
        \tl_set:Nn \exp_not:N \l_stex_struct_this_tl {
          \exp_args:No \exp_not:n \l_stex_struct_this_tl
        }
        \exp_not:n {
          \tl_set_eq:NN \this \l_stex_struct_this_tl
        }
        \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
          \exp_args:No \exp_not:n \l_stex_return_notation_tl
        }
      }
      \exp_not:n{#2!}
    }
  }
}
```

```
%\cs_new_protected:Nn \__stex_expr_set_custom_comp:n {
%  \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
%    \cs_set_protected:Npx \_customthiscomp ##1 {
%      \group_begin:
%        \bool_set_true:N \l_stex_allow_semantic_bool
%        \exp_not:n{
%          \cs_set:Npn \l__stex_expr_comp_cs ##1  {
%            #1
%          }
%          \def\maincomp
%        }{\comp}
%        \exp_not:N\l__stex_expr_comp_cs{\comp{##1}}
%      \group_end:
%    }
%    \def\maincomp {\_customthiscomp}
%  }
%}

\cs_new_protected:Nn \__stex_expr_set_custom_comp:n {
  \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
    \stex_debug:nn{structs}{Setting~custom~comp:~\tl_to_str:n{#1}}
    \exp_args:Nne \__stex_expr_set_custom_i:nn{#1}{\comp}
    \def\maincomp {\_customthiscomp}
  }
}

\cs_new_protected:Nn \__stex_expr_set_custom_i:nn {
  \cs_set_protected:Npn \_customthiscomp ##1 {
    \group_begin:
      \bool_set_true:N \l_stex_allow_semantic_bool
      \cs_set:Npn \l__stex_expr_comp_cs ####1 { #1 }
      \def \maincomp {#2}
      \l__stex_expr_comp_cs{#2{##1}}
    \group_end:
  }
}
```

`this` (of type `structure`):

```
\cs_new_protected:Nn \__stex_expr_invoke_this:n {
  \peek_charcode_remove:NTF ! {
    \exp_args:Nne\use:nn{
      \group_end:\symuse{Metatheory?of~type}[invisible]{
        \tl_if_empty:nTF{#1}{\_stex_annotate_force_break:n{}}{#1}
      }
    }{
      {
        \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
        \__stex_expr_current_type:
      }
    }
  }{
    \__stex_expr_invoke_maybe_field:nn{#1}
  }
}
```

```
\cs_new_protected:Nn \__stex_expr_invoke_maybe_field:nn {
  \__stex_expr_make_prop:
  \__stex_expr_set_this:n{#1}
  \tl_if_empty:nTF{#2}{
    \__stex_expr_make_prop_assign:
    \__stex_expr_present:
  }{
    \__stex_expr_invoke_field:n{#2}
  }
}

\cs_new_protected:Nn \__stex_expr_set_this:n {
  \tl_if_empty:nTF{#1}{
    %\tl_put_right:Nn \l_stex_current_redo_tl {
    %  \tl_clear:N \l_stex_struct_this_tl
    %}
  }{
    \tl_set:Ne \l_stex_struct_this_tl {{
      \bool_set_true:N \l_stex_allow_semantic_bool
      \bool_set_true:N \l_stex_in_this_bool
      \tl_set:Nn \exp_not:N \this {
        \exp_args:No \exp_not:n \this
      }
      \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl }
      \exp_not:n{#1}
    }}
    \tl_set_eq:NN \this \l_stex_struct_this_tl
    % \l_stex_return_notation_tl
  }
}

\cs_new_protected:Nn \__stex_expr_get_field_name:n {
  \str_set:Ne \l__stex_expr_field_name_str {
    \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
    {\prop_item:Nn \l__stex_expr_prop {#1}}
  }
  \str_if_empty:NT \l__stex_expr_field_name_str {
    \str_set:Nn \l__stex_expr_field_name_str {#1}
  }
}

\cs_new_protected:Nn \__stex_expr_invoke_field:n {
  \prop_if_in:NnTF \l__stex_expr_prop {#1}{
    \__stex_expr_get_field_name:n{#1}
    \tl_clear:N \l__stex_expr_more_nextsymbol_tl
    %\exp_args:NNe \seq_if_in:NnF \l__stex_expr_assigned_seq {\tl_to_str:n{#1}}{
      \tl_set:Ne \l__stex_expr_more_nextsymbol_tl {
        \tl_set:Nn \exp_not:N \l_stex_struct_this_tl {
          \exp_args:No \exp_not:n \l_stex_struct_this_tl
        }
        \exp_not:n {
          \tl_set_eq:NN \this \l_stex_struct_this_tl
        }
        \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
```

```
          \exp_args:No \exp_not:n \l_stex_return_notation_tl
        }
        \exp_args:No \exp_not:n \l__stex_expr_set_comp_tl
      }
    %}
    \exp_args:NNe \use:nn \group_end: {
      \_stex_next_symbol:n {
        \exp_args:No \exp_not:n \l__stex_expr_redo_tl
        \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
          \symuse{Metatheory?record~field}{
            \symuse{Metatheory?of~type}{
              \exp_args:No\tl_if_empty:nTF \l_stex_struct_this_tl {\_stex_annotate_force_br
            }{
              \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
              \__stex_expr_current_type:
            }
          }{
            \stex_annotate:nn{data-ftml-term=OML,data-ftml-head={\l__stex_expr_field_name_s
          }
        }
        \exp_args:No \exp_not:n \l__stex_expr_more_nextsymbol_tl
      }
      \exp_not:N \use_ii:nn
      \prop_item:Nn \l__stex_expr_prop {#1}
    }
  }{
    \msg_error:nnn{stex}{error/unknownfield}{#1}
  }
}

\cs_new_protected:Nn \__stex_expr_make_prop: {
  \prop_clear:N \l__stex_expr_prop
  \seq_clear:N \l__stex_expr_seq
  \seq_clear:N \l__stex_expr_assigned_seq
  \tl_clear:N \l__stex_expr_redo_tl
  \__stex_expr_prop_do_decls:
}

\cs_new_protected:Nn \__stex_expr_prop_do_decls: {
  \group_begin:
    \stex_debug:nn{expressions}{constructing~record}
    \seq_clear:N \l_stex_all_module_seq
    \exp_args:Ne \stex_activate_module:n {\clist_item:Nn \l_stex_current_type_tl 1}
    \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
      % uri, macro name, name, arity, args, type, def, return, macro
      \tl_if_empty:nTF{##2}{
        \exp_args:Nne\__stex_expr_do_decl_nomacro:nnnnnnnnn{##3}
      }{
        \exp_args:Nne\__stex_expr_do_decl:nnnnnnnnn{##2}
      }
      {\stex_new_symbol_uri:nn{##1}{##3}}{##3}{##4}{##5}{##6}{##7}{##8}##9
    }
    \exp_after:wN \tl_set:Nn \exp_after:wN \l__stex_expr_after_tl \exp_after:wN {
      \exp_after:wN \tl_set:Nn \exp_after:wN \l__stex_expr_prop \exp_after:wN { \l__stex_ex
```

```
    }
    \__stex_expr_prop_do_notations:
    \stex_debug:nn{expressions}{record:~\meaning\l__stex_expr_after_tl}
    \tl_put_right:Ne \l__stex_expr_after_tl {\tl_set:Nn \exp_not:N \l__stex_expr_redo_tl {\
  \exp_after:wN \group_end: \l__stex_expr_after_tl
}

% id, uri, name, arity, args, type, def, return, macro
\cs_new_protected:Nn \__stex_expr_do_decl_nomacro:nnnnnnnnn {
  \prop_if_in:NnF \l__stex_expr_prop {#1} {
    \seq_put_left:Ne \l__stex_expr_seq {\tl_to_str:n{#2}}
    \prop_put:Nnn \l__stex_expr_prop {#1}{
      {}{
        \_stex_invoke_symbol:nnnnnnN
        {#2}
        {#4}
        {#5}{#6}{#7}{#8}#9
      }
    }
  }
}

\cs_new_protected:Nn \__stex_expr_do_decl:nnnnnnnnn {
  \prop_if_in:NnF \l__stex_expr_prop {#1} {
    \seq_put_left:Ne \l__stex_expr_seq {\tl_to_str:n{#2}}
    \prop_put:Nnn \l__stex_expr_prop {#1}{
      {#3}{
        \_stex_invoke_symbol:nnnnnnN
        {#2}
        {#4}
        {#5}{#6}{#7}{#8}#9
      }
    }
  }
}

\cs_new_protected:Nn \__stex_expr_make_prop_assign: {
  \clist_if_empty:NF \l__stex_expr_fields_clist {
    \clist_map_inline:Nn \l__stex_expr_fields_clist {
      \__stex_expr_make_prop_assign:nn ##1
    }
  }
}

\cs_new_protected:Nn \__stex_expr_make_prop_assign:nn {
  \prop_if_in:NnTF \l__stex_expr_prop {#1}{
    \exp_args:NNe \seq_put_right:Nn \l__stex_expr_assigned_seq {\tl_to_str:n{#1}}
    \exp_args:Nne \use:nn {\__stex_expr_make_prop_assign_replace:nnnn {#1}{#2}}
    {\prop_item:Nn \l__stex_expr_prop {#1}}
  }{
    \msg_error:nnn{stex}{error/unknownfieldass}{#1}
  }
}
\cs_new_protected:Nn \__stex_expr_make_prop_assign_replace:nnnn {
```

```
  \prop_put:Nnn \l__stex_expr_prop {#1}{{#3}{#2}}
  \tl_if_empty:nF{#3}{
    \tl_set:cn{#1}{ #2 }
    \tl_put_right:Nn \l__stex_expr_redo_tl {
      \tl_set:cn{#1}{ #2 }
    }
  }
}

\cs_new_protected:Nn \__stex_expr_prop_do_notations: {
  \exp_args:No \stex_iterate_notations:nn \l_stex_current_type_tl {
    \exp_args:NNe \seq_if_in:NnT \l__stex_expr_seq {\tl_to_str:n{##1}}{
      \tl_set:Nn \l_tmpa_tl {
        \cs_if_exist:cF{l_stex_notation_ \stex_use_symbol_uri:n{##1}?##2_cs}{
          \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?##2_cs}{##4}
        }
        \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?_cs}{
          \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?_cs}{##4}
        }
      }
      \exp_args:NNo \tl_put_right:Nn \l__stex_expr_redo_tl \l_tmpa_tl
      \exp_args:NNo \tl_put_right:Nn \l__stex_expr_after_tl \l_tmpa_tl
      \tl_if_empty:nF{##5}{
        \tl_set:Nn \l_tmpa_tl {
          \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?##2_op_cs}{
            \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?##2_op_cs}{##5}
          }
          \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?_op_cs}{
            \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?_op_cs}{##5}
          }
        }
        \exp_args:NNo \tl_put_right:Nn \l__stex_expr_redo_tl \l_tmpa_tl
        \exp_args:NNo \tl_put_right:Nn \l__stex_expr_after_tl \l_tmpa_tl
      }
    }
  }
}
```

(*End of definition for* `\stex_invoke_structure:`. *This function is documented on page 150.*)

sfunction

`\_stex_invoke_variable:nnnnnn`

```
\cs_new_protected:Nn \_stex_invoke_variable:nnnnnnN {
  \bool_if:NTF \l_stex_allow_semantic_bool{
    \group_begin:
    \tl_set:Nn \l_stex_current_full_tl { #1 }
    \tl_set:Nn \l_stex_current_display_tl { #1 }
    \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
    \__stex_expr_setup:nnnnn{\_varcomp}{#2}{#3}{#6}{#5}
    \cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_omv:nn
    \tl_put_right:Nn \l_stex_current_redo_tl {
      \cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_omv:nn
    }
    #7
```

```
  }{
    \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
  }
}
```

(*End of definition for* `\_stex_invoke_variable:nnnnnn`. *This function is documented on page 150.*)

sfunction

`\svar`

```
\NewDocumentCommand \svar {O{} m}{
  \group_begin:
    \tl_if_empty:nTF{#1}{
      \tl_set:Nn \l_stex_current_full_tl { #2 }
      \tl_set:Nn \l_stex_current_display_tl { #2 }
    }{
      \tl_set:Nn \l_stex_current_full_tl { #1 }
      \tl_set:Nn \l_stex_current_display_tl { #1 }
    }
    \bool_if:NTF \l_stex_allow_semantic_bool{
        \tl_clear:N \l_stex_current_term_tl
        \_stex_term_omv:nn{}{\_varcomp{#2}}
    }{
      \msg_error:nnxx{stex}{error/notallowed}{Variable}{\l_stex_current_full_tl}
    }
  \group_end:
}
```

(*End of definition for* `\svar`. *This function is documented on page 150.*)

sfunction

Now for the brunt of the work:

### Math

Modifiers (`!` or `*`)?

```
\cs_new_protected:Nn \__stex_expr_math: {
  \stex_debug:nn{expressions}{math~mode}
  \peek_charcode_remove:NTF ! {
    % operator
    \peek_charcode_remove:NTF * \__stex_expr_op_custom:n {
      % op notation
      \peek_charcode:NTF [ \__stex_expr_op_notation:w {
        \__stex_expr_op_notation:w []
      }
    }
  }{
    \peek_charcode_remove:NTF * \__stex_expr_custom:n {
      % normal
      \peek_charcode:NTF [ \__stex_expr_notation:w {
        \__stex_expr_notation:w []
      }
    }
  }
}
```

**Text**

Operator?

```
\cs_new_protected:Nn \__stex_expr_text: {
  \stex_debug:nn{expressions}{text~mode}
  \peek_charcode_remove:NTF ! \__stex_expr_op_custom:n \__stex_expr_custom:n
}
```

**Notation**

Operator?

```
\cs_new_protected:Npn \__stex_expr_notation:w [ #1 ] {
  \peek_charcode_remove:NTF ! {
    \__stex_expr_op_notation:w [#1]
  }{
    \__stex_expr_full_notation:n {#1}
  }
}
```

(Possibly) complex notation taking arguments with optional particular identifier:

```
\cs_new_protected:Nn \__stex_expr_full_notation:n {
  \stex_use_notation:nnTF \l_stex_current_full_tl {#1}{
    \stex_debug:nn{expressions}{using~notation~"#1"~for~\l_stex_current_full_tl}
    \tl_if_empty:NTF \l_stex_current_return_tl {
      \stex_debug:nn{expressions}{return~empty}
      \l_stex_notation_cs{\group_end:\_stex_eat_exclamation_point:}
    }{
      \stex_debug:nn{expressions}{return?}
      \exp_after:wN \__stex_expr_maybe_return:n \exp_after:wN {
        \l_stex_notation_cs{}
      }
    }
  }{
    \stex_debug:nn{expressions}{notation~"#1"~for~\l_stex_current_full_tl{}~does~not~exist;
    \stex_do_default_notation:
    \tl_if_empty:NTF \l_stex_current_return_tl {
      \l_stex_default_notation{\group_end:\_stex_eat_exclamation_point:}
    }{
      \exp_after:wN
      \__stex_expr_maybe_return:n
      \exp_after:wN
      {\l_stex_default_notation {}}
    }
  }
}
```

Operator notation:

```
\cs_new_protected:Npn \__stex_expr_op_notation:w [#1] {
  \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_full_tl}
  \stex_use_op_notation:nnTF \l_stex_current_full_tl {#1}{
    \_stex_maybe_brackets:nn{\neginfprec}{
      \_stex_term_oms_or_omv:nn{#1}
      {\l_stex_notation_cs}
    }
    \group_end:
  }{
```

```
    \int_compare:nNnTF \l_stex_current_arity_str = 0 {
      \tl_clear:N \l_stex_current_return_tl
      \__stex_expr_notation:w [#1]
    }{
      \stex_do_default_notation_op:
      \_stex_maybe_brackets:nn{\neginfprec}{
        \_stex_term_oms_or_omv:nn{#1}
        {\l_stex_default_notation}
      }
      \group_end:
    }
  }
}
```

### Custom Notations

Operator:

```
\cs_new_protected:Nn \__stex_expr_op_custom:n {
  \stex_debug:nn{expressions}{custom~op}
  \bool_set_true:N \l_stex_allow_semantic_bool
  \_stex_term_oms_or_omv:nn{}{\maincomp{#1}}
  \group_end:
}
```

Complex:

```
\int_new:N \l__stex_expr_arg_counter_int

\cs_new_protected:Nn \__stex_expr_custom:n {
  \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_full_tl}
  \stex_pseudogroup:nn{
    \bool_set_true:N \l_stex_allow_semantic_bool
    \prop_gclear:N \l__stex_expr_customs_prop
    \seq_gclear:N \l__stex_expr_customs_seq
    \int_gzero:N \l__stex_expr_arg_counter_int
    \tl_if_empty:NF \l_stex_current_args_tl {
      \exp_after:wN \__stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \_stex_args_end:
      \cs_set_eq:NN \arg \__stex_expr_arg:n
    }
    \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
    \cs_set_eq:NN \__stex_expr_do_ab_next:nn \_stex_term_oma:nn
    \_stex_map_args:N \__stex_expr_check_b:nn
    \__stex_expr_do_ab_next:nn{}{#1}
  }{
    \prop_if_exist:NT \l__stex_expr_customs_prop {
      \prop_gset_from_keyval:Nn \exp_not:N \l__stex_expr_customs_prop {
        \prop_to_keyval:N \l__stex_expr_customs_prop
      }
    }
    \int_gset:Nn \l__stex_expr_arg_counter_int { \int_use:N \l__stex_expr_arg_counter_int}
    \seq_if_exist:NT \l__stex_expr_customs_seq {
      \seq_gset_split:Nnn \exp_not:N \l__stex_expr_customs_seq , {
        \seq_use:Nn \l__stex_expr_customs_seq ,
      }
    }
  }
```

```
  % TODO check that all arguments are present
  \group_end:
}

\cs_new_protected:Npn \__stex_expr_add_prop_arg:nnw #1 #2 #3\_stex_args_end: {
  \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {}
  \seq_gput_right:Nn \l__stex_expr_customs_seq {#2}
  \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nnw #3 \_stex_args_end:}
}

\cs_new:Nn \__stex_expr_check_b:nn {
  \str_case:nn #2 {
    b {\cs_set_eq:NN \__stex_expr_do_ab_next:nn \_stex_term_omb:nn}
    B {\cs_set_eq:NN \__stex_expr_do_ab_next:nn \_stex_term_omb:nn}
  }
}
```

Arguments:

```
\NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
  \IfBooleanTF #1 {
    \stex_annotate_invisible:n{
      \__stex_expr_arg_inner:nn{#2}{#3}
    }
  }{
    \__stex_expr_arg_inner:nn{#2}{#3}
  }
}

\cs_new_protected:Nn \__stex_expr_arg_inner:nn {
  \tl_if_empty:nTF{#1}{
    \int_gincr:N \l__stex_expr_arg_counter_int
    \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
      \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
    }{
      \__stex_expr_arg_inner:nn{}
    }{ #2 }
  }{
    \__stex_expr_check:nTF {#1}{
      \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
    }{
      \exp_args:No \str_case:nnTF \l_tmpb_tl {
        {a}{
          \exp_args:NNne \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
            \l_tmpa_tl X
          }
          \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
        }
        {B}{
          \exp_args:NNne \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
            \l_tmpa_tl X
          }
          \tl_set:Ne \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
        }
      }{
        \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
```

```
      }{
        \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_full_tl}
      }
    }
  }
}

\prg_new_conditional:Nnn \__stex_expr_check:n {TF} {
  \exp_args:NNe \prop_get:NnNTF \l__stex_expr_customs_prop {#1} \l_tmpa_tl {
    \tl_set:Ne \l_tmpb_tl {\seq_item:Nn \l__stex_expr_customs_seq {#1} }
    \tl_if_empty:NTF \l_tmpa_tl {
      \exp_args:NNe \prop_gput:Nnn \l__stex_expr_customs_prop
        { #1 }{X}
      \exp_args:No \str_case:nnF \l_tmpb_tl {
        {a}{
          \tl_set:Ne \l_tmpa_tl{ #1 1 }
        }
        {B}{
          \tl_set:Ne \l_tmpa_tl{ #1 1 }
        }
      }{
        \tl_set:Ne \l_tmpa_tl{ #1 }
      }
      \prg_return_true:
    }{
      \prg_return_false:
    }
  }{
    \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_full_tl}
    \prg_return_false:
  }
}

% #1 argnum #2 argmode #3 code
\cs_new_protected:Nn \__stex_expr_arg_do:nnn {
  \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
  \group_begin:
    \bool_set_true:N \l_stex_allow_semantic_bool
    \_stex_term_arg:nnn {#2}{#1}{#3}
  \group_end:
}
\cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}
```

### Return code

```
\cs_new_protected:Nn \__stex_expr_maybe_return:n {
  \tl_set:Nn \l__stex_expr_return_this_tl {#1}
  \tl_clear:N \l__stex_expr_return_args_tl
  \exp_args:Nne \use:n {
  \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs:
    \cs_set:Npn \l_stex_current_arity_str } {
      \int_step_function:nN \l_stex_current_arity_str \__stex_expr_return_arg:n
      \__stex_expr_return_next:
    }
```

```
  \__stex_expr_ret_cs:
}

\cs_new:Nn \__stex_expr_return_arg:n {
  \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{## #1}}
}

\cs_new_protected:Nn \__stex_expr_return_next: {
  \peek_charcode_remove:NTF ! {
    \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
  }\__stex_expr_return:
}

\cs_new_protected:Nn \__stex_expr_return: {
  \tl_set:Ne \l__stex_expr_return_this_tl {
    \tl_if_empty:NTF \l_stex_return_notation_tl {
      \exp_after:wN \exp_after:wN \exp_after:wN
      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
        \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
      }
    }{
      \l_stex_return_notation_tl
    }
  }
  \stex_debug:nn{return}{Notation:~\meaning\l__stex_expr_return_this_tl}

  \exp_after:wN
  \tl_put_left:Nn \exp_after:wN \l__stex_expr_return_this_tl \exp_after:wN {
    \exp_after:wN \group_begin: \l_stex_current_redo_tl
  }
  \exp_args:Nne \use:n {
  \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs:
    \cs_set:Npn \l_stex_current_arity_str } {
      \exp_args:No \exp_not:n \l_stex_current_return_tl
    }
  \stex_debug:nn{return}{
    \meaning\__stex_expr_ret_cs:^^J
    \meaning\l__stex_expr_return_this_tl^^J
    \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^^J
  }
  \exp_args:Nne \use:nn {
    \exp_after:wN \group_end: \__stex_expr_ret_cs:
  }{
    \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
    {
      \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
      \group_end:
    }
  }
}
```

### 32.4.1 Term Annotations

_stex_eat_exclamation_point:

```
\cs_new_protected:Nn \_stex_eat_exclamation_point: {
  \peek_charcode_remove:NT ! \_stex_eat_exclamation_point:
}
```

(*End of definition for* `\_stex_eat_exclamation_point:`. *This function is documented on page 150.*)

sfunction

We occasionally allow for semantic macros where they otherwise shouldn't be, if we are in "invisible" parts:

```
\bool_new:N \stex_in_invisible_html_bool
```

_stex_term_oms:nn

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_stex_term_oms:nn {
    \bool_if:NTF\l_stex_in_this_bool{#2}{
      \__stex_expr_check_comp:
      \tl_if_empty:NTF \l_stex_current_term_tl {
        \__stex_expr_annotate:nnn{OMID}{#1}{#2}
      }{
        \__stex_expr_do_headterm:nn{#1}{#2}
      }
    }
  }
}{
  \cs_new_protected:Nn \_stex_term_oms:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_chec
}

\cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_oms:nn
```

(*End of definition for* `\_stex_term_oms:nn`. *This function is documented on page 150.*)

sfunction

_stex_term_omv:nn

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_stex_term_omv:nn {
    \bool_if:NTF\l_stex_in_this_bool{#2}{
      \__stex_expr_check_comp:
      \tl_if_empty:NTF \l_stex_current_term_tl {
        \__stex_expr_annotate:nnn{OMV}{#1}{#2}
      }{
        \__stex_expr_do_headterm:nn{#1}{#2}
      }
    }
  }
}{
  \cs_new_protected:Nn \_stex_term_omv:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_chec
}
```

(*End of definition for* `\_stex_term_omv:nn`. *This function is documented on page 150.*)

sfunction

In the case where the "symbol" (OMS or OMV) is the field of some structure or otherwise a complex expression, we have to insert the full term representing the operator:

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
    \bool_if:NTF \stex_in_invisible_html_bool {
      {\bool_set_true:N \l_stex_allow_semantic_bool
        \ensuremath{\l_stex_current_term_tl}
      }
    }{
      \__stex_expr_annotate:nnn{complex}{#1}{
        \stex_annotate_invisible:nn{data-ftml-headterm={}}{
          {\bool_set_true:N \l_stex_allow_semantic_bool
            \ensuremath{\l_stex_current_term_tl}
          }
        }
        #2
      }
    }
  }
}{
  \cs_new_protected:Nn \__stex_expr_do_headterm:nn { #2 }
}
```

_stex_term_oma:nn

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_stex_term_oma:nn {
    \bool_if:NTF\l_stex_in_this_bool{#2}{
      \__stex_expr_check_comp:
      \__stex_expr_annotate:nnn{OMA}{#1}{
        \tl_if_empty:NF \l_stex_current_term_tl {
          \stex_annotate_invisible:nn{data-ftml-headterm={}}{{
            \bool_set_true:N \l_stex_allow_semantic_bool
            \l_stex_current_term_tl
          }}
        }
        #2
      }
    }
  }
}{
  \cs_new_protected:Nn \_stex_term_oma:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_chec
}
```

(*End of definition for* `\_stex_term_oma:nn`. *This function is documented on page 150.*)

sfunction

_stex_term_omb:nn

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_stex_term_omb:nn {
    \bool_if:NTF\l_stex_in_this_bool{#2}{
      \__stex_expr_check_comp:
      \__stex_expr_annotate:nnn{OMBIND}{#1}{
        \tl_if_empty:NF \l_stex_current_term_tl {
          \stex_annotate_invisible:nn{data-ftml-headterm={}}{{
            \bool_set_true:N \l_stex_allow_semantic_bool
            \l_stex_current_term_tl
```

```
            }}
          }
          #2
        }
      }
    }
  }{
    \cs_new_protected:Nn \_stex_term_omb:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_chec
  }
```

(*End of definition for* `\_stex_term_omb:nn`. *This function is documented on page 150.*)

sfunction

Does the actual annotating: TODO: This shouldn't use `\l_stex_current_symbol_uri!`

```
% kind, notation id, body
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \__stex_expr_annotate:nnn {
    \exp_args:Ne \stex_annotate:nn{
      data-ftml-term=#1,
      data-ftml-head={\l_stex_current_full_tl},
      data-ftml-notationid={#2},
    }{
      \_stex_annotate_force_break:n{#3}
    }
  }
}{
  \cs_new_protected:Nn \__stex_expr_annotate:nnn { #3 }
}
```

### 32.4.2 Symbol Arguments

`\_stex_term_arg:nnnnn`

`#1` : argument number
`#2` : argument mode
`#3` : precedence
`#4` : argument name (WiP)
`#5` : code / body

```
\cs_new_protected:Nn \_stex_term_arg:nnnnn {
  \group_begin:
    \tl_clear:N \l_stex_current_symbol_uri
    \tl_clear:N \l_stex_current_full_tl
    \tl_clear:N \l_stex_current_display_tl
    \tl_clear:N \l_stex_current_term_tl
    \int_set:Nn \l_stex_notation_downprec { #3 }
    \bool_set_true:N \l_stex_allow_semantic_bool
    \_stex_term_arg:nnn {#2}{#1}{#5}
  \group_end:
}
```

(*End of definition for* `\_stex_term_arg:nnnnn`. *This function is documented on page 151.*)

sfunction

_stex_term_arg:nnn

```
\cs_new_protected:Nn \__stex_expr_check_comp: {
  \bool_if:NT \g_stex_in_comp_bool {
    \msg_error:nn{stex}{error/argincomp}
  }
}
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_stex_term_arg:nnn {
    \stex_if_do_html:TF{
      \bool_if:NTF\l_stex_in_this_bool{#3}{
        \_stex_annotate_force_break:n{
          \stex_annotate:nn{
            data-ftml-arg={#2}, data-ftml-argmode={#1}
          }{
            \_stex_annotate_force_break:n{}
            \__stex_expr_check_comp:
            #3
          }
        }
      }
    }{ #3 }
  }
}{
  \cs_new_protected:Nn \_stex_term_arg:nnn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_che
}
```

(*End of definition for* `\_stex_term_arg:nnn`. *This function is documented on page 151.*)

sfunction

_stex_term_arg_aB:nnnnn

The following macros fill up `\l_stex_aB_args_seq` and ultimately call `\_stex_term_-do_aB_clist:`. By default, this does:

```
\cs_new:Nn \__stex_expr_do_aB_clist: {
  \_stex_annotate_force_break:n{
    \seq_use:Nn \l_stex_aB_args_seq {
      \mathpunct{\comp{,}}
    }
  }
}

\cs_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:

\cs_new_protected:Nn \_stex_is_sequentialized:n {
  \group_begin: #1 \group_end:
}
```

Setup:

```
\int_new:N \l__stex_expr_count_int
\cs_new_protected:Nn \_stex_term_arg_aB:nnnnn {
  \stex_debug:nn{sequences}{Processing~argument:~ \tl_to_str:n{#5}}
  \tl_if_empty:nTF{#5}{
    \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{}
  }{
    \seq_clear:N \l_stex_aB_args_seq
    \int_zero:N \l__stex_expr_count_int
```

```
    \clist_map_inline:nn{#5}{
      \__stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
    }
    \_stex_term_do_aB_clist:
  }
}
```

We “pattern match” the sequence argument - is it a comma-separated list? A sequence variable macro? \argmap? \seqmap?

```
% 1: code 2: argnum 3: argmode 4: precedence 5: argname
\cs_new_protected:Npn \__stex_expr_aB_arg:nnnnn #1 #2 #3 {
  \int_incr:N \l__stex_expr_count_int
  \stex_debug:nn{expressions}{matching~argument~ #2 #3:~ \tl_to_str:n{#1}}
  \__stex_expr_is_varseq:nTF{#1}{
    \stex_debug:nn{expressions}{=sequence~variable}
    \exp_after:wN \exp_after:wN \exp_after:wN
    \__stex_expr_assoc_seq:nnnnnnn
    \exp_after:wN
    \__stex_expr_gobble:nnnnnnnn #1 \__stex_expr_end:
  }{
    \__stex_expr_is_seqmap:nTF{#1}{
      \stex_debug:nn{expressions}{=seqmap}
      \exp_args:NNe \use:nn \__stex_expr_do_seqmap:nnnnnn {\tl_tail:n{#1}}
    }{
      \stex_debug:nn{expressions}{=simple}
      \__stex_expr_aB_simple_arg:nnnnn{#1}
    }
  }
  {#2}{#3}
}

\cs_new:Npn \__stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \__stex_expr_end: {
  {#2} #3 {#6}
}
```

Is it a varseq?

```
\prg_new_conditional:Nnn \__stex_expr_is_varseq:n {TF} {
  \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
    \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
    \_stex_invoke_variable:nnnnnnN {
      \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:nn{#1}{8}}
        \stex_invoke_sequence:
          \prg_return_true:\prg_return_false:
    }\prg_return_false:
  }\prg_return_false:
}
```

Is it a seqmap?

```
\prg_new_conditional:Nnn \__stex_expr_is_seqmap:n {TF} {
  \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
    \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
      \prg_return_true:\prg_return_false:
  }\prg_return_false:
}
```

Is it an argsep?

```
\prg_new_conditional:Nnn \__stex_expr_is_argsep:n {TF} {
  \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
    \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\argsep}
      \prg_return_true:\prg_return_false:
  }\prg_return_false:
}
```

Simple argument:

```
\cs_new_protected:Nn \__stex_expr_aB_simple_arg:nnnnn{
  \seq_put_right:Ne \l_stex_aB_args_seq {
    \_stex_term_arg:nnnnn{#2\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{
      \exp_not:n{
        \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
        #1
      }
    }
  }
}
```

Sequence variable:

```
% 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
\cs_new_protected:Nn \__stex_expr_assoc_seq:nnnnnnn {
  \group_begin:
    \seq_clear:N \l_stex_aB_args_seq
    \tl_set:Nn \l_stex_current_full_tl{#1}
    \tl_set:Nn \l_stex_current_display_tl{#1}
    \__stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
  \exp_args:NNe \use:nn \group_end: {
    \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
      \_stex_is_sequentialized:n{
        \_stex_term_arg:nnnnn{#4\int_use:N\l__stex_expr_count_int}{#5}{#6}{#7}{
          \bool_set_true:N \l_stex_allow_semantic_bool
          \tl_if_empty:NF \l_stex_current_term_tl {
            \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
              \exp_args:No \exp_not:n \l_stex_current_term_tl
            }
          }
          \stex_annotate:nn{
            data-ftml-term=OMV,
            data-ftml-head={#1},
            data-ftml-notationid={}
          }{
            \_stex_annotate_force_break:n{
              \_stex_term_do_aB_clist:
            }
          }
        }
      }
    }
  }
}

% #1: name, #2: clist, #3:arity
\cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
```

```
  \stex_use_notation:nnTF {#1}{}{
    \cs_set_eq:NN \l__stex_expr_cs \l_stex_notation_cs
  }{
    \stex_do_default_notation:
    \cs_set_eq:NN \l__stex_expr_cs \l_stex_default_notation
  }
  \clist_map_inline:nn{#2}{
    \tl_if_eq:nnTF{##1}{\ellipses}{
      \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
    }{
      \int_compare:nNnTF {#3} = 1 {
        \tl_set:Nn \l__stex_expr_iarg_tl { {##1} }
      }{
        \tl_set:Nn \l__stex_expr_iarg_tl { ##1 }
      }
      \seq_put_right:Ne \l_stex_aB_args_seq {
        \group_begin:
        \exp_not:n {
          \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
          \def\comp{\_varcomp}
          \tl_set:Nn \l_stex_current_full_tl{#1}
        }
        \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
        \exp_after:wN \exp_after:wN \exp_after:wN {
          \exp_after:wN \l__stex_expr_cs \exp_after:wN \group_end: \l__stex_expr_iarg_tl
        }
      }
    }
  }
}
```

seqmap:

```
% 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
\cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
  \group_begin:
    \cs_set:Npn \l_tmpa_cs ##1 {#1}
    \seq_clear:N \l_stex_aB_args_seq
    \clist_map_inline:nn{#2}{
      \__stex_expr_is_varseq:nTF{##1}{
        \exp_after:wN
        \__stex_expr_varseq_in_map:nnnnnnnn ##1
      }{
        \seq_put_right:Nn \l_stex_aB_args_seq {##1}
      }
    }
    \seq_clear:N \l__stex_expr_old_seq
    \seq_map_inline:Nn \l_stex_aB_args_seq {
      \tl_if_eq:nnTF{##1}{\ellipses}{
        \seq_put_right:Nn \l__stex_expr_old_seq {##1}
      }
      % TODO \_stex_is_sequentialized:n
      {
        \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
          \l_tmpa_cs {##1}
        }
```

```
      }
    }
    \seq_set_eq:NN \l_stex_aB_args_seq \l__stex_expr_old_seq
  \tl_set:Ne \l_tmpa_tl {
    \symuse{Metatheory?bind}{
      \svar[MAP_DUMMY]{\exp_not:N\mathcal{m}}
    }{
      \exp_args:No\exp_not:n{\l_tmpa_cs{\svar[MAP_DUMMY]{\mathcal{m}}}}
    }
  }
  \exp_args:NNe \use:nn \group_end: {
    \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
      \_stex_is_sequentialized:n{
        \_stex_term_arg:nnnnn{#3\int_use:N\l__stex_expr_count_int}{#4}{#5}{#6}{
          \bool_set_true:N \l_stex_allow_semantic_bool
          \tl_set:Nn \exp_not:N \l_stex_current_full_tl
            {\l_stex_current_full_tl}
          \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
            \exp_not:N \symuse{Metatheory?sequence~map}
            {\exp_not:n{#2}}
            {\exp_args:No \exp_not:n \l_tmpa_tl}
          }
          \__stex_expr_do_headterm:nn{}{
            \_stex_term_do_aB_clist:
          }
        }
      }
    }
  }
}

\cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnnn {
  \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
}
```

(*End of definition for* `\_stex_term_arg_aB:nnnnn`. *This function is documented on page 151.*)

sfunction

`\seqmap`

```
\cs_new_protected:Npn \seqmap #1 #2 {
  \symuse{Metatheory?sequence~expression}{\seqmap{#1}{#2}}
  %  \cs_set:Npn \l_tmpa_cs ##1 {#1}
  %  \tl_set:Ne \l_tmpa_tl {
  %    \symuse{Metatheory?bind}{
  %      \svar[MAP_DUMMY]{\exp_not:N\mathcal{m}}
  %    }{
  %      \exp_args:No\exp_not:n{\l_tmpa_cs{\svar[MAP_DUMMY]{\mathcal{m}}}}
  %    }
  %  }
  %  \__stex_expr_annotate:nnn{complex}{OMS}{
  %    \stex_annotate_invisible:nn{data-ftml-headterm={}}{
  %        {\bool_set_true:N \l_stex_allow_semantic_bool
  %          \ensuremath{
```

```
%             \exp_args:Nnno \symuse{Metatheory?sequence~map}{#2}\l_tmpa_tl
%           }
%         }
%    }
%    \symuse{Metatheory?sequence~expression}{\seqmap{#1}{#2}}
%  }
}
```

(*End of definition for* `\seqmap`. *This function is documented on page 151.*)

sfunction

### 32.4.3 Notation components

```
⟨@@=stex_comps⟩
```

`\comp`
`\compemph@uri`
`\compemph`

```
\cs_set_protected:Npn \comp {}

\cs_new_protected:Npn \compemph@uri #1 #2 {
    \compemph{ #1 }
}

\cs_new_protected:Npn \compemph #1 { #1 }

\cs_new_protected:Npn \_comp {
  \_do_comp:nnNn {comp}{}\compemph@uri
}

\cs_new_protected:Npn \_thiscomp #1 #2 {
  {\tl_set:cn{this}{{}}#1{#2}\c_math_subscript_token{
    \group_begin:
    \bool_set_true:N \l_stex_allow_semantic_bool
    \bool_set_true:N \l_stex_in_this_bool
    \l_stex_struct_this_tl
    \group_end:
  }
  }
}

\def\maincomp{\comp}
```

(*End of definition for* `\comp`, `\compemph@uri`, *and* `\compemph`. *These functions are documented on page 151.*)

sfunction

`\varemph@uri`
`\varemph`

```
\cs_new_protected:Npn \varemph@uri #1 #2 {
    \varemph{#1}
}

\cs_new_protected:Npn \varemph #1 { #1 }

\cs_new_protected:Npn \_varcomp {
  \_do_comp:nnNn {comp}{}\varemph@uri
}
```

(*End of definition for* `\varemph@uri` *and* `\varemph`. *These functions are documented on page 151.*)

sfunction

`\defemph@uri`
`\defemph`

```
\cs_new_protected:Npn \defemph@uri #1 #2 {
    \defemph{#1}
}

\cs_new_protected:Npn \defemph #1 {
    \ifmmode\else\expandafter\textbf\fi{#1}
}

\cs_new_protected:Npn \_defcomp {
  \_do_comp:nnNn {defcomp}{}\defemph@uri
}
```

(*End of definition for* `\defemph@uri` *and* `\defemph`. *These functions are documented on page 151.*)

sfunction

`\symrefemph@uri`
`\symrefemph`

```
\cs_new_protected:Npn \symrefemph@uri #1 #2 {
    \symrefemph{#1}
}

\cs_new_protected:Npn \symrefemph #1 { \emph{#1} }
```

(*End of definition for* `\symrefemph@uri` *and* `\symrefemph`. *These functions are documented on page 151.*)

sfunction

The following commands do the actual attributes:

`\_do_comp:nnNn`

```
\bool_new:N \g_stex_in_comp_bool

\cs_new_protected:Nn \_do_comp:nnNn {
  \stex_pseudogroup_with:nn{\comp}{
    \def\comp##1{##1}
    \bool_set_true:N \g_stex_in_comp_bool
    \stex_if_html_backend:TF {
      \stex_annotate:nn { data-ftml-#1 = {#2}}{ #4 }
    }{
      \tl_if_empty:NTF \l_stex_current_full_tl {
        #4
      }{
        #3 { #4 } \l_stex_current_full_tl

      }
    }
    \bool_set_false:N \g_stex_in_comp_bool
  }
}
```

(*End of definition for* `\_do_comp:nnNn`. *This function is documented on page 151.*)

sfunction

### 32.4.4 Symbol References

Keys:

```
\stex_keys_define:nnnn{symname}{
  \tl_clear:N \l_stex_key_pre_tl
  \tl_clear:N \l_stex_key_post_tl
  %\tl_clear:N \l_stex_key_root_tl
}{
  pre    .tl_set:N   = \l_stex_key_pre_tl ,
  post   .tl_set:N   = \l_stex_key_post_tl ,
  %root   .code:n     = {}%.tl_set:N   = \l_stex_key_root_tl
}{}

\stex_keys_define:nnnn{symref}{
  %\tl_clear:N \l_stex_key_root_tl
}{
  root   .code:n     = {}%.tl_set:N   = \l_stex_key_root_tl
}{}
```

\symref
\sr

```
\NewDocumentCommand \symref { O{} m m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \__stex_comps_do_ref:nNn{#3}\symrefemph@uri\_stex_term_oms:nn
}
\let\sr\symref
```

(*End of definition for* `\symref` *and* `\sr`. *These functions are documented on page 152.*)

sfunction

\symname
\sn
\sns
\Symname
\Sn\Sns

```
\cs_new:Nn \__stex_comps_split_slash:n {
  \__stex_comps_split_slash_i:nw #1//
}

\cs_new:Npn \__stex_comps_split_slash_i:nw #1/#2/ {
  \tl_if_empty:nTF{#2}{#1}{
    \__stex_comps_split_slash_i:nw #2/
  }
}

\NewDocumentCommand \symname { O{} m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
  \tl_set:Ne \l_stex_current_display_tl {
    \exp_args:Ne \__stex_comps_split_slash:n {\stex_symbol_uri_name:N \l_stex_get_symbol_ur
  }
  \__stex_comps_do_ref:nNn{
    \l_stex_key_pre_tl \l_stex_current_display_tl \l_stex_key_post_tl
  }\symrefemph@uri\_stex_term_oms:nn
}
```

```
\let\sn\symname
\protected\def\sns{\symname[post=s]}

\cs_new:Nn \__stex_comps_uppercase:n { \uppercase{#1}}

\NewDocumentCommand \Symname { O{} m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
  \tl_set:Ne \l_stex_current_display_tl {
    \exp_args:Ne \__stex_comps_split_slash:n {\stex_symbol_uri_name:N \l_stex_get_symbol_ur
  }
  \__stex_comps_do_ref:nNn{
    \l_stex_key_pre_tl \exp_args:NNe \use:nn \__stex_comps_uppercase:n \l_stex_current_disp
  }\symrefemph@uri\_stex_term_oms:nn
}
\let\Sn\Symname
\protected\def\Sns{\Symname[post=s]}
```

(*End of definition for* `\symname` *and others. These functions are documented on page 152.*)

sfunction

`\varref` `\varname` `\Varname`

```
\NewDocumentCommand \varref { O{} m m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_var:n{#2}
  \__stex_comps_do_ref:nNn{#3}\varemph@uri{
    \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
    \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
    \def\comp{\_varcomp}
    \_stex_term_omv:nn
  }
}

\NewDocumentCommand \varname { O{} m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_var:n{#2}
  \__stex_comps_do_ref:nNn{
    \l_stex_key_pre_tl\l_stex_get_variable_str\l_stex_key_post_tl
  }\varemph@uri{
    \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
    \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
    \def\comp{\_varcomp}
    \_stex_term_omv:nn
  }
}

\NewDocumentCommand \Varname { O{} m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_var:n{#2}
```

```
  \__stex_comps_do_ref:nNn{
    \l_stex_key_pre_tl\exp_after:wN\__stex_comps_uppercase:n\l_stex_get_variable_str\l_stex
  }\varemph@uri{
    \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
    \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
    \def\comp{\_varcomp}
    \_stex_term_omv:nn
  }
}
```

(*End of definition for* `\varref`, `\varname`, *and* `\Varname`. *These functions are documented on page 152.*)

sfunction

`\definiendum` `\definame` `\Definame` `\defnotation`

```
\stex_keys_define:nnnn{defname}{}{
  gf     .code:n       = {},
  root   .code:n       = {}
}{symname}

\stex_keys_define:nnnn{definiendum}{}{
  gf     .code:n       = {},
  root   .code:n       = {}
}{}

\NewDocumentCommand \definiendum { O{} m m} {
  \stex_keys_set:nn{definiendum}{ #1 }
  \__stex_comps_do_defref:nn{#2}{#3}
}
\stex_deactivate_macro:Nn \definiendum {definition~environments}

\NewDocumentCommand \definame { O{} m } {
  \stex_keys_set:nn{defname}{#1}
  \__stex_comps_do_defref:nn{#2}{
    \l_stex_key_pre_tl\l_stex_current_display_tl\l_stex_key_post_tl
  }
}
\protected\def\definames{\definame[post=s]}
\stex_deactivate_macro:Nn \definame {definition~environments}


\NewDocumentCommand \Definame { O{} m } {
  \stex_keys_set:nn{defname}{#1}
  \__stex_comps_do_defref:nn{#2}{
    \l_stex_key_pre_tl \exp_args:NNe \use:nn \__stex_comps_uppercase:n \l_stex_current_disp
  }
}
\protected\def\Definames{\Definame[post=s]}
\stex_deactivate_macro:Nn \Definame {definition~environments}


\NewDocumentCommand \defnotation{ m } {
  \_stex_next_symbol:n { \def\comp{\_defcomp}}#1
}
\stex_deactivate_macro:Nn \defnotation {definition~environments}
```

(*End of definition for* `\definiendum` *and others. These functions are documented on page 152.*)

sfunction

Does the actual referencing:

```
\cs_new_protected:Nn \__stex_comps_do_ref:nNn {
  \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
  \bool_if:NTF \l_stex_allow_semantic_bool{
    \tl_set_eq:NN \l_stex_current_symbol_uri \l_stex_get_symbol_uri
    \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_current_symbol_uri
    %\str_set:Ne \l_stex_current_symbol_name_str { \stex_symbol_uri_name:N \l_stex_current_
    %\str_if_in:NnT \l_stex_current_symbol_name_str / {
    %  \str_set:Ne \l_stex_current_symbol_name_str {
    %    \exp_after:wN \__stex_comps_slash:w \l_stex_current_symbol_name_str
    %    /\_stex_args_end:
    %  }
    %}
    \tl_clear:N \l_stex_current_term_tl
    \def\comp{\_comp}
    \let\compemph@uri#2
    #3{}{\comp{#1}}
  }{
    \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
  }
  \group_end:
}
```

`\definame` et al:

```
\cs_new_protected:Nn \__stex_comps_do_defref:nn {
  \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
  \group_begin:
  \stex_get_symbol:n{#1}
  \bool_if:NTF \l_stex_allow_semantic_bool{
    \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
    \tl_set:Nn \l_stex_current_display_tl {
      \stex_symbol_uri_name:N \l_stex_get_symbol_uri
    }
    %\str_if_in:NnT \l_stex_get_svariable_str / {
    %  \str_set:Ne \l_stex_get_variable_str {
    %    \exp_after:wN \__stex_comps_slash:w \l_stex_get_variable_str
    %    /\_stex_args_end:
    %  }
    %}
    \exp_args:Ne \stex_ref_new_sym_target:n \l_stex_current_full_tl
    \_do_comp:nnNn {definiendum}{\l_stex_current_full_tl}\defemph@uri{#2}
  }{
    \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
  }
  \group_end:
}

\cs_new:Npn \__stex_comps_slash:w #1/#2/#3\_stex_args_end: {
  \tl_if_empty:nTF{#3}{
  #2
  }{
    \__stex_comps_slash:w #2 / #3 \_stex_args_end:
```

```
  }
}
```

\foldexpr

```
\stex_if_html_backend:TF {
  \NewDocumentCommand \foldexpr { O{} mm } {
    \stex_annotate:nn{
      data-ftml-fold-expression={\exp_args:Ne \str_if_eq:nnT{#1}{show}{show}}
    }{
      \stex_annotate:nn{
        data-ftml-fold-short={}
      }{#2}
      #3
    }
  }
}{
  \NewDocumentCommand \foldexpr { O{} mm } {
    \exp_args:Ne \str_if_eq:nnTF{#1}{hide}{
      \underbrace{...}{#2}
    }{
      \underbrace{#3}{#2}
    }
  }
}
```

(*End of definition for* `\foldexpr`. *This function is documented on page 152.*)

sfunction

## 32.5 Checking

\stex_if_check_terms_p:
\stex_if_check_terms:*TF*

```
⟨@@=stex_check⟩
\stex_if_html_backend:TF {
  \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
    \prg_return_false:
  }
}{
  \stex_get_env:Nn\__stex_check_env_str{STEX_CHECKTERMS}
  \str_if_empty:NF\__stex_check_env_str{
    \exp_args:No \str_if_eq:nnF \__stex_check_env_str{false}{
      \bool_set_true:N \c_stex_check_terms_bool
    }
  }
  \bool_if:NTF \c_stex_check_terms_bool {
    \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
      \prg_return_true:
    }
  }{
    \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
      \prg_return_false:
    }
  }
}
```

(*End of definition for* `\stex_if_check_terms:TF`. *This function is documented on page 152.*)

sfunction

`\stex_check_term:nn`

```
\stex_if_check_terms:TF{
  \cs_new_protected:Nn \stex_check_term:nn {
    \marginpar {\tiny Checking~#1:~
      \group_begin:
        $#2$
      \group_end:
    }
  }
}{
  \cs_new_protected:Nn \stex_check_term:nn {}
}
```

(*End of definition for* `\stex_check_term:nn`. *This function is documented on page 152.*)

sfunction

`\_stex_check_terms:`

```
\stex_if_check_terms:TF{
  \cs_new_protected:Nn \_stex_check_terms: {
    \stex_debug:nn{check_terms}{Checking~type...}
    \tl_if_empty:NF \l_stex_key_type_tl {
      \stex_check_term:nn{type}\l_stex_key_type_tl
    }
    \stex_debug:nn{check_terms}{Checking~definiens...}
    \tl_if_empty:NF \l_stex_key_def_tl {
      \stex_check_term:nn{definiens}\l_stex_key_def_tl
    }
    \stex_debug:nn{check_terms}{Checking~return...}
    \tl_if_empty:NF \l_stex_key_return_tl {
      \stex_check_term:nn{return}{\l_stex_key_return_tl!}
    }
    \stex_debug:nn{check_terms}{Checking~argument~types...}
    \group_begin:\l_stex_key_argtypes_clist\group_end:
    \tl_if_empty:NF \l_stex_key_argtypes_clist {
      \stex_check_term:nn{argument~types}{\l_stex_key_argtypes_clist}
    }
  }
}{
  \cs_new_protected:Nn \_stex_check_terms: {}
}

```

(*End of definition for* `\_stex_check_terms:`. *This function is documented on page 152.*)

sfunction

# Chapter 33

# Notations

```
⟨@@=stex_notations⟩
```

keys:

```
\stex_keys_define:nnnn{notation}{
  \str_clear:N \l_stex_key_variant_str
  \str_clear:N \l_stex_key_prec_str
  \str_clear:N \l_stex_key_op_tl
  \str_clear:N \l_stex_key_intent_str
  \clist_clear:N \l_stex_key_intent_args_clist
}{
  variant    .str_set_x:N  = \l_stex_key_variant_str ,
  prec       .str_set_x:N  = \l_stex_key_prec_str ,
  op         .tl_set:N     = \l_stex_key_op_tl ,
  intent     .str_set:N    = \l_stex_key_intent_str ,
  argnames   .clist_set:N  = \l_stex_key_intent_args_clist ,
  unknown    .code:n       = {
    \str_if_empty:NTF \l_keys_key_str {
        \str_set:Ne \l_stex_key_variant_str {\l_keys_key_tl}
    }{
        \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
    }
  }
}{style}

\stex_keys_define:nnnn{symdef}{}{}{decl,notation}

\stex_keys_define:nnnn{vardef}{
  \bool_set_false:N \l__stex_notations_bind_bool
}{
  bind  .bool_set:N  = \l__stex_notations_bind_bool
}{symdef}

```

\notation

```
\stex_new_stylable_cmd:nnnn {notation} { s m O{} m} {
  \stex_keys_set:nn{notation}{#3}
  \stex_get_symbol:n{#2}
  \stex_notation_parse:n{#4}
  \stex_if_check_terms:T{ \_stex_notation_check: }
```

```
  \_stex_notation_add:
  \stex_if_do_html:T {
    \def\comp{\_comp}
    \_stex_notation_do_html:n{\stex_use_symbol_uri:N \l_stex_get_symbol_uri}
  }
  \IfBooleanTF#1{
    \_stex_notation_set_default:n{
      \l_stex_get_symbol_uri
    }
  }{}
  \stex_if_smsmode:F{
    \group_begin:
    \__stex_notations_styledefs:
    \stex_style_apply:
    \group_end:
  }
  \stex_smsmode_do:
}{}

\stex_deactivate_macro:Nn \notation {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \notation}
\stex_sms_allow_escape:N \notation

\cs_new_protected:Nn \__stex_notations_styledefs: {
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \str_set:Nn \thisdeclname {\stex_symbol_uri_name:N \l_stex_get_symbol_uri}
  \tl_set:Ne \thisdecluri {\stex_use_symbol_uri:N \l_stex_get_symbol_uri}
  \def\thisnotation{
    \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
      \_stex_notation_make_args:
    }
  }
}
```

(*End of definition for* `\notation`. *This function is documented on page 153.*)

sfunction

`\varnotation`

```
\stex_new_stylable_cmd:nnnn {varnotation} { s m O{} m} {
  \stex_keys_set:nn{notation}{#3}
  \stex_get_var:n{#2}
  \str_set_eq:NN \l_stex_key_name_str \l_stex_get_variable_str
  \stex_notation_parse:n{#4}
  \stex_if_check_terms:T{ \_stex_notation_check: }
  \_stex_var_notation_macro:
  \IfBooleanTF#1{
    \_stex_notation_set_default:n{\l_stex_get_variable_str}
  }{}
  \group_begin:
  \tl_set_eq:NN \thisvarname \l_stex_get_variable_str
  \tl_clear:N \thisstyle
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \def\thisnotation{
    \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
```

```
      \_stex_notation_make_args:
    }
  }
  \stex_style_apply:
  \group_end:
}{}
```

(*End of definition for* `\varnotation`. *This function is documented on page 153.*)

sfunction

`\stex_notation_parse:n`

```
\cs_new_protected:Nn \stex_notation_parse:n {
  \tl_if_empty:NF \l_stex_key_op_tl {
    \tl_set:Ne \l_stex_key_op_tl { \exp_not:N\maincomp {
      \exp_args:No \exp_not:n \l_stex_key_op_tl
    } }
  }
  \seq_clear:N \l__stex_notations_precs_seq
  \tl_clear:N \l_stex_notation_args_tl
  \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
    \__stex_notations_const_precs:
    \tl_if_empty:NT \l_stex_key_op_tl {
      \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
    }
  }{
    \__stex_notations_fun_precs:
    \__stex_notations_do_missing_args:n{#1}
    \tl_if_empty:NT \l_stex_key_op_tl {
      \hbox_set:Nn \l_tmpa_box {
        \tl_clear:N \l_stex_current_full_tl
        \tl_clear:N \l_stex_current_display_tl
        \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
        \cs_set:Npn \maincomp ##1 {
          \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
          ##1
        }
        \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
        \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
        \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
          ##1 ##2
        }
        \stex_suppress_html:n{$\l_tmpa_cs abcdefghj$}
      }
    }
  }
  \exp_args:NNe
  \tl_set:Nn \l_stex_notation_macrocode_cs {
    \STEXInternalNotation
      { \l_stex_key_variant_str }
      { \l__stex_notations_opprec_tl }
      { \l_stex_key_intent_str }
      { \l_stex_notation_args_tl }
      {
        \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0
```

```
        { \exp_not:n { \maincomp{ #1 } } }
        { \exp_not:n { #1 } \l__stex_notations_missing_tl }
      }
  }
  \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
}
```

precedences:

```
\cs_new_protected:Nn \__stex_notations_const_precs: {
  \str_if_empty:NTF \l_stex_key_prec_str {
    \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
  }{
    \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
      \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
    }{
      \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
    }
  }
}

\cs_new_protected:Nn \__stex_notations_fun_precs: {
  \str_if_empty:NTF \l_stex_key_prec_str {
    \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
  }{
    \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
      \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
    }{
      \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
    }
  }
  \str_if_empty:NTF \l_stex_key_prec_str {
    \tl_set:Nn \l__stex_notations_opprec_tl { 0 }
    \int_step_inline:nn \l_stex_get_symbol_arity_int {
      \seq_put_right:Nn \l__stex_notations_precs_seq {0}
    }
  }{
    \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
      \stex_debug:nn{notation}{No~brackets}
      \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
      \int_step_inline:nn \l_stex_get_symbol_arity_int {
        \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infprec
      }
    }\__stex_notations_parse_precs:
  }
  \__stex_notations_do_argnames:
}

\cs_new_protected:Nn \__stex_notations_parse_precs: {
  \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}
  \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
  \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
    \tl_set_eq:NN \l__stex_notations_opprec_tl \l__stex_notations_str
    \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
      \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
        {\tl_to_str:n{x}} \l__stex_notations_str
```

```
    }
  }{
    \tl_set:No \l__stex_notations_opprec_tl { 0 }
  }
  \int_step_inline:nn \l_stex_get_symbol_arity_int {
    \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
      \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
    }{
      \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_opprec_tl
    }
  }
}
```

Experimental; named arguments:

```
\cs_new_protected:Nn \__stex_notations_do_argnames: {
  \tl_clear:N \l_stex_notation_args_tl
  \_stex_map_args:N \__stex_notations_do_argname:nn
}

\cs_new_protected:Nn \__stex_notations_do_argname:nn {
  \clist_if_empty:NTF \l_stex_key_intent_args_clist {
    \tl_put_right:Ne \l_stex_notation_args_tl {
      #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}{
        \str_if_empty:NF \l_stex_key_intent_str {#1}
      }
    }
  }{
    \tl_put_right:Ne \l_stex_notation_args_tl {
      #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}
      {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
    }
    \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
  }
}
```

inserts hidden arguments that don't occur in the body of the notation:

```
\cs_new:Nn \__stex_notations_do_missing_args:n {
  \str_set:Nn \l__stex_notations_missing_str {#1}
  \exp_args:NNe \seq_set_split:NnV \l_tmpa_seq {\c_hash_str\c_hash_str\c_hash_str\c_hash_st
  \str_clear:N \l__stex_notations_missing_str
  \seq_map_inline:Nn \l_tmpa_seq {
    \tl_put_right:Nn \l__stex_notations_missing_str { ##1 }
  }
  \tl_clear:N \l__stex_notations_missing_tl
  \_stex_map_args:N \__stex_notations_add_missing_args:nn
}

\cs_new:Nn \__stex_notations_add_missing_args:nn {
  % TODO this skips arguments if e.g. ##1 occurs rather than #1!!
  \exp_args:NNe \str_if_in:NnF \l__stex_notations_missing_str {\c_hash_str\c_hash_str#1}{
    \tl_put_right:Nn \l__stex_notations_missing_tl{\STEXinvisible{## #1}}
  }
}
```

(*End of definition for* `\stex_notation_parse:n`. *This function is documented on page 153.*)

sfunction

_stex_notation_add:

```
\cs_new_protected:Nn \_stex_notation_add: {
  \stex_add_notation:ooeoo
    {\l_stex_get_symbol_uri}
    \l_stex_key_variant_str
    {\int_use:N \l_stex_get_symbol_arity_int}
    \l_stex_notation_macrocode_cs
    \l_stex_key_op_tl
}
```

(*End of definition for* `\_stex_notation_add:`. *This function is documented on page 153.*)

sfunction

\stex_add_notation:nnnnn
\stex_add_notation:ooeoo

```
\cs_new:Nn \__stex_notations_macro:nn {
  l_stex_notation_ #1?#2 _cs
}
\cs_new:Nn \__stex_notations_op_macro:nn {
  l_stex_notation_ #1?#2 _op_cs
}

\cs_new_protected:Nn \stex_add_notation:nnnnn {
  \exp_args:Nne \stex_debug:nn{notations}{Adding~notation:^^J
    #1~\c_hash_str#2~#3^^J
    \tl_to_str:n{#4}^^J\tl_to_str:n{#5}^^J
    to~\stex_use_module_uri:N \l_stex_current_module_uri
  }
  \prop_gput:cen{ \_stex_notations_macro:N \l_stex_current_module_uri }
  {\stex_use_symbol_uri:n{#1}?#2}{{#1}{#2}{#3}{#4}{#5}}
  \stex_do_up_to_module:n {
    \__stex_notations_set_macro:nnnnn{#1}{#2}{#3}{#4}{#5}
    %\__stex_notations_activate_not:nn{#1}{#2}
  }
}
\cs_generate_variant:Nn \stex_add_notation:nnnnn {ooeoo}

\cs_new_protected:Nn \_stex_activate_notations: {
  \stex_debug:nn{activating}{All~notations:~\cs_meaning:c{ \_stex_notations_macro:N \l_stex
  \prop_map_inline:cn{ \_stex_notations_macro:N \l_stex_current_module_uri }{
    \__stex_notations_set_macro:nnnnn ##2
  }
}

\cs_new_protected:Nn \__stex_notations_activate_not:nn {
  \bool_set_false:N \l_tmpa_bool
  \stex_debug:nn{notations}{activating~\stex_use_symbol_uri:n{#1}?#2}
  \prop_map_inline:cn{ \_stex_notations_macro:N \l_stex_current_module_uri }{
    %\stex_debug:nn{notations}{##1?}
    \exp_args:Ne \str_if_eq:nnT{\stex_use_symbol_uri:n{#1}?#2}{##1}{
      \prop_map_break:n{
        %\stex_debug:nn{notations}{Found:~\tl_to_str:n{##2}}
        \bool_set_true:N \l_tmpa_bool
```

```
        \__stex_notations_set_macro:nnnnn ##2
      }
    }
  }
  \bool_if:NF \l_tmpa_bool {
    \errmessage{Woop}
  }
}

\cs_new_protected:Nn \__stex_notations_set_macro:nnnnn {
  \tl_set:cn {\__stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{#2}}{#4}
  \cs_if_exist:cF{\__stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{
    \tl_set:cn {\__stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{#4}
  }
  \tl_if_empty:nF{#5}{
    \tl_set:cn{\__stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{#2}}{#5}
    \cs_if_exist:cF{\__stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{
      \tl_set:cn{\__stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{#5}
    }
  }
}
```

(*End of definition for* `\stex_add_notation:nnnnn`. *This function is documented on page 153.*)

sfunction

`\_stex_map_args:N`
`\_stex_map_notation_args:N`

```
\cs_new:Nn \_stex_map_args:N {
  \tl_if_empty:NF \l_stex_get_symbol_args_tl {
    \exp_after:wN \__stex_notations_map_args_i:w \exp_after:wN
    #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
  }
}
\cs_new:Npn \__stex_notations_map_args_i:w #1 #2 #3 #4 \__stex_notations_args_end: {
  #1 {#2} {#3}
  \tl_if_empty:nF{#4}{
    \__stex_notations_map_args_i:w #1 #4 \__stex_notations_args_end:
  }
}

\cs_new:Nn \_stex_map_notation_args:N {
  \tl_if_empty:NF \l_stex_notation_args_tl {
    \exp_after:wN \__stex_notations_map_args_ii:w \exp_after:wN
    #1 \l_stex_notation_args_tl \__stex_notations_args_end:
  }
}
\cs_new:Npn \__stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \__stex_notations_args_end: {
  #1 {#2} {#3} {#4} {#5}
  \tl_if_empty:nF{#6}{
    \__stex_notations_map_args_ii:w #1 #6 \__stex_notations_args_end:
  }
}
```

(*End of definition for* `\_stex_map_args:N` *and* `\_stex_map_notation_args:N`. *These functions are documented on page 153.*)

sfunction

_stex_var_notation_macro:

```
\cs_new_protected:Nn \_stex_var_notation_macro: {
  \tl_set_eq:cN {\__stex_notations_macro:nn{\l_stex_key_name_str}{\l_stex_key_variant_str}}
  \cs_if_exist:cF {\__stex_notations_macro:nn{\l_stex_key_name_str}{}}{
    \tl_set_eq:cN{\__stex_notations_macro:nn{\l_stex_key_name_str}{}}
      \l_stex_notation_macrocode_cs
  }
  \tl_if_empty:NF \l_stex_key_op_tl {
    \tl_set_eq:cN {\__stex_notations_op_macro:nn{\l_stex_key_name_str}{\l_stex_key_variant_
    \cs_if_exist:cF{\__stex_notations_op_macro:nn{\l_stex_key_name_str}{}}{
      \cs_set_eq:cN{\__stex_notations_op_macro:nn{\l_stex_key_name_str}{}}
      \l_stex_key_op_tl
    }
  }
}
```

(*End of definition for* _stex_var_notation_macro:. *This function is documented on page 153.*)

sfunction

\setnotation

_stex_notation_set_default:n

```
\cs_new_protected:Nn \_stex_notation_set_default:n{
  \stex_if_in_module:TF{
    \stex_add_notation:ooeoo{#1}{}
      {\int_use:N \l_stex_get_symbol_arity_int}
      \l_stex_notation_macrocode_cs
      \l_stex_key_op_tl
  }{
    \cs_set_eq:cN {\__stex_notations_macro:nn {\stex_use_symbol_uri:N \l_stex_get_symbol_ur
      \l_stex_notation_macrocode_cs
    \tl_if_empty:NF \l_stex_key_op_tl {
      \cs_set_eq:cN{\__stex_notations_op_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbol
        \l_stex_key_op_tl
    }
  }
}

\cs_new_protected:Npn \setnotation #1 #2 {
  \stex_get_symbol:n{#1}
  \cs_if_exist:cTF{\__stex_notations_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbol_uri
    \tl_set_eq:Nc \l_stex_notation_macrocode_cs {\__stex_notations_macro:nn{\stex_use_symbo
    \cs_if_exist:cTF{\__stex_notations_op_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbo
      \tl_set_eq:Nc \l_stex_key_op_tl {\__stex_notations_op_macro:nn{\stex_use_symbol_uri:N
    }{
      \tl_clear:N \l_stex_key_op_tl
    }
    \_stex_notation_set_default:n{
      \l_stex_get_symbol_uri
    }
  }{
    \msg_error:nnxx{stex}{error/unknownnotation}{#2}{
      \stex_use_symbol_uri:N \l_stex_get_symbol_uri
    }
  }
}
```

(*End of definition for* `\setnotation` *and* `\_stex_notation_set_default:n`. *These functions are documented on page 153.*)

sfunction

`\stex_iterate_notations:nn`

```
\cs_new_protected:Nn \stex_iterate_notations:nn {
  \seq_clear:N \l__stex_notations_mods_seq
  \stex_pseudogroup_with:nn{\__stex_notations_not_cs:nnnnn}{
    \cs_set:Npn \__stex_notations_not_cs:nnnnn
    ##1 ##2 ##3 ##4 ##5 { #2 }
    \clist_map_function:nN {#1} \__stex_notations_it_not_i:n
  }
}

\cs_new_protected:Nn \__stex_notations_it_not_i:n {
  \seq_if_in:NnF \l__stex_notations_mods_seq {#1} {
    \seq_put_left:Nn \l__stex_notations_mods_seq {#1}
    \prop_map_inline:cn{\_stex_notations_macro:n{#1}}{
      \__stex_notations_not_cs:nnnnn ##2
    }
    \prop_map_inline:cn{\_stex_morphisms_macro:n{#1}}{
      \__stex_notations_it_not_check:nnnn ##2
    }
  }
}
\cs_new_protected:Nn \__stex_notations_it_not_check:nnnn {
  \tl_if_empty:nT{#1}{
    \__stex_notations_it_not_i:n {#2}
  }
}
```

(*End of definition for* `\stex_iterate_notations:nn`. *This function is documented on page 154.*)

sfunction

`\stex_use_notation:nnTF`
`\stex_use_op_notation:nnTF`

```
\cs_new_protected:Npn \stex_use_notation:nnTF #1 #2 #3 #4 {
  \cs_if_exist:cTF{\__stex_notations_macro:nn{#1}{#2}}{
    \cs_set_eq:Nc \l_stex_notation_cs {\__stex_notations_macro:nn{#1}{#2}}
    #3
  }{#4}
}
\cs_new_protected:Npn \stex_use_op_notation:nnTF #1 #2 #3 #4 {
  \cs_if_exist:cTF{\__stex_notations_op_macro:nn{#1}{#2}}{
    \cs_set_eq:Nc \l_stex_notation_cs {\__stex_notations_op_macro:nn{#1}{#2}}
    #3
  }{#4}
}
```

(*End of definition for* `\stex_use_notation:nnTF` *and* `\stex_use_op_notation:nnTF`. *These functions are documented on page 154.*)

sfunction

`\_stex_notation_check:`

```
\cs_new_protected:Nn \_stex_notation_check: {
```

```
  \stex_check_term:nn{notation}{
    \cs_set:Npn \comp ##1 {##1}
    \stex_debug:nn{check}{Checking~notation:~\cs_meaning:N \l_stex_notation_macrocode_cs ^^
    \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
      \_stex_notation_make_args:
    }
  }
}
```

(*End of definition for* `\_stex_notation_check:`. *This function is documented on page 154.*)

sfunction

`\_stex_notation_do_html:n`

```
\cs_new_protected:Nn \_stex_notation_do_html:n {
  \stex_annotate_invisible:n{\_stex_annotate_force_break:n{\hbox{\stex_annotate:nn {
    data-ftml-notation={#1},
    data-ftml-notationfragment={\l_stex_key_variant_str},
    data-ftml-precedence={\l__stex_notations_opprec_tl},
    data-ftml-argprecs={\seq_use:Nn \l__stex_notations_precs_seq ,}
  }{
    \cs_set_protected:Npn \argsep ##1 ##2 {
      \stex_annotate:nn{data-ftml-argsep={}}{
        ##1 \tl_if_empty:nTF{##2}{\!\,}{##2}
      }
    }
    \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
      \cs_set:Npn \__stex_notations_map_cs: ####1 { ##2 }
      \stex_annotate:nn{data-ftml-argmap={}}{
        \__stex_notations_map_cs:{##1} \stex_annotate:nn{data-ftml-argmap-sep={}}{##3}
      }
    }
    \cs_set_protected:Npn \maincomp {
      \_do_comp:nnNn {maincomp}{}\compemph@uri
    }
    \stex_debug:nn{notation}{Doing~notation~HTML:\meaning\l_stex_get_symbol_args_tl}
    $
      \tl_clear:N \l_stex_current_full_tl
      \stex_annotate:nn{data-ftml-notationcomp={}}{
        \exp_args:Nne \use:nn {
          \l_stex_notation_macrocode_cs {}
        }{
          \_stex_map_args:N \__stex_notations_make_arg_html:nn
        }
      }
    $
    \tl_if_empty:NF \l_stex_key_op_tl {
      $
      \tl_clear:N \l_stex_current_full_tl
      \stex_annotate:nn{data-ftml-notationopcomp={}}{
        \_stex_term_oms:nn{\l_stex_key_variant_str}{\l_stex_key_op_tl}
      }
      $
    }
  }}
```

```
  }}
}

\cs_new:Nn \__stex_notations_make_arg_html:nn {
  {\stex_annotate:nn{data-ftml-argnum=#1}{x}}
}
```

(*End of definition for* `\_stex_notation_do_html:n`. *This function is documented on page 154.*)

sfunction

`\_stex_notation_make_args:`

```
\cs_new:Nn \_stex_notation_make_args: {
  \_stex_map_notation_args:N \__stex_notations_make_arg:nnnn
}

\cs_new:Nn \__stex_notations_make_arg:nnnn  {
  \str_case:nnF #2 {
    a {{
        a\c_math_subscript_token{#1,1},
        a\c_math_subscript_token{#1,2}
      }}
    B {{
        B\c_math_subscript_token{#1,1},
        B\c_math_subscript_token{#1,2}
      }}
  }{{
    \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}
      {{#2}\c_math_subscript_token{#1}}
  }}
}
```

(*End of definition for* `\_stex_notation_make_args:`. *This function is documented on page 154.*)

sfunction

`\stex_do_default_notation:`

`\stex_do_default_notation_op:`

```
\cs_new_protected:Nn \stex_do_default_notation: {
  \stex_do_default_notation_op:
  \tl_if_empty:NTF \l_stex_current_args_tl {
    \tl_clear:N \l__stex_notations_args_tl
  }\__stex_notations_default_args:
  \tl_set:Ne \l_stex_default_notation {\STEXInternalNotation{}{0}{}{\l__stex_notations_args
    \exp_args:No \exp_not:n \l_stex_default_notation
  }}
}

\cs_new_protected:Nn \stex_do_default_notation_op: {
  \__stex_notations_make_name:
  \tl_set:Ne \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l__stex_n
}

\cs_new_protected:Nn \__stex_notations_default_args: {
  \__stex_notations_make_name:
  \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
  \tl_set:Ne \l__stex_notations_args_tl {
```

```
    \_stex_map_args:N \__stex_notations_augment_arg:nn
  }
  \tl_put_right:Nn \l_stex_default_notation {\comp(}
  \seq_clear:N \l_tmpa_seq
  \int_step_inline:nn \l_stex_current_arity_str {
    \seq_put_right:Nn \l_tmpa_seq {#### ##1}
  }
  \tl_put_right:Ne \l_stex_default_notation {
    \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}
  }
  \tl_put_right:Nn \l_stex_default_notation {\comp)}
}

\cs_new:Nn \__stex_notations_augment_arg:nn {
  #1#2{0}{}
}

\cs_new_protected:Nn \__stex_notations_make_name: {
  \exp_args:NNNe \seq_set_split:Nnn \l_tmpa_seq / \l_stex_current_display_tl
  \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
}
```

(*End of definition for* `\stex_do_default_notation:` *and* `\stex_do_default_notation_op:`*. These functions are documented on page 154.*)

sfunction

`\STEXInternalNotation`

```
\cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
  \__stex_notations_process:nnnnnn{#1}{#2}{#3}{#4}{#5}{
    \l__stex_notations_code_tl
    #6
  }
}

\cs_new_protected:Npn \__stex_notations_process:nnnnnn #1 #2 #3 #4 {
  \tl_if_empty:nTF{#4}{
    \__stex_notations_simple:nnnnn{#1}{#2}{#3}
  }{
    \__stex_notations_complex:nnnnnn{#1}{#2}{#3}{#4}
  }
}

\cs_new_protected:Nn \__stex_notations_simple:nnnnn {
  \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
  \tl_set:Nn \l__stex_notations_code_tl {
    \cs_set:Npn \l__stex_notations_cs {
      \_stex_maybe_brackets:nn{#2}{
        \_stex_term_oms_or_omv:nn{#1}{#4}
      }
    }
    \l__stex_notations_cs
  }
  #5
}
```

```
\cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
  \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
  \int_zero:N \l_tmpa_int
  \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \_stex_term_oma_or_omb:nn \_stex_term
  \tl_set:Nn \l__stex_notations_code_tl {
    \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
    {
      \_stex_maybe_brackets:nn{#2}{
        \_stex_term_oma_or_omb:nn{#1}{
          \bool_set_false:N \l_stex_brackets_dones_bool
          #5
        }
      }
    }
    \l__stex_notations_cs
  }
  \tl_set:Nn \l__stex_notations_after_tl{
    \exp_args:NNo
    \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
    \tl_put_left:Ne \l__stex_notations_code_tl {
      \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
    }
    #6
  }
  \__stex_notations_parse_args:nnnnw #4 \__stex_notations_args_end:
}

\cs_new_protected:Npn \__stex_notations_parse_args:nnnnw #1 #2 #3 #4 #5 \__stex_notations_a
  \tl_if_empty:nTF{#5}{
    \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}
  }{
    \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}
  }
}

\cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
  \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
  \__stex_notations_parse_args:nnnnw #5 \__stex_notations_args_end:
}

\cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
  \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
  \stex_debug:nn{sequences}{Executing~notation:~\meaning\l__stex_notations_code_tl}
  \l__stex_notations_after_tl
}

\cs_new_protected:Nn \__stex_notations_add:nnnnn {
  \int_incr:N \l_tmpa_int
  \str_case:nn{#2}{
    i {
      \tl_put_right:Nn \l__stex_notations_code_tl {
        {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
      }
```

```
    }
    b {
      \tl_set:Nn \l__stex_notations_pre_tl {
        \cs_set_eq:NN \_stex_term_oma_or_omb:nn \_stex_term_omb:nn
      }
      \tl_put_right:Nn \l__stex_notations_code_tl {
        {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
      }
    }
    a {
      \tl_put_right:Nn \l__stex_notations_code_tl {
        {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
      }
    }
    B  {
      \tl_set:Nn \l__stex_notations_pre_tl {
        \cs_set_eq:NN \_stex_term_oma_or_omb:nn \_stex_term_omb:nn
      }
      \tl_put_right:Nn \l__stex_notations_code_tl {
        {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
      }
    }
  }
}
```

(*End of definition for* `\STEXInternalNotation`. *This function is documented on page 154.*)

sfunction

`\argsep`

```
\cs_new_protected:Npn \argsep #1 #2 {
  \__stex_notations_check_aB_arg:Nn\argsep{#1}
  \stex_pseudogroup_with:nn{\_stex_term_do_aB_clist:}{
    \tl_set:Nn \_stex_term_do_aB_clist: {
      \seq_use:Nn \l_stex_aB_args_seq {#2}
    }
    #1
  }
}

\cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
  \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
    \_stex_term_arg_aB:nnnnn {
    \msg_error:nnx{stex}{error/assocarg}{\tl_to_str:n{#1}}
  }
}
```

(*End of definition for* `\argsep`. *This function is documented on page 154.*)

sfunction

`\argmap`

```
\cs_new_protected:Npn \argmap #1 #2 #3 {
  \__stex_notations_check_aB_arg:Nn\argmap{#1}
  \stex_pseudogroup_with:nn{
    \_stex_term_do_aB_clist:
```

```
    \__stex_notations_map_cs:
  }{
    \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }
    \tl_set:Nn \_stex_term_do_aB_clist: {
      \seq_clear:N \l_tmpa_seq
      \seq_map_inline:Nn \l_stex_aB_args_seq {
        \tl_if_eq:nnTF{##1}{\ellipses}{
          \seq_put_right:Nn \l_tmpa_seq \ellipses
        }{
          \seq_put_right:Nx \l_tmpa_seq {
            \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
          }
        }
      }
      \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
      \seq_use:Nn \l_stex_aB_args_seq {#3}
    }
    #1
  }
}
```

(*End of definition for* \argmap*. This function is documented on page 154.*)

sfunction

\argarraymap

```
\int_new:N \l__stex_notations_clist_count_int
\cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
  \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
  \stex_pseudogroup_with:nn{
    \_stex_term_do_aB_clist:
    \__stex_notations_map_cs:
  }{
    \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }
    \int_set:Nn \l__stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n
    \tl_set:Nn \_stex_term_do_aB_clist: {
      \tl_clear:N \l_tmpa_tl
      \int_zero:N \l_tmpa_int
      \seq_map_inline:Nn \l_stex_aB_args_seq {
        \int_incr:N \l_tmpa_int
        \int_compare:nNnT \l_tmpa_int > \l__stex_notations_clist_count_int {
          \int_set:Nn \l_tmpa_int 1
        }
        \tl_put_right:Ne \l_tmpa_tl {
          \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
          \clist_item:nn{#4}\l_tmpa_int
        }
      }
      \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
      \begin{array}{#2}
          \l_tmpa_tl
      \end{array}
    }
    #1
  }
}
```

(*End of definition for* `\argarraymap`. *This function is documented on page 154.*)

sfunction

## 33.1 Automated Bracketing

Variables for current (downwards) precedence, set of parentheses etc.:

```
\int_new:N \l_stex_notation_downprec
\tl_set:Nn \l__stex_notations_left_bracket_str (
\tl_set:Nn \l__stex_notations_right_bracket_str )
\bool_new:N \l_stex_brackets_dones_bool
```

`\infprec`
`\neginfprec`

```
\tl_const:Ne \infprec {\int_use:N \c_max_int}
\tl_const:Ne \neginfprec {-\int_use:N \c_max_int}

\int_set:Nn \l_stex_notation_downprec \infprec
```

(*End of definition for* `\infprec` *and* `\neginfprec`. *These variables are documented on page 154.*)

`\_stex_maybe_brackets:nn`

```
\cs_new_protected:Nn \_stex_maybe_brackets:nn {
  \bool_if:NTF \l_stex_brackets_dones_bool {
    \bool_set_false:N \l_stex_brackets_dones_bool
    #2
  } {
    \stex_debug:nn{brackets}{#1>\int_eval:n \l_stex_notation_downprec?}
    \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
      %\bool_if:NTF \l_stex_inparray_bool { #2 }{
        \dobrackets {
          #2
        }
      %}
    }{
      #2
    }
  }
}
```

(*End of definition for* `\_stex_maybe_brackets:nn`. *This function is documented on page 154.*)

sfunction

`\dobrackets`

```
\cs_new_protected:Npn \dobrackets #1 {
  \group_begin:
    \bool_set_true:N \l_stex_brackets_dones_bool
    \mathopen{\tl_if_exist:NT\l_stex_current_symbol_uri\comp
      \l__stex_notations_left_bracket_str
    }
    #1
  \group_end:
  \mathclose{\tl_if_exist:NT\l_stex_current_symbol_uri\comp
    \l__stex_notations_right_bracket_str
  }
}
```

(*End of definition for* `\dobrackets`. *This function is documented on page 155.*)

sfunction

`\withbrackets`

```
\cs_new_protected:Npn \withbrackets #1 #2 #3 {
  \stex_pseudogroup:nn{
    \tl_set:Nn \l__stex_notations_left_bracket_str { #1 }
    \tl_set:Nn \l__stex_notations_right_bracket_str { #2 }
    #3
  }{
    \stex_pseudogroup_restore:N \l__stex_notations_left_bracket_str
    \stex_pseudogroup_restore:N \l__stex_notations_right_bracket_str
  }
}
```

(*End of definition for* `\withbrackets`. *This function is documented on page 155.*)

sfunction

`\dowithbrackets`

```
\cs_new_protected:Npn \dowithbrackets #1 #2 #3 {
  \withbrackets{#1}{#2}{\dobrackets{#3}}
}
```

(*End of definition for* `\dowithbrackets`. *This function is documented on page 155.*)

sfunction

# Chapter 34

# Structures

⟨@@=stex_structures⟩

Keys:

```
\stex_keys_define:nnnn{mathstructure}{
  \tl_clear:N \l_stex_current_this_tl
  \str_clear:N \l__stex_structures_name_str
}{
  this    .tl_set:N = \l_stex_current_this_tl ,
  unknown .code:n   = {
    \str_if_empty:NTF \l_keys_key_str {
      \str_set:Ne \l__stex_structures_name_str {\l_keys_key_tl}
    }{
      \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
    }
  }
}{}
```

`mathstructure` (*env.*)

```
\stex_new_stylable_env:nnnnnnn {mathstructure}{m O{}}{
  \__stex_structures_begin:nn{#1}{#2}
  \stex_smsmode_do:
}{
  \stex_structural_feature_module_end:
  \__stex_structures_do_externals:
}{}{}{}

\stex_sms_allow_env:n{mathstructure}
\stex_deactivate_macro:Nn \mathstructure {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
```

Shared code for `mathstructure` and `extstructure`:

```
\cs_new_protected:Nn \__stex_structures_begin:nn {
  \stex_keys_set:nn {mathstructure}{#2}
  \str_if_empty:NT \l__stex_structures_name_str {
    \str_set:Nn \l__stex_structures_name_str {#1}
  }
  \def\comp{\_comp}

  \tl_set:Ne \l__stex_structures_struct_uri {
```

```
    \exp_args:No \stex_new_module_uri:n \l__stex_structures_name_str
  }

  \exp_args:Nne \use:nn { \stex_add_symbol:nnnnnnnN }
    { {#1}{\l__stex_structures_name_str}{0}{}{defed}{\l__stex_structures_struct_uri}}
    {}\stex_invoke_structure:
  \str_set:Ne \l_stex_macroname_str {#1}

  \exp_args:No \stex_structural_feature_module:nn
    {\l__stex_structures_name_str}{structure}
}

\cs_new_protected:Nn \__stex_structures_do_externals: {
  \tl_set:Nn \l__stex_structures_replace_this_tl {####1}
}
```

extstructure (*env.*)

```
\stex_new_stylable_env:nnnnnnn {extstructure}{m O{} m}{
  \seq_clear:N \l__stex_structures_imports_seq
  \clist_map_inline:nn{#3}{
    \stex_get_mathstructure:n{##1}
    \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
      \stex_if_module_exists:nT{####1}{
        \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
      }
    }
  }
  \__stex_structures_begin:nn{#1}{#2}
  \seq_map_inline:Nn\l__stex_structures_imports_seq{
    \stex_if_do_html:T {
      \hbox{\stex_annotate_invisible:nn
        {data-ftml-import={\stex_use_module_uri:n{##1}}} {}}
    }
    \stex_add_morphism:nonn
      {}{##1}{import}{}
    \stex_execute_in_module:e{
      \stex_activate_module:n {##1}
    }
  }
  \stex_smsmode_do:
}{
  \stex_structural_feature_module_end:
  \__stex_structures_do_externals:
}{}{}{}

\stex_sms_allow_env:n{extstructure}
\stex_deactivate_macro:Nn \extstructure {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \extstructure
}

\stex_new_stylable_env:nnnnnnn {extstructure*}{m}{
  \__stex_structures_new_extstruct_name:
```

```
  \seq_clear:N \l__stex_structures_imports_seq
  \stex_get_mathstructure:n{#1}

  \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
    \stex_if_module_exists:nT{##1}{
      \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
    }
  }

  \stex_module_uri_split_name:NNN \l__stex_structures_parent_uri \l__stex_structures_last_n

  \stex_execute_in_module:e{
    \__stex_structures_extend_structure:nnnn{\l__stex_structures_parent_uri}{\l__stex_struc
  }

  \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{}

  \seq_map_inline:Nn \l__stex_structures_imports_seq {
    \stex_if_do_html:T {
      \stex_annotate_invisible:nn
        {data-ftml-import= {\stex_use_module_uri:n{##1}}} {}
    }
    %\stex_add_morphism:nonn
    %  {}{##1}{import}{}
    %\stex_execute_in_module:e{
    %  \stex_activate_module:n {##1}
    %}
    \stex_activate_module:n {##1}
  }

  \stex_smsmode_do:
}{
  \prop_map_inline:cn{\_stex_symbol_macro:N \l_stex_current_module_uri }{
    \__stex_structures_check_def:nnnnnnnn ##2
  }
  \stex_structural_feature_module_end:
  %\exp_args:Ne \stex_do_up_to_module:n {
  %  \stex_activate_module:n {\exp_args:No \stex_new_module_uri:n {\l__stex_structures_exst
  %}
  \__stex_structures_do_externals:
}{}{}{}

\stex_sms_allow_env:n{extstructure*}
\exp_after:wN \stex_deactivate_macro:Nn
  \cs:w extstructure*\cs_end: {module~environments}
\stex_every_module:n {
  \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
}

\cs_new_protected:Nn \__stex_structures_extend_structure:nnnn {
  \stex_debug:nn{ext}{Extending~\stex_use_module_uri:n {#1} / #2~by~\stex_use_module_uri:n
  \exp_args:Nne \tl_put_right:cn{\_stex_module_code_macro:n{#4}}{
    \stex_activate_module:n{#3}
  }
```

```
  \exp_args:Nne \prop_put:cnn{\_stex_morphisms_macro:n{#4}}{\stex_use_module_uri:n{#3}}{
    {}{#3}{extstruct}{}
  }
  \exp_args:Nne \use:nn {\__stex_structures_extend_ii:nnnn}{
    \exp_args:NNe \use:nn \__stex_structures_extend_i:nnnnnnnNn {
      \prop_item:cn{\_stex_symbol_macro:n {#1} } { #2 }
    } { #3 }
  }{#1}{#2}
}

\cs_new_protected:Nn \__stex_structures_extend_ii:nnnn {
  \prop_put:cnn{\_stex_symbol_macro:n {#3} } { #4 }{#2}
  \tl_set:ce{#1}{
    \_stex_invoke_symbol:nnnnnnN {\stex_new_symbol_uri:nn {#3}{#4}}
    \use_none:nn #2
  }
}

\cs_new:Nn \__stex_structures_extend_i:nnnnnnnNn {
  {#1}{{#1}{#2}{#3}{#4}{#5}{#6,#9}{#7}#8}
}

\cs_new_protected:Nn \__stex_structures_check_def:nnnnnnnn {
  \tl_if_empty:nT{#5}{
    \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
  }
}

\cs_new_protected:Nn \__stex_structures_new_extstruct_name: {
  \stex_do_up_to_module:n {
    \str_set:Ne \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_
  }
  \str_set:Ne \l__stex_structures_exstruct_name_str {EXTSTRUCT_\l__stex_structures_extname_
}

\stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}
```

\this

```
\bool_new:N \l_stex_in_this_bool

\cs_new_protected:Npn \stex_current_this: {
  { \bool_set_true:N \l_stex_allow_semantic_bool
    \tl_if_empty:NTF \l_stex_current_this_tl {{}}{
      \tl_set:Nn \l_stex_current_full_tl {
        \exp_args:Ne \stex_use_symbol_uri:n {
          \exp_args:No \stex_new_symbol_uri:n \l__stex_structures_name_str
        }
      }
      \str_set_eq:NN \l_stex_current_display_tl \l__stex_structures_name_str
      \bool_set_true:N \l_stex_in_this_bool
      \maincomp{\l_stex_current_this_tl}
      \bool_set_false:N \l_stex_in_this_bool
    }
  }
```

```
}

\let \this \stex_current_this:
```

(*End of definition for* `\this`. *This function is documented on page 156.*)

sfunction

`\stex_get_mathstructure:n`

```
\cs_new_protected:Nn \stex_get_mathstructure:n {
  \stex_get_mathstructure_maybe:nTF{#1}{}{
    \msg_error:nnn{stex}{error/unknownstructure}{#1}
  }
}
\cs_new_protected:Npn \stex_get_mathstructure_maybe:nTF #1 #2 #3 {
  \tl_clear:N \l_stex_get_structure_module_uri
  \stex_get_symbol:nTF{#1}{
    \exp_args:No \tl_if_eq:NNTF \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
      \tl_set:Ne \l_stex_get_structure_module_uri { \exp_after:wN \__stex_structures_get:w
      #2
    }{
      #3
    }
  }{
    #3
  }
}

\cs_new:Npn \__stex_structures_get:w #1 , #2 \stex_end: { #1 }
```

(*End of definition for* `\stex_get_mathstructure:n`. *This function is documented on page 156.*)

sfunction

`\usestructure`

```
\cs_new_protected:Npn \usestructure #1 {
  \stex_if_smsmode:TF{
    \stex_get_mathstructure_maybe:nTF{ #1 }{
      \stex_debug:nn{usestructure}{Using~structure~#1:~\stex_use_module_uri:N \l_stex_get_s
      \__stex_structures_do_usestructure:
    }{
      \stex_debug:nn{usestructure}{Structure~#1~not~found~in~sms~mode}
    }
  }{
    \stex_get_mathstructure:n{ #1 }
    \stex_debug:nn{usestructure}{Using~structure~#1:~\stex_use_module_uri:N \l_stex_get_str
    \__stex_structures_do_usestructure:
  }
  \stex_smsmode_do:
}
\stex_sms_allow_escape:N \usestructure

\cs_new_protected:Nn \__stex_structures_do_usestructure: {
  \seq_clear:N \l__stex_structures_imports_seq
  \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
    \stex_if_module_exists:nT{##1}{
```

```
      \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
    }
  }
  \seq_map_inline:Nn \l__stex_structures_imports_seq {
    \stex_if_do_html:T {
      \hbox{\stex_annotate_invisible:nn
        {data-ftml-usemodule=\stex_use_module_uri:n {##1}} {}}
    }
    \stex_activate_module:n {##1}
  }
}
```

(*End of definition for* `\usestructure`. *This function is documented on page 156.*)

sfunction

# Chapter 35

# Statements

```
⟨@@=stex_statements⟩
```

Keys:

```
\stex_keys_define:nnnn{statement}{
  \str_clear:N \l_stex_key_name_str
  \str_clear:N \l_stex_key_macroname_str
  \clist_clear:N \l_stex_key_for_clist
  \str_clear:N \l_stex_key_args_str
  \tl_clear:N \l_stex_key_type_tl
  \tl_clear:N \l_stex_key_def_tl
  \tl_clear:N \l_stex_key_return_tl
  \clist_clear:N \l_stex_key_argtypes_clist
}{
  name     .str_set:N = \l_stex_key_name_str ,
  for      .clist_set:N = \l_stex_key_for_clist ,
  macro    .str_set:N = \l_stex_key_macroname_str ,
  % start  .str_set:N = \l_stex_key_title_str , % TODO remove
  type     .tl_set:N = \l_stex_key_type_tl ,
  judgment .code:n = {},
  from     .code:n= {}, % TODO remove
  to       .code:n={} % TODO remove
}{id,title,style,symargs}
```

`\_stex_do_for_list:`

```
\cs_new_protected:Npn \_stex_do_for_list: {
  \seq_clear:N \l_stex_fors_seq
  \clist_map_inline:Nn \l_stex_key_for_clist {
    \exp_args:Ne\stex_get_symbol:n{\tl_to_str:n{##1}}
    \seq_put_right:No \l_stex_fors_seq
      {\l_stex_get_symbol_uri}
  }
}
```

(*End of definition for* `\_stex_do_for_list:`*. This function is documented on page 157.*)

sfunction

`\stex_new_statement:nnn`

```
\cs_new_protected:Nn \stex_new_statement:nnn {
  \stex_new_stylable_env:nnnnnnn {#1}{O{}}{
```

```
    \stex_keys_set:nn{statement}{##1}
    #3

    \stex_if_smsmode:F {
      \exp_args:Nne \begin{stex_annotate_env}{
        \__stex_statements_html_keyvals:nn{#1}{false}
      }
      \tl_set_eq:NN \thistitle \l_stex_key_title_tl
      \str_set_eq:NN \thisname \l_stex_key_name_str
      \clist_set_eq:NN \thisfor \l_stex_key_for_str
      \stex_if_html_backend:TF {
        \noindent
        \stex_annotate:nn{data-ftml-title={}, style:display={none}}{\_stex_annotate_force_b
          \tl_if_empty:NF \l_stex_key_title_tl{~}
          \l_stex_key_title_tl
        }}
      }
      \stex_style_apply:
    }
    \_stex_do_id:
    \stex_smsmode_do:
  }{
    \stex_if_smsmode:F {
      \stex_if_html_backend:F \stex_style_apply:
      \end{stex_annotate_env}
    }
  }{}{}{s}
  \stex_sms_allow_env:n{s#1}

  \tl_if_empty:nF{#2}{\__stex_statements_make_macro:nnn{#1}{#2}{#3}}
}

\cs_new_protected:Nn \__stex_statements_make_macro:nnn {
  \exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { O{} m}{
    \group_begin:
      \stex_keys_set:nn{statement}{##1}
      #3
      \_stex_do_id:
      \stex_if_smsmode:F{
        \exp_args:Ne \stex_annotate:nn{\__stex_statements_html_keyvals:nn{#1}{true}}{
          \_stex_annotate_force_break:n{##2}
        }
      }
    \group_end:
    %\stex_if_smsmode:TF \stex_smsmode_do: \stex_ignore_spaces_and_pars:
    \stex_smsmode_do:
  }
  \exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
}

\cs_new:Nn \__stex_statements_html_keyvals:nn {
  data-ftml-#1={},
  data-ftml-inline={#2},
  \seq_if_empty:NF \l_stex_fors_seq {,
```

```
    data-ftml-fors={\seq_map_indexed_function:NN \l_stex_fors_seq \_stex_comma_sep_uri:nn }
  }
  \str_if_empty:NF \l_stex_key_id_str {,
    data-ftml-id={\l_stex_key_id_str}%{\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_ke
  }
  \clist_if_empty:NF \l_stex_key_style_clist {,
    data-ftml-styles={\l_stex_key_style_clist}
  }
}

\cs_new:Nn \_stex_comma_sep_uri:nn {
  \int_compare:nNnF{#1}=1 {,}
  \stex_use_symbol_uri:n{#2}
}
```

(*End of definition for* `\stex_new_statement:nnn`. *This function is documented on page 157.*)

sfunction

`\__stex_statements_setup:n` sets up a statement that may declare a new symbol with the given `role` by processing the optional arguments, calling `\stex_symdecl_do:`, etc.

```
\cs_new_protected:Nn \__stex_statements_setup:n {
  \str_if_empty:NF \l_stex_key_macroname_str {
    \str_if_empty:NT \l_stex_key_name_str {
      \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
    }
  }
  \_stex_do_for_list:
  \tl_clear:N \l_stex_current_def_uri

  \str_if_empty:NTF \l_stex_key_name_str \__stex_statements_setup_noname: {
    \__stex_statements_setup_named:n{#1}
  }
}

\cs_new_protected:Nn \__stex_statements_setup_named:n {
  \__stex_statements_force_id:
  \seq_put_right:Ne \l_stex_fors_seq {
    \l_stex_current_module_uri {\l_stex_key_name_str}
  }
  \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
  \str_set:Nn \l_stex_key_role_str {#1}
  \stex_symdecl_do:
  \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnnN}{
    {\l_stex_key_macroname_str}{\l_stex_key_name_str}
    {\int_use:N \l_stex_get_symbol_arity_int}
    {\l_stex_get_symbol_args_tl}
    {#1}{}{}\stex_invoke_symbol:
  }
  \stex_if_do_html:T \_stex_symdecl_html:
  \tl_set:Ne \l_stex_current_def_uri {\exp_args:No \stex_new_symbol_uri:nn \l_stex_current_
  \stex_debug:nn{statement}{name:~\stex_use_symbol_uri:N \l_stex_current_def_uri}
}

\cs_new_protected:Nn \__stex_statements_setup_noname: {
```

```
  \stex_debug:nn{statement}{no~name}
  \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
    \tl_set:Ne \l_stex_current_def_uri {\seq_item:Nn \l_stex_fors_seq 1}
    \stex_debug:nn{statement}{for:~\stex_use_symbol_uri:N \l_stex_current_def_uri}
  }{
    \stex_debug:nn{statement}{no~for}
  }
}

\cs_new_protected:Nn \__stex_statements_force_id: {
  %\str_if_empty:NT \l_stex_key_id_str {
  %  \_stex_ref_new_id:n{}
  %  \str_set_eq:NN \l_stex_key_id_str \l__stex_refs_str
  %}
}
```

(*End of definition for* `\__stex_statements_setup:n`.)

`\__stex_statements_setup_def:` Sets up all the macros allowed in definition-like environments:

```
\cs_new_protected:Nn \__stex_statements_setup_def: {
  \stex_if_smsmode:F{
    \seq_map_inline:Nn \l_stex_fors_seq {
      \exp_args:Ne \stex_ref_new_sym_target:n{\stex_use_symbol_uri:n{##1}}
    }
  }
  \stex_reactivate_macro:N \definiendum
  \stex_reactivate_macro:N \defnotation
  \stex_reactivate_macro:N \definame
  \stex_reactivate_macro:N \Definame
  \stex_reactivate_macro:N \varbind
  \stex_reactivate_macro:N \definiens
}
```

(*End of definition for* `\__stex_statements_setup_def:`.)

`sdefinition` (*env.*)

```
\stex_new_statement:nnn{definition}{def}{
  \__stex_statements_force_id:
  \__stex_statements_setup:n{}
  \__stex_statements_setup_def:
}
```

`sassertion` (*env.*)

```
\stex_new_statement:nnn{assertion}{ass}{
  \__stex_statements_setup:n{assertion}
  \stex_if_smsmode:F{
    \seq_map_inline:Nn \l_stex_fors_seq {
      \exp_args:Ne \stex_ref_new_sym_target:n{\stex_use_symbol_uri:n{##1}}
    }
  }
  \stex_reactivate_macro:N \varbind
  \stex_reactivate_macro:N \conclusion
  \stex_reactivate_macro:N \premise
  \stex_reactivate_macro:N \definiendum
```

```
  \stex_reactivate_macro:N \defnotation
  \stex_reactivate_macro:N \definame
  \stex_reactivate_macro:N \Definame
}
```

sexample (*env.*)

```
\stex_new_statement:nnn{example}{ex}{
  \stex_if_smsmode:F {\__stex_statements_setup:n{example}}
}
```

sparagraph (*env.*)

```
\stex_new_statement:nnn{paragraph}{}{
  \clist_if_in:NnTF \l_stex_key_style_clist {symdoc}{
    \__stex_statements_force_id:
    \__stex_statements_setup:n{}
    \__stex_statements_setup_def:
  }{
    \__stex_statements_setup:n{}
  }
}
```

definiens

```
\NewDocumentCommand \definiens { O{} m }{
  \group_begin:
  \_stex_definiens_impl:nn{#1}{#2}
  \group_end:
  \stex_smsmode_do:
}

\cs_new_protected:Nn \_stex_definiens_impl:nn {
  \tl_if_empty:nF {#1} {
    \stex_get_symbol:n { #1 }
    \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
  }
  \tl_if_empty:NT \l_stex_current_def_uri {
    \msg_error:nn{stex}{error/definiensfor}
  }
  \stex_debug:nn{definiens}{Checking~\stex_use_symbol_uri:N \l_stex_current_def_uri }

  \exp_args:No \_stex_add_definiens:nn \l_stex_current_def_uri {#2}
}

\stex_deactivate_macro:Nn \definiens {definition~environments}
\stex_sms_allow_escape:N \definiens
```

(*End of definition for* `definiens`. *This function is documented on page 157.*)

sfunction

`\_stex_add_definiens:nn`

```
\cs_new_protected:Nn \_stex_add_definiens:nn {
  \stex_if_do_html:TF {
    \stex_annotate:nn{data-ftml-definiens={\stex_use_symbol_uri:n{#1}}}{
      #2 %\_stex_annotate_force_break:n{ #2 }
    }
```

```
  }{ \stex_if_smsmode:F{#2} }
  \stex_if_in_module:T {
    \exp_args:Ne \str_if_eq:noT {\stex_symbol_uri_module:n{#1}}\l_stex_current_module_uri{
      \__stex_statements_add_definiens:n{#1}
    }
  }
}

\cs_new_protected:Nn \__stex_statements_add_definiens:n {
  \stex_debug:nn{definiens}{Adding~definiens~to~\stex_use_symbol_uri:n{#1}}
  \exp_args:Nne \prop_gput:cne{\exp_args:Ne \_stex_symbol_macro:n {\stex_symbol_uri_module:
    {\stex_symbol_uri_name:n {#1}} {
      \exp_args:Nne \use:nn { \__stex_statements_definiens_inner:nnnnnnnN }{ \exp_args:Nne
  }
}

\cs_new:Nn \__stex_statements_definiens_inner:nnnnnnnN {
  {#1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}
}
```

(*End of definition for* `\_stex_add_definiens:nn`. *This function is documented on page 157.*)

sfunction

`\varbind`

```
\NewDocumentCommand \varbind {m} {
  \clist_map_inline:nn {#1} {
    \stex_get_var:n {##1}
    \stex_if_do_html:T {
      \stex_annotate_invisible:nn {data-ftml-bind=\l_stex_get_variable_str}{}
    }
  }
}
\stex_deactivate_macro:Nn \varbind {definition~or~assertion~environments}
```

(*End of definition for* `\varbind`. *This function is documented on page 157.*)

sfunction

`\conclusion`

```
\NewDocumentCommand \conclusion { O{} m} {
  \group_begin:
  \tl_if_empty:nF {#1} {
    \stex_get_symbol:n { #1 }
    \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
  }
  \tl_if_empty:NT \l_stex_current_def_uri {
    \msg_error:nn{stex}{error/conclusionfor}
  }
  \stex_annotate:nn{ data-ftml-conclusion={\stex_use_symbol_uri:N \l_stex_current_def_uri}}
    #2 %\_stex_annotate_force_break:n{ #2 }
  }
  \group_end:
}
\stex_deactivate_macro:Nn \conclusion {assertion~environments}
```

(*End of definition for* `\conclusion`. *This function is documented on page 157.*)

sfunction

`\premise`

```
\NewDocumentCommand \premise {O{} m} {
  \tl_if_empty:nF {#1} {
    \stex_debug:nn{Here:}{Variable~#1}
    \exp_args:Nne\use:nn{\vardef}{{v#1}[name=#1]{#1}}
  }
  \stex_annotate:nn{data-ftml-premise={#1}}{#2}
}
\stex_deactivate_macro:Nn \premise {assertion~environments}
```

(*End of definition for* `\premise`. *This function is documented on page 157.*)

sfunction

# Chapter 36

# Proofs

⟨@@=stex_proof⟩

Keys:

```
\stex_keys_define:nnnn{ spfcommon }{
  \tl_clear:N \l_stex_key_for_clist
  \tl_clear:N \l_stex_key_term_tl
  \tl_clear:N \l_stex_key_method_tl
  \tl_clear:N \l_stex_key_just_tl
}{
  for         .clist_set:N  = \l_stex_key_for_clist ,
  term        .tl_set:N     = \l_stex_key_term_tl,
  method      .tl_set:N     = \l_stex_key_method_tl,
  just        .tl_set:N     = \l_stex_key_just_tl,
}{id,style,title}

\bool_set_false:N \l_stex_key_showname_bool

\stex_keys_define:nnnn{ spfnamed }{
  \str_clear:N \l_stex_key_name_str
  \str_clear:N \l_stex_key_numname_str
  \bool_set_false:N \l_stex_key_showname_bool
}{
  name        .str_set_x:N  = \l_stex_key_name_str,
  numname     .str_set_x:N  = \l_stex_key_numname_str,
  showname    .bool_set:N   = \l_stex_key_showname_bool
}{spfcommon}

\cs_new_protected:Nn \__stex_proof_do_spf_name: {
  \str_if_empty:NTF \l_stex_key_numname_str {
    \str_if_empty:NTF \l_stex_key_name_str {
      \bool_set_false:N \l_stex_key_showname_bool
    } {
      \exp_args:Nne\use:nn{\vardef}{{v\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_s
    }
  }{
    \bool_set_false:N \l_stex_key_showname_bool
    \__stex_proof_number_as_string:N \l__stex_proof_counter_str
    \exp_args:Nne \use:nn {\vardef}{{\l_stex_key_numname_str}[name=\l__stex_proof_counter_s
```

```
  }
}

\cs_new_protected:Nn \__stex_proof_do_spf_name_i: {
  \str_if_empty:NTF \l__stex_proof_key_numname_str {
    %\str_if_empty:NF \l__stex_proof_key_name_str {
      \exp_args:Nne\use:nn{\vardef}{{v\l__stex_proof_key_name_str}[name=\l__stex_proof_key_
    %}
  }{
    \exp_args:Nne \use:nn {\vardef}{{\l__stex_proof_key_numname_str}[name=\l__stex_proof_co
  }
}

\cs_new_protected:Nn \__stex_proof_do_spf_varname: {
  \bool_if:NT \l_stex_key_showname_bool {
    ~($\csname v\l_stex_key_name_str\endcsname$)~
  }
}

\stex_keys_define:nnnn{ spftop }{
  \tl_set_eq:NN \l_stex_key_proofend_tl \__stex_proof_proof_box_tl
  \bool_set_false:N \l_stex_key_hide_bool
  \tl_clear:N \l_stex_key_continues_tl
  \tl_clear:N \l_stex_key_functions_tl
  \tl_clear:N \l_stex_key_from_tl
}{
  proofend    .tl_set:N     = \l_stex_key_proofend_tl,
  hide        .code:n       = {
    \str_if_eq:eeT{#1}{true}{
      \bool_set_true:N \l_stex_key_hide_bool
    }
  },
  continues   .tl_set:N     = \l_stex_key_continues_tl, % <- unused
  functions   .tl_set:N     = \l_stex_key_functions_tl, % <- unused
  from        .tl_set:N     = \l_stex_key_from_tl % <- unused
}{spfcommon}

\stex_keys_define:nnnn{ subproof }{}{}{spftop,spfnamed}

\bool_set_true:N \l__stex_proof_inc_counter_bool
```

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, LaTeX only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

```
\intarray_new:Nn\l__stex_proof_counter_intarray{50}

\cs_new_protected:Npn \__stex_proof_insert_number: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
```

```
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
    \int_incr:N \l_tmpa_int
  }
}
\cs_new_protected:Nn \__stex_proof_number_as_string:N {
  \str_clear:N #1
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \str_put_right:Ne #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
    \int_incr:N \l_tmpa_int
  }
}

\cs_new_protected:Npn \__stex_proof_inc_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
  }
  \int_compare:nNnF \l_tmpa_int = 1 {
    \int_decr:N \l_tmpa_int
  }
  \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
    \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
  }
}

\cs_new_protected:Npn \__stex_proof_add_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
  }
  \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
}

\cs_new_protected:Npn \__stex_proof_remove_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
```

```
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
  }
  \int_decr:N \l_tmpa_int
  \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
}
```

sproof (*env.*)

```
\bool_set_false:N \l__stex_proof_in_spfblock_bool

\cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
  \intarray_gzero:N \l__stex_proof_counter_intarray
  \intarray_gset:Nnn \l__stex_proof_counter_intarray 1 1
  \stex_keys_set:nn{spftop}{#1}
  \_stex_do_for_list:
  \stex_if_do_html:T {
    \__stex_proof_html_env_proof:
  }
  \seq_map_inline:Nn \l_stex_fors_seq {
    \stex_debug:nn{definiens}{Adding~definiens~to~##1}
    \stex_if_do_html:TF{
      \STEXinvisible{\hbox{\_stex_add_definiens:nn {##1}{proven}}}
    }{
      \_stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
    }
  }
  \stex_if_do_html:F\stex_style_apply:
  %\_stex_do_id:
  \stex_reactivate_macro:N \subproof
  \stex_reactivate_macro:N \spfstep
  \stex_reactivate_macro:N \conclude
  \stex_reactivate_macro:N \assumption
  \stex_reactivate_macro:N \eqstep
  \stex_reactivate_macro:N \yield
  \stex_reactivate_macro:N \spfescape
  \stex_reactivate_macro:N \spfblock
  \stex_reactivate_macro:N \spfjust
  \stex_reactivate_macro:N \spfby
  \stex_reactivate_macro:N \spfarg
  \noindent\stex_annotate:nn{data-ftml-title={}}{\_stex_annotate_force_break:n{#2}}\par
  \stex_if_do_html:TF{
    \use:c{stex_annotate_env}{data-ftml-proofbody={}}
    \_stex_annotate_force_break:n{}
  }{
    \bool_if:NT \l_stex_key_hide_bool{\setbox0\vbox\bgroup}
  }
}

\cs_new_protected:Nn \__stex_proof_html_env_proof: {
  \exp_args:Nne \use:c{stex_annotate_env}{
    data-ftml-proof={}
    \seq_if_empty:NF \l_stex_fors_seq {,
```

```
      data-ftml-fors={\seq_map_indexed_function:NN \l_stex_fors_seq \_stex_comma_sep_uri:nn
    }
    \bool_if:NT \l_stex_key_hide_bool {,
      data-ftml-proofhide=true
    }
  }
  \__stex_proof_html:
}

\cs_new_protected:Nn \__stex_proof_html_env_subproof: {
  \str_if_empty:NF \l_stex_key_numname_str {
    \__stex_proof_number_as_string:N \l__stex_proof_counter_str
  }
  \exp_args:Nne \use:c{stex_annotate_env}{
    data-ftml-subproof={}
    \seq_if_empty:NF \l_stex_fors_seq {,
      data-ftml-fors={\seq_map_indexed_function:NN \l_stex_fors_seq \_stex_comma_sep_uri:nn
    }
    \bool_if:NT \l_stex_key_hide_bool {,
      data-ftml-proofhide=true
    }
    \str_if_empty:NTF \l_stex_key_numname_str {
      \str_if_empty:NF \l_stex_key_name_str {,
        data-ftml-stepname={\l_stex_key_name_str}
      }
    }{
      , data-ftml-stepname={\l__stex_proof_counter_str}
    }
  }
  \__stex_proof_html:
}

\cs_new_protected:Nn \__stex_proof_html: {
  \_stex_annotate_force_break:n{}
  \tl_set:Ne\l_tmpa_tl {\l_stex_key_term_tl\l_stex_key_method_tl\l_stex_key_just_tl}
  \tl_if_empty:NF\l_tmpa_tl{
    \stex_annotate_invisible:n{\hbox{
      \tl_if_empty:NF \l_stex_key_term_tl {
        $\stex_annotate:nn{data-ftml-proofterm={}}{\l_stex_key_term_tl}$
      }
      \tl_if_empty:NF \l_stex_key_just_tl {
        $\stex_annotate:nn{data-ftml-spfjust={}}{\l_stex_key_just_tl}$
      }
      \tl_if_empty:NF \l_stex_key_method_tl {
        \stex_annotate:nn{data-ftml-proofmethod={}}{\l_stex_key_method_tl}
      }
    }}
  }
}

\cs_new_protected:Nn \__stex_proof_start_comment: {
  \bool_if:NT \l__stex_proof_in_spfblock_bool {
    \par
    \group_begin:\stexcommentfont
```

```
  }
}
\cs_new_protected:Nn \__stex_proof_end_comment: {
  \bool_if:NT \l__stex_proof_in_spfblock_bool {
    \par
    \group_end:
  }
}
\cs_new_protected:Npn \spfescape #1{
  \__stex_proof_end_comment: #1 \__stex_proof_start_comment:
}
\stex_deactivate_macro:Nn \spfescape {sproof~environments}

\stex_new_stylable_env:nnnnnnn{proof}{O{} m}{
  \__stex_proof_begin_proof:nn{#1}{#2}
  \bool_set_true:N\l__stex_proof_in_spfblock_bool
  %\__stex_proof_start_list:n{}
  \__stex_proof_start_comment:
}{
  \stex_if_do_html:TF{
    %\__stex_proof_end_list:
    \use:c{endstex_annotate_env}\use:c{endstex_annotate_env}
  }\stex_style_apply:
}{
  \emph{\sproofautorefname :}~
}{
  \sproofend
}{s}

\AddToHook{env/sproof/end}{
  \__stex_proof_end_comment:
  \stex_if_do_html:F{\bool_if:NT \l_stex_key_hide_bool\egroup}
}


\stex_new_stylable_env:nnnnnnn{proof*}{O{}}{
  \__stex_proof_begin_proof:nn{#1}{}
  \bool_set_false:N\l__stex_proof_in_spfblock_bool
}{
  \stex_if_do_html:TF{\use:c{endstex_annotate_env}\use:c{endstex_annotate_env}}\stex_style_
}{
  \emph{Proof:}~
}{
  \sproofend
}{s}

\cs_new_protected:Nn \__stex_proof_start_list:n {
  %\par
  \group_begin:\list{}{
    \setlength\topsep{0pt}
    \setlength\parsep{0pt}
    \setlength\rightmargin{0pt}
  }\item[#1]
}
```

```
7829
7830 \cs_new_protected:Nn \__stex_proof_end_list: {
7831   %\par
7832   \endlist\group_end:
7833 }
```

subproof (*env.*)

```
7834 \str_set_eq:NN \subproofautorefname \spfstepautorefname
7835
7836 \cs_new_protected:Nn \__stex_proof_do_after_subproof:N {
7837   \bgroup
7838     \expandafter \__stex_proof_do_after_subproof_i:nn #1
7839   \egroup
7840   \__stex_proof_do_spf_name_i:
7841 }
7842 \cs_new_protected:Nn \__stex_proof_do_after_subproof_i:nn {
7843     \exp_args:Nno \stex_keys_set:nn{subproof}{#1}
7844     \str_gset_eq:NN \l__stex_proof_key_name_str \l_stex_key_name_str
7845     \str_gset_eq:NN \l__stex_proof_key_numname_str \l_stex_key_numname_str
7846     \bool_gset_eq:NN \l__stex_proof_key_showname_bool \l_stex_key_showname_bool
7847     \str_gset:Nn \l__stex_proof_counter_str {#2}
7848 }
7849 \cs_new_protected:Npn \__stex_proof_set_after_subproof:n {
7850   \str_if_empty:NTF \l_stex_key_numname_str {
7851     \str_if_empty:NTF \l_stex_key_name_str \use_none:n \__stex_proof_set_after_subproof_i:n
7852   }\__stex_proof_set_after_subproof_i:n
7853 }
7854 \cs_new_protected:Npn \__stex_proof_set_after_subproof_with_number:n {
7855   \str_if_empty:NTF \l_stex_key_numname_str {
7856     \str_if_empty:NTF \l_stex_key_name_str \use_none:n \__stex_proof_set_after_subproof_wit
7857   }\__stex_proof_set_after_subproof_with_number_i:n
7858 }
7859 \cs_new_protected:Nn \__stex_proof_set_after_subproof_i:n {
7860   \tl_gset:cn{subproof_arg_ \int_use:N \currentgrouplevel}{{#1}{}}
7861
7862   \__stex_proof_set_aftergroup: % \bool_if:NTF \l__stex_proof_in_spfblock_bool \__stex_proo
7863 }
7864 \cs_new_protected:Nn \__stex_proof_set_after_subproof_with_number_i:n {
7865   \__stex_proof_number_as_string:N \l__stex_proof_counter_str
7866   \tl_gset:ce{subproof_arg_ \int_use:N \currentgrouplevel}{{\exp_not:n{#1}}{\l__stex_proof_
7867   \__stex_proof_set_aftergroup:
7868 }
7869 \cs_new_protected:Nn \__stex_proof_set_aftergroup: {
7870     \aftergroup \__stex_proof_do_after_subproof:N
7871     \expandafter \aftergroup \csname subproof_arg_ \int_use:N \currentgrouplevel\endcsname
7872 }
7873
7874 \stex_new_stylable_env:nnnnnnn{subproof}{s O{} m}{
7875   \stex_keys_set:nn{subproof}{#2}
7876   \_stex_do_for_list:
7877   \stex_if_do_html:T {
7878     \__stex_proof_html_env_subproof:
7879   }
7880   \seq_map_inline:Nn \l_stex_fors_seq {
```

```
    \stex_debug:nn{definiens}{Adding~definiens~to~##1}
    \stex_if_do_html:TF{
      \STEXinvisible{\hbox{\_stex_add_definiens:nn {##1}{proven}}}
    }{
      \_stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
    }
  }

  \IfBooleanTF #1 {
    \bool_set_false:N \l__stex_proof_in_spfblock_bool
    \__stex_proof_set_after_subproof:n{#2}
    \stex_if_do_html:F \stex_style_apply:
    \str_if_empty:NF \l_stex_key_id_str {
      \__stex_proof_number_as_string:N \@currentlabel
      %\str_set:Ne \@currentHref{subproof.\@currentlabel}
      %\_stex_do_id:
    }


    \stex_annotate:nn{data-ftml-title={}}{#3}
  }{
    \bool_if:NTF \l__stex_proof_in_spfblock_bool {

      \__stex_proof_set_after_subproof_with_number:n{#2}

      %\str_if_empty:NF \l_stex_key_id_str {
        %\__stex_proof_number_as_string:N \@currentlabel
        %\str_set:Ne \@currentHref{subproof.\@currentlabel}
        %\_stex_do_id:
      %}
      \use:c{stex_annotate_env}{data-ftml-title={}} \__stex_proof_start_list:n{\__stex_proo
        \__stex_proof_add_counter:
        \stex_if_do_html:F \stex_style_apply:
    }{
      \noindent \stex_annotate:nn{data-ftml-title={}}{#3} \par
      \stex_if_do_html:F \stex_style_apply:

      \__stex_proof_set_after_subproof:n{#2}

      %\_stex_do_id:
    }
  }
  \stex_if_do_html:TF{
    \use:c{stex_annotate_env}{data-ftml-proofbody={}}
    \_stex_annotate_force_break:n{}
  }{\bool_if:NT \l_stex_key_hide_bool{\setbox0\vbox\bgroup}}
  \__stex_proof_start_comment:
}{
  \bool_if:NT\l__stex_proof_in_spfblock_bool {
    \__stex_proof_remove_counter:
    \__stex_proof_inc_counter:
  }

  \stex_if_do_html:TF{
```

```
    \use:c{endstex_annotate_env} %proofbody
    \bool_if:NT\l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
    \use:c{endstex_annotate_env} % subproof
  }{
    \stex_style_apply:
    \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
  }
}{}{}{}

\AddToHook{env/subproof/before}{
  \__stex_proof_end_comment:
}

\AddToHook{env/subproof/after}{
  \__stex_proof_start_comment:
}

\AddToHook{env/subproof/end}{
  \__stex_proof_end_comment:
  \stex_if_do_html:F{\bool_if:NT \l_stex_key_hide_bool\egroup}
}

\stex_deactivate_macro:Nn \subproof {sproof~environments}

```

`spfsketchenv` (*env.*) TODO:

```
\newenvironment{spfsketchenv}{}{}
```

`\spfsketch`

```
\stex_new_stylable_cmd:nnnn{spfsketch}{O{} m}{\par
  \begin{spfsketchenv}
  \stex_keys_set:nn{spftop}{#1}
  \_stex_do_for_list:
  \_stex_do_id:
  \par
  \exp_args:Nne \use:c{stex_annotate_env}{
    data-ftml-proofsketch={
      \seq_if_empty:NF \l_stex_fors_seq {
        \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
      }
    }
  }
    \stex_style_apply:
    #2\par
    \use:c{endstex_annotate_env}
  \end{spfsketchenv}
}{
  \noindent\emph{\spfsketchenvautorefname :}~
}
```

(*End of definition for* `\spfsketch`. *This function is documented on page 158.*)

sfunction

Keys for proof step macros:

```
\stex_keys_define:nnnn { spfsteps } {
```

```
  %\str_clear:N \l_stex_key_name_str
}{
  %name        .str_set_x:N  = \l_stex_key_name_str
}{spfcommon,spfnamed}
```

Common setup for all the proof step macros:

```
\cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
  \NewDocumentCommand #1 {s O{} +m} {
    \__stex_proof_end_comment:
    \stex_keys_set:nn{spfsteps}{##2}
    %\str_if_empty:NF \l_stex_key_name_str {
    %  \exp_args:Nne\use:nn{\vardef}{{v\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_
    %}
    \__stex_proof_do_spf_name:

    \spfstepenv
    \str_if_empty:NF \l_stex_key_id_str {
      %\__stex_proof_number_as_string:N \@currentlabel
      %\str_set:Ne \@currentHref{spfstep.\@currentlabel}
      %\_stex_do_id:
    }

    \bool_if:NTF \l__stex_proof_in_spfblock_bool {
      \IfBooleanTF ##1 {
        \__stex_proof_step_html_i:nn{#2}{##3}
      }{
        \__stex_proof_step_html:nn{#2}{\__stex_proof_start_list:n{#3} ##3 \__stex_proof_end
        #5
      }
      \endspfstepenv
      \__stex_proof_start_comment:
    }{
      \__stex_proof_step_html_i:nn{#2}{##3}
      \endspfstepenv
    }
  }
  \stex_deactivate_macro:Nn #1 {sproof~environments}
}

\cs_new_protected:Nn \__stex_proof_step_html:nn {
  \str_if_empty:NF \l_stex_key_numname_str {
    \__stex_proof_number_as_string:N \l__stex_proof_counter_str
  }
  \stex_if_do_html:TF{
    %\group_begin:
    \exp_args:Nne \use:c{stex_annotate_env}{
      data-ftml-spf#1={
        \seq_if_empty:NF \l_stex_fors_seq {
          \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
        }
      }
      \str_if_empty:NTF \l_stex_key_numname_str {
        \str_if_empty:NF \l_stex_key_name_str {,
          data-ftml-stepname={\l_stex_key_name_str}
        }
```

```
      }{
        , data-ftml-stepname={\l__stex_proof_counter_str}
      }
    }
    \__stex_proof_html:
    #2
    \par
    \use:c{endstex_annotate_env}
    %\group_end:
  }{ #2 }
}
\cs_new_protected:Nn \__stex_proof_step_html_i:nn {
  \stex_if_do_html:TF{
    \noindent
    %\group_begin:
    \exp_args:Ne \stex_annotate:nn{
      data-ftml-spf#1={
        \seq_if_empty:NF \l_stex_fors_seq {
          \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
        }
      }
      \str_if_empty:NTF \l_stex_key_numname_str {
        \str_if_empty:NF \l_stex_key_name_str {,
          data-ftml-stepname={\l_stex_key_name_str}
        }
      }{
        , data-ftml-stepname={\l__stex_proof_counter_str}
      }
    }{
      \__stex_proof_html:
      #2
    }
    %\group_end:
  }{ #2 }
}
```

`spfstepenv` (*env.*) TODO:

```
\newenvironment{spfstepenv}{
  \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
}{}
```

`spfblock` (*env.*)

```
\NewDocumentEnvironment{spfblock}{}{
  \bool_set_false:N \l__stex_proof_in_spfblock_bool
}{
  \aftergroup\__stex_proof_start_comment:
}

\stex_deactivate_macro:Nn \spfblock {sproof~environments}
\AddToHook{env/spfblock/before}{
  \__stex_proof_end_comment:
}
```

`\spfstep`

```
\__stex_proof_make_step_macro:Nnnnn \spfstep {step} {\__stex_proof_insert_number:\__stex_pr
```

(*End of definition for* `\spfstep`. *This function is documented on page 158.*)

sfunction

`\assumption`

```
\__stex_proof_make_step_macro:Nnnnn \assumption {assumption} {\__stex_proof_insert_number:\
```

(*End of definition for* `\assumption`. *This function is documented on page 158.*)

sfunction

`\conclude`

```
\__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {$\Rightarrow$} {} {}
```

(*End of definition for* `\conclude`. *This function is documented on page 158.*)

sfunction

`\eqstep`

```
\NewDocumentCommand \eqstep {s m}{
  \__stex_proof_end_comment:
  \spfstepenv

  \bool_if:NTF \l__stex_proof_in_spfblock_bool {
    \IfBooleanTF #1 {
      \__stex_proof_step_html_i:nn{eqstep}{$= #2$}
    }{
      \__stex_proof_step_html:nn{eqstep}{\__stex_proof_start_list:n{$=$} $#2$ \__stex_proof
    }
    \endspfstepenv
    \__stex_proof_start_comment:
  }{
    \__stex_proof_step_html_i:nn{eqstep}{$= #2$}
    \endspfstepenv
  }
}
\stex_deactivate_macro:Nn \eqstep {sproof~environments}
```

(*End of definition for* `\eqstep`. *This function is documented on page 158.*)

sfunction

`\yield`

```
\NewDocumentCommand \yield {+m}{
  \stex_annotate:nn{data-ftml-proofterm={}}{ #1 }
}
\stex_deactivate_macro:Nn \yield {sproof~environments}
```

(*End of definition for* `\yield`. *This function is documented on page 158.*)

sfunction

`\spfjust`

```
\newcommand\spfjust[1]{
  \stex_annotate:nn{data-ftml-spfjust={}}{ #1 }
}
\stex_deactivate_macro:Nn \spfjust {sproof~environments}
```

(*End of definition for* `\spfjust`. *This function is documented on page 158.*)

sfunction

`\spfby`

```
\newcommand\spfby[1]{
  \stex_annotate:nn{data-ftml-proofmethod={}}{ #1 }
}
\stex_deactivate_macro:Nn \spfby {sproof~environments}
```

(*End of definition for* `\spfby`. *This function is documented on page 158.*)

sfunction

`\spfarg`

```
\newcommand\spfarg[2][]{
  \stex_annotate:nn{data-ftml-spfarg={#1}}{ #2 }
}
\stex_deactivate_macro:Nn \spfarg {sproof~environments}
```

(*End of definition for* `\spfarg`. *This function is documented on page 159.*)

sfunction

`\sproofend`

```
\tl_set:Nn \__stex_proof_proof_box_tl {
  \ltx@ifpackageloaded{amssymb}{$\square$}{
    \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
  }
}

\tl_set:Nn \sproofend {
  \tl_if_empty:NF \l_stex_key_proofend_tl {
    \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip
  }
}
```

(*End of definition for* `\sproofend`. *This function is documented on page 159.*)

sfunction

`\stexcommentfont`

```
\cs_new_protected:Npn \stexcommentfont {
  \small\itshape
}
```

(*End of definition for* `\stexcommentfont`. *This function is documented on page 159.*)

sfunction

# Chapter 37

# Metatheory

⟨@@=stex_meta⟩

Setup:

```
\group_begin:
\cs_set:Npn \__stex_modules_persist_module: {}
\cs_set:Npn \stex_check_term:nn #1 #2 {}
\cs_set:Npn \_stex_sref_do_aux:n #1 { #1 }
\bool_set_false:N \c_stex_check_terms_bool
\bool_set_false:N \l__stex_annotate_do_output_bool
\_stex_set_meta_archive:
\tl_set:Ne \l_stex_current_document_uri {
  {\tl_to_str:n{FTML/meta}}{}
  {\tl_to_str:n{Metatheory}}{\tl_to_str:n{en}}{}
}
\_stex_module_setup_top_nosig:n{Metatheory}
\g_stex_every_module_tl

\symdef{typeof}[name=type~of,args=1]{\maincomp{\mathrm{tp}}\dobrackets{#1}}

\symdef{commented}[args=2]{#2 \;(#1)}
\notation{commented}[inv]{#2}

\symdef{of~type}[args=ii,invisible]{#1}
\notation{of~type}[colon]{#1 \mathbin{\comp{:}} #2}

\symdef{apply}[args=ia,prec=0;\infprec x\infprec]{#1\mathopen{\comp(} #2 \mathclose{\comp)}}
\notation{apply}[lambda]{#1\; \argsep{#2}{\;}}
\notation{apply}[infixop]{\argsep{#2}{\mathbin{#1}}\STEXinvisible{#1}}
\notation{apply}[infixrel]{\argsep{#2}{\mathrel{#1}}\STEXinvisible{#1}}

\symdef{autoprove}[name=prove automatically]{\mathrm{automatically\ proved}}

% structures
\symdef{module~type}[args=i,op=\mathtt{MOD}]
  {\mathopen{\comp{\mathtt{MOD}(}}#1\mathclose{\comp{)}}}}
\symdef{module~type~merge}[args=a,op=\oplus]
  {\argsep{#1}{\mathbin{\comp{\oplus}}}}
\symdef{anonymous~record}[args=a]
```

```
  {\mathopen{\comp{[[}}#1\mathclose{\comp{]]}}}}
\symdef{record~field}[args=2]{#1\comp{.}#2}
\symdecl*{record~type}

\symdecl{mathstruct}[name=mathematical~structure,args=a] % TODO
\notation{mathstruct}[angle,prec=nobrackets]
  {\mathopen{\comp\langle} #1 \mathclose{\comp\rangle}}
\notation{mathstruct}[parens,prec=nobrackets]
  {\mathopen{\comp(} #1 \mathclose{\comp)}}

% sequences
\symdef{ellipses}[ldots]{\ldots}
\symdef{sequence~expression}[comma,args=a]{#1}
\symdef{sequence~type}[args=1]{#1^{\comp\ast}}
\symdef{ranged~sequence~type}[args=ia]{{#1}\c_math_subscript_token{\dobrackets{#2}}}
\symdef{sequence~range}[inv,args=a]{#1}
\symdef{sequence~map}[args=ai]{
  \comp{\mathrm{map}}\mathopen{\comp{(}}#1\mathpunct{\comp{,}}
  #2\mathclose{\comp{)}}
}
\symdef{fold~right}[args=aibbi]{
  \maincomp{\mathrm{fold}}\dobrackets{#2\mathpunct{\comp;}#1\mathpunct{\comp;}
    \dobrackets{#3,#4} \mathrel{\comp\mapsto} #5
  }
}

\symdef{fold}[args=abbi]{
  \maincomp{\mathrm{fold}}\dobrackets{#1\mathpunct{\comp;}
    \dobrackets{#2,#3} \mathrel{\comp\mapsto} #4
  }
}

\symdecl{bind}[args=Bi,assoc=pre]
\notation{bind}[depfun,prec=nobrackets,op=(\cdot)\;\to\;\cdot]
  {\mathopen{\comp(} #1 \mathclose{\comp{)}\mathbin{\comp{\to}}} #2}
\notation{bind}[forall]{\comp\forall #1.\;#2}
\notation{bind}[Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}

\symdef{implicit~bind}[args=Bi,assoc=pre]{\mathopen{\comp\{} #1 \mathclose{\comp{\}\c_math_
\symdef{apply~implicits}[args=ia,prec=nobrackets]{
  \underbrace{#1}_{#2}
}

\symdecl{arrow}[args=ai,assoc=pre]
\notation{arrow}[single]{
  \argsep{#1}{\mathbin{\comp\times}}
  \mathrel{\maincomp\to} #2
}
\notation{arrow}[double]{
  \argsep{#1}{\mathbin{\comp\times}}
  \mathrel{\maincomp\Rightarrow} #2
}

\symdef{intsecttp}[intersection type,args=a,sqcap,prec=0;-10]{ \argsep{#1}{\maincomp{\mathr
```

```
8223
8224 \symdef{bindin}[args=Bi]{ \mathop{\maincomp{\amalg}}_{#1}#2 }
8225
8226 \symdecl*{integer~literal}
8227 \notation{integer~literal}{\mathbb Z}
8228
8229 \symdecl*{ordinal}
8230 \notation{ordinal}{\mathtt{Ord}}
8231
8232 % propositions
8233 \symdef{prop}[name=proposition]{\mathtt{Prop}}
8234 \symdef{judgment~holds}[args=i,role=judgment]{\comp\vdash\;#1}
8235
8236 % any object
8237 \symdef{object}{\mathtt{Obj}}
8238
8239 \symdef{letin}[name=let in,args=Bi,role=let]{\maincomp{\mathrm{let}}\ #1\ \comp{\mathrm{in}}
8240
8241 % TODO DELETE
8242 \symdef{aseqdots}[args=a,prec=nobrackets]
8243   {#1\comp{,\ldots}}%{##1\comp,##2}
8244 \symdef{aseqfromto}[args=ai,prec=nobrackets]
8245   {#1\comp{,\ldots,}#2}%{##1\comp,##2}
8246 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]
8247   {#1\comp{,\ldots,}#2\comp{,\ldots,}#3}%{##1\comp,##2}
```

Closing:

```
8248 \global \let \l_stex_metatheory_uri \l_stex_current_module_uri
8249 \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
8250 \stex_close_module:
8251 \group_end:
```

# Chapter 38

# Others

```
⟨@@=stex_others⟩

\cs_new_protected:Npn \MSC #1 {}

\_stex_persist_read_now:
\cs_new_protected:Nn \__stex_others_newlabel:n {
  \tl_gset:cn{__stex_sref_aux_sym_ \tl_to_str:n{#1}}{X}
  \exp_args:Ne\__stex_others_old_newlabel:{\tl_to_str:n{#1}}
}
\AtBeginDocument{
  \iow_now:Nn \@auxout {
    \ExplSyntaxOn
    \let\__stex_others_old_newlabel:\newlabel
    \let\newlabel\__stex_others_newlabel:n
    \ExplSyntaxOff
  }
}
```

Inference Rules

```
\NewDocumentCommand \FTMLrule {m m}{
  \tl_if_empty:nTF{#2}{
    \seq_clear:N \l_tmpa_seq
  }{
    \seq_set_split:Nnn \l_tmpa_seq , {#2}
  }
  \int_zero:N \l_tmpa_int
  \stex_annotate_invisible:n{\hbox{
    $
    \stex_annotate:nn{data-ftml-inferencerule={#1}}{
      \_stex_annotate_force_break:n{~
        \seq_if_empty:NF \l_tmpa_seq {
          \seq_map_inline:Nn \l_tmpa_seq {
            \int_incr:N \l_tmpa_int
            \stex_annotate:nn{
              data-ftml-argmode=i,
              data-ftml-arg={\int_use:N \l_tmpa_int}
            }{ ##1 }
          }
        }
```

```
        ~}
      }$
    }}
}
\stex_deactivate_macro:Nn \FTMLrule {module~environments}
\stex_every_module:n{\stex_reactivate_macro:N \FTMLrule}
```

# Chapter 39

# Additional Packages

## 39.1 Implementation: The notesslides Package

### 39.1.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
⟨∗cls⟩
⟨@@=notesslides⟩
\ProvidesExplClass{notesslides}{2026/06/24}{4.1.0}{notesslides Class}
\RequirePackage{l3keys2e}

\str_const:Nn \c__notesslides_class_str {article}

\keys_define:nn{notesslides / cls}{
  class   .str_set_x:N = \c__notesslides_class_str,
  notes   .bool_set:N  = \c_notesslides_notes_bool ,
  slides  .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
  %docopt  .str_set_x:N = \c__notesslides_docopt_str,
  unknown .code:n      = {
    \PassOptionsToClass{\CurrentOption}{beamer}
    \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
    \PassOptionsToPackage{\CurrentOption}{notesslides}
    \PassOptionsToPackage{\CurrentOption}{stex}
  }
}
\ProcessKeysOptions{ notesslides / cls }

\RequirePackage{stex}
\stex_if_html_backend:T {
  \bool_set_true:N\c_notesslides_notes_bool
}

\bool_if:NTF \c_notesslides_notes_bool {
  \PassOptionsToPackage{notes=true}{notesslides}
  \message{notesslides.cls:~Formatting~document~in~notes~mode}
```

```
}{
  \PassOptionsToPackage{notes=false}{notesslides}
  \message{notesslides.cls:~Formatting~document~in~slides~mode}
}

\bool_if:NTF \c_notesslides_notes_bool {
  \LoadClass{\c__notesslides_class_str}
}{
  \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
  %\newcounter{Item}
  %\newcounter{paragraph}
  %\newcounter{subparagraph}
  %\newcounter{Hfootnote}
}
\RequirePackage{notesslides}
```
⟨/cls⟩

now we do the same for the `notesslides` package.

⟨∗package⟩
```
\ProvidesExplPackage{notesslides}{2026/06/24}{4.1.0}{notesslides Package}
\RequirePackage{l3keys2e}

\keys_define:nn{notesslides / pkg}{
  notes           .bool_set:N   = \c_notesslides_notes_bool ,
  slides          .code:n       = { \bool_set_false:N \c_notesslides_notes_bool },
  sectocframes    .bool_set:N   = \c_notesslides_sectocframes_bool ,
  topsect         .str_set_x:N  = \c_notesslides_topsect_str,
  unknown         .code:n       = {
    \PassOptionsToPackage{\CurrentOption}{stex}
    \PassOptionsToPackage{\CurrentOption}{tikzinput}
  }
}
\ProcessKeysOptions{ notesslides / pkg }

\RequirePackage{stex}
\stex_if_html_backend:T {
  \bool_set_true:N\c_notesslides_notes_bool
}

\cs_set:Npn \sectiontitleemph #1 {
  \textbf{\Large #1}
}

\newif\ifnotes
\bool_if:NTF \c_notesslides_notes_bool {
  \notestrue
  \PassOptionsToPackage{dvipsnames,svgnames}{xcolor}
  \RequirePackage[noamsthm,hyperref]{beamerarticle}
  \RequirePackage{mdframed}
  \str_if_empty:NTF \c_notesslides_topsect_str{
    %\setsectionlevel{section}
  } {
    \exp_args:No \setsectionlevel \c_notesslides_topsect_str
  }
```

```
}{
  \notesfalse

  \cs_new_protected:Nn \__notesslides_do_sectocframes: {
    \cs_set_protected:Nn \__notesslides_do_label:n {
      \str_case:nnF{##1}{
        {part}    {
          \tl_set:Nx\l__notesslides_num{\thepart}
          \tl_set:cx{@ @ label}{
            \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{}
        }
        {chapter} {
          \tl_set:Nx\l__notesslides_num{\thechapter}
          \tl_set:cx{@ @ label}{
            \cs_if_exist:NTF\chaptertitlename{\exp_not:N\chaptertitlename}{\exp_not:N\chapt
        }
        {section} {
          \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection
          \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
        }
        {subsection} {
          \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection
          \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
        }
        {subsubsection} {
          \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection
          \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
        }
        {paragraph} {
          \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection
          \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
        }
      }{
        \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.\
        \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
      }
    }
    \cs_set_protected:Nn \_sfragment_do_level:nn {
      \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
      \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
      \__notesslides_do_label:n{##1}
      \pdfbookmark[\int_use:N \l_stex_current_section_level_int]{\l__notesslides_num\ ##2}{
      \begin{frame}[noframenumbering]
        \vfill\centering
        \sectiontitleemph{
          \use:c{@ @ label} ##2
        }
      \end{frame}
      \int_incr:N \l_stex_current_section_level_int
      \str_set:Nn \l_stex_current_section_level_str{##1}
    }
  }

  \cs_set_protected:Nn \_stex_titlefragment: {
```

```
    \begin{frame}[noframenumbering]\maketitle\end{frame}
    \let\maketitle\relax
  }

  \AtBeginDocument{
    \str_if_empty:NTF \c_notesslides_topsect_str {
      \setsectionlevel{section}
    } {
      \exp_args:No \setsectionlevel \c_notesslides_topsect_str
      \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
        \__notesslides_define_chapter:
      }{
        \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
          \__notesslides_define_chapter:
          \__notesslides_define_part:
        }
      }
    }
  }

  \bool_if:NT \c_notesslides_sectocframes_bool {
    \__notesslides_do_sectocframes:
  }
}

\cs_new_protected:Nn \__notesslides_define_chapter: {
  \cs_if_exist:NF \chaptername {
    \cs_set_protected:Npn \chaptername {Chapter}
  }
  \cs_if_exist:NF \chapter {
    \cs_set_protected:Npn \chapter {INVALID}
  }
  \cs_if_exist:NF \c@chapter {
    \newcounter{chapter}\counterwithin*{section}{chapter}
  }
}

\cs_new_protected:Nn \__notesslides_define_part: {
  \cs_if_exist:NF \partname {
    \cs_set_protected:Npn \partname {Part}
  }
  \cs_if_exist:NF \part {
    \cs_set_protected:Npn \part {INVALID}
  }
  \cs_if_exist:NF \c@part {
    \newcounter{part}\counterwithin*{chapter}{part}
  }
}
```

`\prematurestop` We initialize `\afterprematurestop`, and provide `\prematurestop@endsfragment` which looks up `\sfragment@level` and recursively ends enough `{sfragment}`s.

```
\def \c__notesslides_document_str{document}
\newcommand\afterprematurestop{}
\def\prematurestop@endsfragment{
```

```
  \unless\ifx\@currenvir\c__notesslides_document_str
    \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandaft
    \expandafter\prematurestop@endsfragment
  \fi
}
\providecommand\prematurestop{
  \stex_if_html_backend:F{
  \message{Stopping~sTeX~processing~prematurely}
  \prematurestop@endsfragment
  \afterprematurestop
  \end{document}
  }
}
```

(*End of definition for* `\prematurestop`. *This function is documented on page 111.*)

### 39.1.2 Notes and Slides

For the notes case, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class.

```
\bool_if:NT \c_notesslides_notes_bool {
  \renewcommand\usetheme[2][]{\usepackage[#1]{beamertheme#2}}
}
\NewDocumentCommand \libusetheme {O{} m} {
  \libusepackage[#1]{beamertheme#2}
}
```

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

```
\newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
\newlength{\slideheight}\setlength{\slideheight}{9cm}
```

We first set up the slide boxes in `notes` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
\ifnotes

\newlength{\slideframewidth}
\setlength{\slideframewidth}{1.5pt}
```

`frame` (*env.*) We first define the keys.

```
\cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
  \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
    \bool_set_true:N #1
  }{
    \bool_set_false:N #1
  }
}

\stex_keys_define:nnnn{notesslides / frame}{
  \str_clear:N \l__notesslides_frame_label_str
  \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
  \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
  \bool_set_true:N \l__notesslides_frame_fragile_bool
  \bool_set_true:N \l__notesslides_frame_shrink_bool
```

```
  \bool_set_true:N \l__notesslides_frame_squeeze_bool
  \bool_set_true:N \l__notesslides_frame_t_bool
}{
  label               .str_set_x:N  = \l__notesslides_frame_label_str,
  allowframebreaks    .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
  },
  allowdisplaybreaks  .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
  },
  fragile             .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
  },
  shrink              .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
  },
  squeeze             .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
  },
  t                   .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
  },
  unknown   .code:n       = {}
}{}
```

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

```
\cs_new_protected:Nn \__notesslides_setup_itemize: {
  \def\itemize@level{outer}
  \def\itemize@outer{outer}
  \def\itemize@inner{inner}
  %\newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
  \renewenvironment{itemize}{
    \ifx\itemize@level\itemize@outer
      \def\itemize@label{$\rhd$}
    \fi
    \ifx\itemize@level\itemize@inner
      \def\itemize@label{$\scriptstyle\rhd$}
    \fi
    \begin{list}
    {\itemize@label}
    {\setlength{\labelsep}{.3em}
      \setlength{\labelwidth}{.5em}
      \setlength{\leftmargin}{1.5em}
    }
    \edef\itemize@level{\itemize@inner}
  }{
    \end{list}
  }
}
```

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

```
\stex_if_html_backend:TF {
  %\stex_css_literal:n{
  %  .stex-frame {
```

```
  %    background-color:#fff;
  %    margin:5px ~ 0;
  %    border:2px ~ solid ~ #000;
  %    border-radius:5px;
  %    padding:5px;
  %    padding-right:10px;
  %    width: var(--rustex-curr-width);
  %    display:block;
  %  }
  %}

  \cs_new_protected:Nn \__notesslides_frame_box_begin: {
    \vbox\bgroup
    \begin{stex_annotate_env}{data-ftml-slide={}}%{class:stex-frame={}}
      \mdf@patchamsthm\notesslidesfont
  }
  \cs_new_protected:Nn \__notesslides_frame_box_end: {
    %^^A \notesslides@slidelabel
    \medskip\par\hbox to \textwidth{\tiny\notesslidesfooter}
    \end{stex_annotate_env}\egroup
  }
}{
  \cs_new_protected:Nn \__notesslides_frame_box_begin: {
    \begin{mdframed}[
      linewidth=\slideframewidth,
      skipabove=1ex,
      skipbelow=1ex,
      userdefinedwidth=\slidewidth,
      align=center,
      %usetwoside=false
    ]\notesslidesfont
  }
  \cs_new_protected:Nn \__notesslides_frame_box_end: {
    \medskip\par\noindent\tiny\notesslidesfooter%^^A\notesslides@slidelabel
    \end{mdframed}
  }
}
```

We define the environment, read them, and construct the slide number and label.

```
\renewenvironment{frame}[1][]{
  \stex_keys_set:nn{notesslides / frame}{#1}
  \stepcounter{framenumber}
  \renewcommand\newpage{\addtocounter{framenumber}{1}}
  \def\@currentlabel{\theframenumber}
  \str_if_empty:NF \l__notesslides_frame_label_str {
    \label{\l__notesslides_frame_label_str}
  }
  \__notesslides_setup_itemize:
  \__notesslides_frame_box_begin:
}{
  \__notesslides_frame_box_end:
}
```

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

\frametitle

```
\renewcommand{\frametitle}[1]{
  \stexdoctitle { #1 }
  \stex_if_html_backend:TF{
    \notesslidestitleemph{\noindent\stex_annotate:nn{data-ftml-title={}}{#1}}\par
  }{
    \notesslidestitleemph{#1}
  }
  \medskip
}
```

(*End of definition for* \frametitle*. This function is documented on page* ??.)

\pause

```
\newcommand\pause{}
```

(*End of definition for* \pause*. This function is documented on page* ??.)

We redefine the columns and column environments:

```
\renewenvironment{columns}[1][]{
  \par\noindent
  \begin{minipage}
  \slidewidth\centering\leavevmode
%  \stex_if_html_backend:T{
%    \cs_if_exist:NT \rustex_if:T {
%      \rustex_if:T {\par
%        \rustex_direct_HTML:n{<table><tr><td>}
%      }
%    }
%  }
}{
%  \stex_if_html_backend:T{
%    \cs_if_exist:NT \rustex_if:T {
%      \rustex_if:T {\par
%        \rustex_direct_HTML:n{</td></tr></table>}
%      }
%    }
%  }
  \end{minipage}\par\noindent
}
\newsavebox\columnbox
\renewenvironment<>{column}[2][]{
  \begin{lrbox}{\columnbox}
%    \stex_if_html_backend:T{
%      \cs_if_exist:NT \rustex_if:T {
%        \rustex_if:T {\par
%          \rustex_direct_HTML:n{</td><td>}
%        }
%      }
%    }
    \begin{minipage}{#2}
}{
  \end{minipage}
%  \stex_if_html_backend:T{
%    \cs_if_exist:NT \rustex_if:T {
```

```
%       \rustex_if:T {\par
%         \rustex_direct_HTML:n{</td><td>}
%       }
%     }
%  }
  \end{lrbox}\usebox\columnbox
}
\fi
```

### 39.1.3 Environment and Macro Patches

The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment to produce no output.

```
\cs_if_exist:cTF{note}{
  \bool_if:NTF \c_notesslides_notes_bool {
    \renewenvironment{note}{\ignorespaces}{}
  }{
    \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup}{\egroup}
  }
}{
  \bool_if:NTF \c_notesslides_notes_bool {
    \newenvironment{note}{\ignorespaces}{}
  }{
    \newenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup}{\egroup}
  }
}
```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```
\cs_new_protected:Nn \__notesslides_notes_env:nnnn {
  \bool_if:NTF \c_notesslides_notes_bool {
    \newenvironment{#1}#2{#3}{#4}
  }{
    \newenvironment{#1}#2{
      \cs_set:Npn \__notesslides_eat: ####1 \end ####2 {
        \str_if_eq:nnTF{#1}{####2}{
          \end{#1}
        }{
          \__notesslides_eat:
        }
      }
      \__notesslides_eat:
      %\setbox\l_tmpa_box\vbox\bgroup#3
    }{
      %#4\egroup
    }
  }
}

\__notesslides_notes_env:nnnn{nparagraph}{[1][]}{\begin{sparagraph}[#1]}{\end{sparagraph}}
\__notesslides_notes_env:nnnn{nfragment}{[2][]}{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
\__notesslides_notes_env:nnnn{ndefinition}{[1][]}{\begin{sdefinition}[#1]}{\end{sdefinition
```

```
\__notesslides_notes_env:nnnn{nassertion}{[1][]}{\begin{sassertion}[#1]}{\end{sassertion}}
\__notesslides_notes_env:nnnn{nproof}{[2][]}{\begin{sproof}[#1]{#2}}{\end{sproof}}
\__notesslides_notes_env:nnnn{nexample}{[1][]}{\begin{sexample}[#1]}{\end{sexample}}

\RequirePackage{graphicx}

\NewDocumentCommand\frameimage{s O{} m}{
  \IfBooleanTF #1 {
    \begin{frame}[plain]
  }{
    \begin{frame}
  }
    \bool_if:NTF \c_notesslides_notes_bool {
      \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
    }{
      \slidewidth=\textwidth\relax
    }
    \def\Gin@ewidth{}\setkeys{Gin}{#2}
    \tl_if_empty:NTF \Gin@ewidth {
      \mhgraphics[width=\slidewidth,#2]{#3}
    }{
      \mhgraphics[#2]{#3}
    }
  \end{frame}
}
```

hacking inputref:

`\inputref*`

```
\cs_set_eq:NN\__notesslides_inputref:\inputref
\cs_set_protected:Npn\inputref{\@ifstar\ninputref\__notesslides_inputref:}
\bool_if:NTF \c_notesslides_notes_bool {
  \newcommand\ninputref[2][]{
    \__notesslides_inputref:[#1]{#2}
  }
}{
  \newcommand\ninputref[2][]{}
}
```

(*End of definition for* `\inputref*`. *This function is documented on page 110.*)

### 39.1.4 Styling Across Notes/Slides

```
\def\notesslidestitleemph#1{
  {\Large\bf\sf#1}
  \vskip0.1\baselineskip
  \leaders\vrule width \textwidth
  \vskip0.4pt%
  \nointerlineskip
}

\def\notesslidesfooter{}

\let\notesslidesfont\sffamily
```

### 39.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```
\stex_if_do_html:TF{
  \NewDocumentEnvironment{slideshow}{}{
    \begin{stex_annotate_env}{data-ftml-slideshow={}}
    \cs_set_protected:Npn \nextslide ##1 {
      \begin{stex_annotate_env}{data-ftml-slideshow-slide={}} ##1
      \end{stex_annotate_env}
    }
    \cs_set_protected:Npn \lastslide ##1 {
      \begin{stex_annotate_env}{data-ftml-slideshow-slide={}} ##1
      \end{stex_annotate_env}
    }
  }{
    \end{stex_annotate_env}
  }
}{
  \int_new:N \l__notesslides_slideshow_counter_int
  \NewDocumentEnvironment{slideshow}{}{
    \int_zero:N \l__notesslides_slideshow_counter_int
    \cs_set_protected:Npn \nextslide ##1 {
      \int_incr:N \l__notesslides_slideshow_counter_int
      \only<\int_use:N \l__notesslides_slideshow_counter_int>{##1}
    }
    \cs_set_protected:Npn \lastslide ##1 {
      \int_incr:N \l__notesslides_slideshow_counter_int
      \only<\int_use:N \l__notesslides_slideshow_counter_int ->{##1}
    }
  }{

  }
}

\bool_if:NT \c_notesslides_notes_bool {
  \def\author{\@dblarg\ns@author}
  \long\def\ns@author[#1]#2{%
    \tl_if_empty:nTF{#1}{
      \def\beamer@shortauthor{#2}
    }{
      \def\beamer@shortauthor{#1}
    }
    \def\@author{#2}
  }
  \def\title{\@dblarg\ns@title}
  \long\def\ns@title[#1]#2{%
    \tl_if_empty:nTF{#1}{
      \def\beamer@shorttitle{#2}
    }{
      \def\beamer@shorttitle{#1}
    }
    \def\@title{#2}
    \stexdoctitle{#2}
```

```
  }
  \def\insertshortauthor{
    \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
  }
  \def\insertshorttitle{
    \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
  }
  \stex_if_html_backend:TF{
    \def\insertframenumber{\stex_annotate:nn{data-ftml-slide-number={}}{}}
  }{
    \def\insertframenumber{\@arabic\c@framenumber}
  }
  \def\insertshortdate{\today}
}
```

### 39.1.6 TODO Excursions

\excursion The excursion macros are very simple, we define a new internal macro \excursionref and use it in \excursion, which is just an \inputref that checks if the new macro is defined before formatting the file in the argument.

```
\gdef\printexcursions{}



\stex_if_html_backend:TF{
  \newcommand\excursionref[4][]{% repos, label, path, text
    \begin{sparagraph}[title=Excursion]
      #4~
      \stex_in_archive:nn{#1}{
        \stex_document_uri_from_archive_file:Nn \l__notesslides_uri {#3}
        \exp_args:Ne\href{\stex_use_document_uri:N\l__notesslides_uri}{here}
      }
    \end{sparagraph}
  }
  \def\excursion{\excursionref}
}{
  \newcommand\excursionref[4][]{% repos, label, path, text
    \bool_if:NT \c_notesslides_notes_bool {
      \begin{sparagraph}[title=Excursion]
        #4~\sref[fallback=the appendix]{#2}.
      \end{sparagraph}
    }
  }
  \newcommand\activate@excursion[2][]{
    \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}
  }
  \newcommand\excursion[4][]{% repos, label, path, text
    \bool_if:NT \c_notesslides_notes_bool {
      \activate@excursion[#1]{#3}
      \excursionref[#1]{#2}{#3}{#4}
    }
  }

```

```
}
```

(*End of definition for* \excursion. *This function is documented on page 112.*)

\excursiongroup

```
\keys_define:nn{notesslides / excursiongroup }{
  id        .str_set_x:N  = \l__notesslides_excursion_id_str,
  intro     .tl_set:N     = \l__notesslides_excursion_intro_tl,
  archive   .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
}
\cs_new_protected:Nn \__notesslides_excursion_args:n {
  \tl_clear:N \l__notesslides_excursion_intro_tl
  \str_clear:N \l__notesslides_excursion_id_str
  \str_clear:N \l__notesslides_excursion_mhrepos_str
  \keys_set:nn {notesslides / excursiongroup }{ #1 }
}


\stex_if_html_backend:TF{
  \newcommand\excursiongroup[1][]{}
}{
  \newcommand\excursiongroup[1][]{
    \__notesslides_excursion_args:n{ #1 }
    \tl_if_empty:NF\printexcursions
    {\IfInputref{}{\begin{note}
      \begin{sfragment}{Excursions}% TODO pass on id
        \ifdefempty\l__notesslides_excursion_intro_tl{}{
          \exp_args:NNe \use:nn \inputref{[\l__notesslides_excursion_mhrepos_str]{
            \l__notesslides_excursion_intro_tl
          }}
        }
        \printexcursions%
      \end{sfragment}
    \end{note}}}
  }
}

\ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
```

(*End of definition for* \excursiongroup. *This function is documented on page 112.*)

```
\prop_new:N \g__notesslides_variables_prop
\cs_set_protected:Npn \setSGvar #1 #2 {
  \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
}
\cs_set_protected:Npn \useSGvar #1 {
  \prop_item:Nn \g__notesslides_variables_prop {#1}
}
\cs_set_protected:Npn \ifSGvar #1 #2 #3 {
  \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
    \PackageError{document-structure}
    {The sTeX Global variable #1 is undefined}
    {set it with \protect\setSGvar}\TODO better error
  }
  \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
```

```
}


⟨/package⟩
```

## 39.2 Implementation: The problem Package

### 39.2.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
⟨*package⟩
⟨@@=problems⟩
\ProvidesExplPackage{problem}{2026/06/24}{4.1.0}{Semantic Markup for Problems}
\RequirePackage{l3keys2e}

\keys_define:nn { problem / pkg }{
  notes     .default:n    = { true },
  notes     .bool_set:N   = \c__problems_notes_bool,
  gnotes    .default:n    = { true },
  gnotes    .bool_set:N   = \c__problems_gnotes_bool,
  hints     .default:n    = { true },
  hints     .bool_set:N   = \c__problems_hints_bool,
  solutions .default:n    = { true },
  solutions .bool_set:N   = \c__problems_solutions_bool,
  pts       .default:n    = { true },
  pts       .bool_set:N   = \c__problems_pts_bool,
  min       .default:n    = { true },
  min       .bool_set:N   = \c__problems_min_bool,
  %boxed     .default:n    = { true },
  %boxed     .bool_set:N   = \c__problems_boxed_bool,
  test     .default:n    = { true },
  test     .bool_set:N   = \c__problems_test_bool,
  unknown     .code:n       = {
    \PassOptionsToPackage{\CurrentOption}{stex}
  }
}
\newif\ifsolutions

\ProcessKeysOptions{ problem / pkg }
\bool_if:NTF \c__problems_solutions_bool {
  \solutionstrue
}{
  \solutionsfalse
}
\newif\ifintest
\bool_if:NTF \c__problems_test_bool {
  \intesttrue
}{
  \intestfalse
}

```

```
\RequirePackage{stex}

\ifintest
  \AtBeginDocument{
    \def\symrefemph#1{#1}
    \def\compemph#1{#1}
    \def\varemph#1{#1}
  }
\fi

\newcommand\precondition[2]{
  \stex_get_symbol:n{#2}
  \tl_if_empty:NF \l_stex_get_symbol_uri {
    \str_case:nnTF {#1}{
      {remember}{}
      {understand}{}
      {analyze}{}
      {evaluate}{}
      {apply}{}
      {create}{}
    }{
      \ifstexhtml\else\bool_if:NT\c_stex_metadata_bool {
        \marginpar{\tiny \textbf{Precondition:}~#1~
          \exp_args:Ne\symrefemph@uri{\stex_symbol_uri_name:N \l_stex_get_symbol_uri}{\l_st
        }
      }\fi
      \stex_annotate_invisible:nn{
        data-ftml-preconditionsymbol={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
        data-ftml-preconditiondimension={#1}
      }{}
    }{\errmessage{Unknown~cognitive~dimension~#1}}
  }
}
\newcommand\objective[2]{
  \stex_get_symbol:n{#2}
  \tl_if_empty:NF \l_stex_get_symbol_uri {
    \str_case:nnTF {#1}{
      {remember}{}
      {understand}{}
      {analyze}{}
      {evaluate}{}
      {apply}{}
      {create}{}
    }{
      \ifstexhtml\else\bool_if:NT\c_stex_metadata_bool {
        \marginpar{\tiny \textbf{Objective:}~#1~
        \exp_args:Ne\symrefemph@uri{\stex_symbol_uri_name:N \l_stex_get_symbol_uri}{\l_stex
        }
      }\fi
      \stex_annotate_invisible:nn{
        data-ftml-objectivesymbol={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
        data-ftml-objectivedimension={#1}
      }{}
    }{\errmessage{Unknown~cognitive~dimension~#1}}
```

```
  }
}
```

`\problem@kw@*` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
\AddToHook{begindocument}{
  \ExplSyntaxOn\makeatletter
  \input{problem-english.ldf}
  \ltx@ifpackageloaded{babel}{
   \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
        \input{problem-ngerman.ldf}
      }
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
        \input{problem-finnish.ldf}
      }
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
        \input{problem-french.ldf}
      }
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
        \input{problem-russian.ldf}
      }
  }{}
  \makeatother\ExplSyntaxOff
}
```

(*End of definition for* `\problem@kw@*`. *This function is documented on page* **??**.)

### 39.2.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
\bool_new:N \l_stex_key_autogradable_bool
\stex_keys_define:nnnn{ problem }{
  \tl_clear:N \l_stex_key_pts_tl
  \tl_set:Nn \l_stex_key_min_tl 0
  \str_clear:N \l_stex_key_name_str
  \str_clear:N \l_stex_key_mhrepos_str
  \bool_set_false:N \l_stex_key_autogradable_bool
}{
  pts     .tl_set:N     = \l_stex_key_pts_tl,
  min     .tl_set:N     = \l_stex_key_min_tl,
  name    .str_set:N    = \l_stex_key_name_str,
  autogradable .bool_set:N = \l_stex_key_autogradable_bool,
  archive  .code:n = {},
  %archive .str_set:N    = \l_stex_key_mhrepos_str,
  %creators .code:n = {}
  %imports .tl_set:N     = \l__problems_prob_imports_tl,
  %refnum  .int_set:N    = \l__problems_prob_refnum_int,
}{id,title,style,uses}
```

Then we set up a counter for problems.

\numberproblemsin

```
\newcounter{sproblem}[section]
\newcommand\numberproblemsin[1]{
  \@addtoreset{sproblem}{#1}
  \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
}
\numberproblemsin{section}
%\def\theplainsproblem{\arabic{sproblem}}
%\def\thesproblem{\thesection.\theplainsproblem}
```

(*End of definition for* `\numberproblemsin`. *This function is documented on page* **??**.)

sproblem (*env.*)

```
\cs_new:Nn \__problems_activate_macros: {
  \stex_reactivate_macro:N \solution
  \stex_reactivate_macro:N \mcb
  \stex_reactivate_macro:N \scb
  \stex_reactivate_macro:N \fillinsol
  \stex_reactivate_macro:N \hint
  \stex_reactivate_macro:N \exnote
  \stex_reactivate_macro:N \gnote
}

\fp_new:N \g_problem_total_pts_fp
\fp_new:N \g_problem_total_min_fp

\bool_new:N \l__problems_in_problem_bool
\bool_new:N \l__problems_has_pts_bool
\bool_new:N \l__problems_has_min_bool
\bool_set_false:N \l__problems_in_problem_bool
\stex_new_stylable_env:nnnnnnn {problem} {O{}}{
  \bool_if:NT \l__problems_in_problem_bool {
    \msg_error:nn{stex}{error/nestedproblem}
  }
  \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
    \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}
    \exp_args:Nno \stex_keys_set:nn{problem}{
      \l_problem_inputproblem_keys_tl
    }
  }{
    \stex_keys_set:nn{problem}{#1}
  }
  \refstepcounter{sproblem}

  \stex_if_do_html:T {
    \str_if_empty:NT \l_stex_key_name_str {
      \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
      \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
    }
    \exp_args:Nne \begin{stex_annotate_env} {
      data-ftml-problem={\l_stex_key_name_str},
      %data-ftml-language={ \l_stex_current_language_str},
      data-ftml-autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
      \tl_if_empty:NF \l_stex_key_pts_tl{,data-ftml-problempoints={\l_stex_key_pts_tl}}
```

```
    }
    \noindent \stex_annotate:nn{data-ftml-title={}}{\_stex_annotate_force_break:n{\l_stex_k
    \tl_if_empty:NF \l_stex_key_title_tl {
      \exp_args:No \stexdoctitle \l_stex_key_title_tl
    }
    \_stex_annotate_force_break:n{}
  }
  \tl_set_eq:NN \thistitle \l_stex_key_title_tl
  \tl_if_empty:NT \l_stex_key_pts_tl { \tl_set:Nn \l_stex_key_pts_tl{0} }


  \bool_set_true:N \l__problems_in_problem_bool
  \tl_set_eq:NN \l__problems_pts_tl \l_stex_key_pts_tl
  \tl_set_eq:NN \l__problems_min_tl \l_stex_key_min_tl
  \tl_if_eq:NnTF \l__problems_pts_tl {0}
    {\bool_set_false:N \l__problems_has_pts_bool}
    {\bool_set_true:N \l__problems_has_pts_bool}
  \tl_if_eq:NnTF \l__problems_min_tl {0}
    {\bool_set_false:N \l__problems_has_min_bool}
    {\bool_set_true:N \l__problems_has_min_bool}
  \int_gzero:N \g__problems_subproblem_int

  \stex_if_do_html:F\stex_style_apply:
  \_stex_do_id:
  \__problems_activate_macros:
}{
  \fp_gadd:Nn \g_problem_total_pts_fp { \l__problems_pts_tl}
  \fp_gadd:Nn \g_problem_total_min_fp { \l__problems_min_tl}
  \__problems_record_problem:
  \stex_if_do_html:F\stex_style_apply:
  \stex_if_do_html:T{ \end{stex_annotate_env} }
}{
  \par\noindent\problemheader
  \stex_if_do_html:F{
    \bool_if:NT \c__problems_pts_bool {
      \tl_if_eq:NnF \l__problems_pts_tl {0}{
        \marginpar{\l__problems_pts_tl{}~\problem@kw@pts\smallskip}
      }
    }
    \bool_if:NT \c__problems_min_bool {
      \tl_if_eq:NnF \l__problems_min_tl {0} {
        \marginpar{\l__problems_min_tl{}~\problem@kw@minutes\smallskip}
      }
    }
  }
  \par
  \stex_ignore_spaces_and_pars:
}{
  \par\bigskip
%  \bool_if:NT \c__problems_test_bool \pagebreak
}{s}

\tl_set:Nn \problemheader {
  \stex_if_do_html:TF{
```

```
    \tl_if_empty:NF \thistitle {
      \stex_annotate:nn{data-ftml-title={}}{\textbf{\thistitle}}
    }
  }{
    \textbf{\sproblemautorefname{~}\thesproblem
      \tl_if_empty:NF \thistitle {
        {~}(\thistitle)
      }
    }
  }
}

\cs_new_protected:Nn \__problems_record_problem: {
  \exp_args:NNe \iow_now:Nn \@auxout {
    \problem@restore {\thesproblem}{\l__problems_pts_tl}{\l__problems_min_tl}
  }
}

\cs_new_protected:Npn \problem@restore #1 #2 #3 {}
```

subproblem (*env.*)

```
\int_new:N \g__problems_subproblem_int

\stex_new_stylable_env:nnnnnnn {subproblem} {O{}}{
  \stex_keys_set:nn{problem}{#1}
  \bool_if:NF \l__problems_in_problem_bool{
    \ifstexhtml\else
      \par{\bfseries WARNING~subproblem~to~be~used~in~some~problem}\par
    \fi
    \__problems_activate_macros:
    \bool_set_true:N \l__problems_in_problem_bool
    \tl_set:Nn \l__problems_pts_tl{0}
    \tl_set:Nn \l__problems_min_tl{0}
  }
  \str_if_empty:NT \l_stex_key_name_str {
    \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
    \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
  }
  \stex_if_do_html:T {
    \str_if_empty:NT \l_stex_key_name_str {
      \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
      \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
    }
    \exp_args:Nne \begin{stex_annotate_env} {
      data-ftml-subproblem={\l_stex_key_name_str},
      %data-ftml-language={ \l_stex_current_language_str},
      data-ftml-autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
      \tl_if_empty:NF \l_stex_key_pts_tl{,data-ftml-problempoints={\l_stex_key_pts_tl}}
    }
    \noindent \stex_annotate:nn{data-ftml-title={}}{\_stex_annotate_force_break:n{\l_stex_k
    \tl_if_empty:NF \l_stex_key_title_tl {
      \exp_args:No \stexdoctitle \l_stex_key_title_tl
    }
    \tl_if_empty:NF \l_stex_key_title_tl {
```

```
      \exp_args:No \stexdoctitle \l_stex_key_title_tl
    }
    \_stex_annotate_force_break:n{}
  }
  \tl_if_empty:NT \l_stex_key_pts_tl { \tl_set:Nn \l_stex_key_pts_tl{0} }
  \int_gincr:N \g__problems_subproblem_int
  \bool_if:NF \l__problems_has_pts_bool {
    \tl_gset:Ne \l__problems_pts_tl {\fp_to_decimal:n {\l__problems_pts_tl + \l_stex_key_pt
  }
  \bool_if:NF \l__problems_has_min_bool {
    \tl_gset:Ne \l__problems_min_tl {\fp_to_decimal:n {\l__problems_min_tl + \l_stex_key_mi
  }
  \stex_if_smsmode:F {\stex_if_do_html:F \stex_style_apply: }
}{
  \stex_if_do_html:TF{ \end{stex_annotate_env} }{\stex_if_smsmode:F\stex_style_apply:}
}{
  \begin{list}{}{
    \setlength\topsep{0pt}
    \setlength\parsep{0pt}
    \setlength\rightmargin{0pt}
  }\item[\int_use:N \g__problems_subproblem_int .]
  \bool_if:NT \c__problems_pts_bool {
    \bool_if:NF \l__problems_has_pts_bool {
      \marginpar{\smallskip\l_stex_key_pts_tl{}~\problem@kw@pts}
    }
  }
  \bool_if:NT \c__problems_min_bool {
    \bool_if:NF \l__problems_has_min_bool{
      \marginpar{\smallskip\l_stex_key_min_tl{}~\problem@kw@minutes}
    }
  }
  \ignorespaces
}{
  \end{list}
}{}
```

\includeproblem

```
\stex_keys_define:nnnn{ includeproblem }{
  \str_clear:N \l_stex_key_mhrepos_str
}{
  archive .str_set:N    = \l_stex_key_mhrepos_str,
  unknown .code:n = {}
}{}

\NewDocumentCommand\includeproblem{O{} m}{
  \group_begin:
  \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
  \stex_keys_set:nn{includeproblem}{#1}
  \exp_args:Nno \use:nn{\inputref[}\l_stex_key_mhrepos_str]{#2}
  \group_end:
}
```

(*End of definition for* \includeproblem. *This function is documented on page 117.*)

`solution` (*env.*)

```
\int_new:N \g_problem_id_counter
\dim_new:N \l_stex_key_testspace_dim
\stex_keys_define:nnnn{ solution }{
  \str_clear:N \l_stex_key_answerclass_str
  \dim_zero:N \l_stex_key_testspace_dim
}{
  testspace   .dim_set:N = \l_stex_key_testspace_dim,
  answerclass .str_set:N = \l_stex_key_answerclass_str
}{id,title,style}

\cs_new_protected:Nn \__problems_solution_start:n {
  \stex_keys_set:nn{ solution }{#1}
  \str_if_empty:NT \l_stex_key_id_str {
    \int_gincr:N \g_problem_id_counter
    \str_set:Nx \l_stex_key_id_str {
      SOLUTION_\int_use:N \g_problem_id_counter
    }
  }
  \stex_if_do_html:T{
    \begin{stex_annotate_env}{
      data-ftml-solution=\l_stex_key_id_str,
      data-ftml-answerclass={\l_stex_key_answerclass_str}
    }
  }
}

\stex_new_stylable_env:nnnnnnn { solution }{ O{} }{
  \stex_if_do_html:TF{
    \__problems_solution_start:n{#1}
  }{
    \ifsolutions
      \__problems_solution_start:n{#1}
      \stex_style_apply:
    \else
      \stex_keys_set:nn{ solution }{#1}
      \testspace{\l_stex_key_testspace_dim}
      \setbox\l_tmpa_box\vbox\bgroup
    \fi
  }
}{
  \stex_if_do_html:TF{
    \end{stex_annotate_env}
  }{
    \ifsolutions
      \stex_style_apply:
      \stex_if_do_html:T{
        \end{stex_annotate_env}
      }
    \else
      \egroup
    \fi
  }
}
```

```
    \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
    \noindent\emph{\problem@kw@solution\tl_if_empty:NF \l_stex_key_title_tl{
      {~}\l_stex_key_title_tl \str_if_empty:NF \l_stex_key_answerclass_str {
        (Answer Class: \l_stex_key_answerclass_str)
      }
    } :~}
  }{
    \par\rule[.3em]{\linewidth}{0.4pt}\newline
  }{}

  \stex_deactivate_macro:Nn \solution {sproblem~environments}
```

\startsolutions
\stopsolutions

```
  \cs_new_protected:Npn \startsolutions{
    \global\solutionstrue
  }
  \cs_new_protected:Npn \stopsolutions{
    \global\solutionsfalse
  }
```

(*End of definition for* \startsolutions *and* \stopsolutions*. These functions are documented on page 114.*)

hint (*env.*)

```

  \stex_keys_define:nnnn{ problemenv }{}{}{id,title,style}

  \cs_new_protected:Nn \__problems_hint_start:n {
    \stex_keys_set:nn{ problemenv }{#1}
    \str_if_empty:NT \l_stex_key_id_str {
      \int_gincr:N \g_problem_id_counter
      \str_set:Nx \l_stex_key_id_str {
        HINT_\int_use:N \g_problem_id_counter
      }
    }
    \stex_if_do_html:T{
      \begin{stex_annotate_env}{
        data-ftml-problemhint=\l_stex_key_id_str
      }
    }
  }

  \stex_new_stylable_env:nnnnnnn { hint }{ O{} }{
    \stex_if_do_html:TF{
      \__problems_hint_start:n{#1}
    }{
      \bool_if:NTF \c__problems_hints_bool {
        \__problems_hint_start:n{#1}
        \stex_style_apply:
      }{
        \setbox\l_tmpa_box\vbox\bgroup
      }
    }
  }{
```

```
  \stex_if_do_html:TF{
    \end{stex_annotate_env}
  }{
    \bool_if:NTF \c__problems_hints_bool {
      \stex_style_apply:
      \stex_if_do_html:T{
        \end{stex_annotate_env}
      }
    }{
      \egroup
    }
  }
}{
  \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
  \noindent\emph{\problem@kw@hint\tl_if_empty:NF \l_stex_key_title_tl{
    {~}\l_stex_key_title_tl
  } :~}
}{
  \par\rule[.3em]{\linewidth}{0.4pt}\newline
}{}
\stex_deactivate_macro:Nn \hint {sproblem~environments}
```

exnote (*env.*)

```
\cs_new_protected:Nn \__problems_exnote_start:n {
  \stex_keys_set:nn{ problemenv }{#1}
  \str_if_empty:NT \l_stex_key_id_str {
    \int_gincr:N \g_problem_id_counter
    \str_set:Nx \l_stex_key_id_str {
      EXNOTE_\int_use:N \g_problem_id_counter
    }
  }
  \stex_if_do_html:T{
    \begin{stex_annotate_env}{
      data-ftml-problemnote=\l_stex_key_id_str
    }
  }
}

\stex_new_stylable_env:nnnnnnn { exnote }{ O{} }{
  \stex_if_do_html:TF{
    \__problems_exnote_start:n{#1}
  }{
    \bool_if:NTF \c__problems_notes_bool {
      \__problems_exnote_start:n{#1}
      \stex_style_apply:
    }{
      \setbox\l_tmpa_box\vbox\bgroup
    }
  }
}{
  \stex_if_do_html:TF{
    \end{stex_annotate_env}
  }{
    \bool_if:NTF \c__problems_notes_bool {
```

```
      \stex_style_apply:
      \stex_if_do_html:T{
        \end{stex_annotate_env}
      }
    }{
      \egroup
    }
  }
}{
  \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
  \noindent\emph{\problem@kw@note\tl_if_empty:NF \l_stex_key_title_tl{
    {~}\l_stex_key_title_tl
  } :~}
}{
  \par\rule[.3em]{\linewidth}{0.4pt}\newline
}{}
\stex_deactivate_macro:Nn \exnote {sproblem~environments}
```

gnote (*env.*)

```
\int_new:N \l__problems_anscls_int

\cs_new_protected:Nn \__problems_gnote_start:n {
  \stex_keys_set:nn{ problemenv }{#1}
  \str_if_empty:NT \l_stex_key_id_str {
    \int_gincr:N \g_problem_id_counter
    \str_set:Nx \l_stex_key_id_str {
      GNOTE_\int_use:N \g_problem_id_counter
    }
  }

  \stex_if_do_html:TF{
    \begin{stex_annotate_env}{
      data-ftml-problemgnote=\l_stex_key_id_str
    }
  }
  \stex_style_apply:
}

\stex_new_stylable_env:nnnnnnn { gnote }{ O{} }{
  \stex_if_do_html:TF{
    \__problems_gnote_start:n{#1}
  }{
    \bool_if:NTF \c__problems_gnotes_bool {
      \__problems_gnote_start:n{#1}
    }{
      \setbox\l_tmpa_box\vbox\bgroup
    }
  }
  \stex_reactivate_macro:N \anscls
}{
  \stex_if_do_html:TF{
    \end{stex_annotate_env}
  }{
    \bool_if:NTF \c__problems_gnotes_bool {
```

```
      \stex_style_apply:
      \stex_if_do_html:T{
        \end{stex_annotate_env}
      }
    }{
      \egroup
    }
  }
}{
  \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
  \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
    {~}\l_stex_key_title_tl
  } :~}
}{
  \par\rule[.3em]{\linewidth}{0.4pt}\newline
}{}
\stex_deactivate_macro:Nn \gnote {sproblem~environments}


\stex_keys_define:nnnn{ anscls }{
  \str_clear:N \l_stex_key_pts_str
  \tl_clear:N \l_stex_key_feedback_tl
}{
  pts     .str_set:N = \l_stex_key_pts_str,
  feedback .tl_set:N = \l_stex_key_feedback_tl,
  update  .code:n    = {}
}{id}
\newcommand \anscls [2][] {
  \stex_keys_set:nn{ anscls }{#1}
  \str_if_empty:NT \l_stex_key_id_str {
    \int_incr:N \l__problems_anscls_int
    \str_set:Nx \l_stex_key_id_str {
      AC\int_use:N \l__problems_anscls_int
    }
  }
  \stex_if_do_html:TF{
    \begin{list}{}{
      \setlength\topsep{0pt}
      \setlength\parsep{0pt}
      \setlength\rightmargin{0pt}
    }\item[] \exp_args:Ne \stex_annotate:nn{
      data-ftml-answerclass={\l_stex_key_id_str}
      \str_if_empty:NF \l_stex_key_pts_str{
        ,data-ftml-answerclass-pts={\l_stex_key_pts_str}
      }
    }{
      #2~
      \tl_if_empty:NF \l_stex_key_feedback_tl{
        \stex_annotate:nn{
          data-ftml-answerclass-feedback={}
        }{\l_stex_key_feedback_tl}
      }
    }
    \end{list}
```

```
  }{
    \begin{list}{}{
      \setlength\topsep{0pt}
      \setlength\parsep{0pt}
      \setlength\rightmargin{0pt}
    }\item[\l_stex_key_id_str] #2
      \str_if_empty:NF \l_stex_key_pts_str {\par
        ~ \problem@kw@points :~\l_stex_key_pts_str
      }
      \tl_if_empty:NF \l_stex_key_feedback_tl {\par
        ~ \problem@kw@feedback :~\l_stex_key_feedback_tl
      }
    \end{list}
  }
}
\stex_deactivate_macro:Nn \anscls {gnote~environments}
```

The margin pars are reader-visible, so we need to translate

```
\def\pts#1{
  \bool_if:NT \c__problems_pts_bool {
    \stex_annotate:nn{data-ftml-problempoints={#1}}{\marginpar{#1~\problem@kw@pts}}
  }\hbox_unpack:N\c_empty_box
}
\def\min#1{
  \bool_if:NT \c__problems_min_bool {
    \stex_annotate:nn{data-ftml-problemminutes={}}{\marginpar{#1~\problem@kw@minutes}}
  }\hbox_unpack:N\c_empty_box
}
```

mcb (*env.*)

```
\stex_new_stylable_env:nnnnnnn{mcb}{O{}}{
  \stex_keys_set:nn{style}{#1}
  \cs_set:Nn \__problems_mccline:n{
    \begin{list}{}{
      \setlength\topsep{0pt}
      \setlength\parsep{0pt}
      \setlength\rightmargin{0pt}
    }\item[\problem_mcc_box_tl] ##1 \end{list}
  }

  \stex_if_do_html:T{
    \tl_set:Nn\problem_mcc_box_tl{}
    \__problems_maybe_inline:n{
      \exp_args:Nne \begin{stex_annotate_env}{
        data-ftml-multiple-choice-block={}
        \clist_if_empty:NF \l_stex_key_style_clist {,
          data-ftml-styles={\l_stex_key_style_clist}
        }
      }
    }
    \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
      \cs_set:Nn \__problems_mccline:n {##1}
    }{
      \clist_if_in:NnT \l_stex_key_style_clist {dropdown} {
```

```
      \cs_set:Nn \__problems_mccline:n {##1}
    }
    }
  }
  \stex_deactivate_macro:Nn \mcb {sproblem~environments}
  \stex_deactivate_macro:Nn \scb {sproblem~environments}
  \stex_deactivate_macro:Nn \solution {sproblem~environments}
  \stex_deactivate_macro:Nn \hint {sproblem~environments}
  \stex_deactivate_macro:Nn \exnote {sproblem~environments}
  \stex_deactivate_macro:Nn \gnote {sproblem~environments}
  \stex_reactivate_macro:N \mcc
  \stex_if_do_html:F \stex_style_apply:
}{
  \stex_if_do_html:TF{
    \__problems_maybe_inline:n{\end{stex_annotate_env}}
  }\stex_style_apply:
}{\par}{}{}

\stexstylemcb[inline]{
  {\Large[}\cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
}{{\Large]}}

\stexstylemcb[dropdown]{
  {\Large[}\cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
}{{\Large]}}

\stex_deactivate_macro:Nn \mcb {sproblem~environments}
```

we define the keys for the mcc macro

```
\cs_new_protected:Nn \__problems_do_yes_param:Nn {
  \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
    \bool_set_true:N #1
  }{
    \bool_set_false:N #1
  }
}
\stex_keys_define:nnnn{mcc}{
  \tl_clear:N \l_stex_key_feedback_tl
  \bool_set_false:N \l_stex_key_T_bool
  \tl_clear:N \l_stex_key_Ttext_tl
  \tl_clear:N \l_stex_key_Ftext_tl
}{
  feedback  .tl_set:N     = \l_stex_key_feedback_tl ,
  T         .code:n       = {\bool_set_true:N \l_stex_key_T_bool} ,
  F         .code:n       = {\bool_set_false:N \l_stex_key_T_bool} ,
  Ttext     .tl_set:N     = \l_stex_key_Ttext_tl ,
  Ftext     .tl_set:N     = \l_stex_key_Ftext_tl ,
}{id}
```

\mcc

```
\cs_set:Nn \__problems_maybe_newline: { \\ }
```

```
\stex_if_html_backend:TF{
  \tl_set:Nn \problem_mcc_box_tl {
    \ltx@ifpackageloaded{amssymb}{$\square$}{
      \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
    }
  }

  \newcommand\mcc[2][]{
    \stex_keys_set:nn{mcc}{#1}
    \__problems_mccline:n{
      \stex_if_do_html:T{
        \stex_annotate:nn{data-ftml-problem-choice={
          \bool_if:NTF \l_stex_key_T_bool {true}{false}
        }}{
          #2~
          \stex_annotate:nn{data-ftml-problem-choice-verdict={}}{
            \bool_if:NTF \l_stex_key_T_bool {
              \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_t
            }{
              \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
            }
          }
          \tl_if_empty:NF \l_stex_key_feedback_tl {
            \stex_annotate:nn{data-ftml-problem-choice-feedback={}}{
              \l_stex_key_feedback_tl
            }
          }
        }
      }
    }
  }
}{
  \tl_set:Nn \problem_mcc_box_default_tl {
    \ltx@ifpackageloaded{amssymb}{$\square$}{
      \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
    }
  }
  \tl_set:Nn \problem_mcc_box_tl {
    \ifsolutions
      \bool_if:NTF \l_stex_key_T_bool {
        \makebox[0pt][l]{\problem_mcc_box_default_tl}
          \hspace{0.1em}$\cs_if_exist:NTF\checkmark{
            \raisebox{.15ex}{\checkmark}
          } X$
      }{\problem_mcc_box_default_tl}
    \else
      \problem_mcc_box_default_tl
    \fi
  }
  \newcommand\mcc[2][]{
    \stex_keys_set:nn{mcc}{#1}
    \ifsolutions
      \tl_set:Nx \l_tmpa_tl{
        \bool_if:NTF \l_stex_key_T_bool {
```

```
          \tl_if_empty:NF \l_stex_key_Ttext_tl {\exp_not:N \l_stex_key_Ttext_tl}
        }{
          \tl_if_empty:NF \l_stex_key_Ftext_tl {\exp_not:N \l_stex_key_Ftext_tl}
        }
        \tl_if_empty:NF \l_stex_key_feedback_tl {
          \exp_not:n{\\\emph{\l_stex_key_feedback_tl}}
        }
      }
    \fi

    \__problems_mccline:n{ #2
      \ifsolutions
        \tl_if_empty:NF \l_tmpa_tl { \footnote{\l_tmpa_tl} }
      \fi
    }
  }
}
\stex_deactivate_macro:Nn \mcc {mcb~environments}
```

(*End of definition for* \mcc. *This function is documented on page 115.*)

scb (*env.*)

```

\cs_new_protected:Nn \__problems_maybe_inline:n {
  \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
    \let\__problems_oldpar:\stex_par:
    \cs_set:Nn\stex_par:{~}
    \cs_set:Nn\__problems_maybe_newline:{}
    #1
    \let\stex_par:\__problems_oldpar:
  }{
    \clist_if_in:NnTF \l_stex_key_style_clist {dropdown} {
      \let\__problems_oldpar:\stex_par:
      \cs_set:Nn\stex_par:{~}
      \cs_set:Nn\__problems_maybe_newline:{}
      #1
      \let\stex_par:\__problems_oldpar:
    }{\par #1}
  }
}

\stex_new_stylable_env:nnnnnnn{scb}{O{}}{
  \stex_keys_set:nn{style}{#1}
  \cs_set:Nn \__problems_sccline:n{
    \begin{list}{}{
      \setlength\topsep{0pt}
      \setlength\parsep{0pt}
      \setlength\rightmargin{0pt}
    }\item[\problem_scc_box_tl] ##1 \end{list}
  }

  \stex_if_do_html:T{
    \__problems_maybe_inline:n{
      \exp_args:Ne\stex_annotate_env{
```

```
        data-ftml-single-choice-block={}
        \clist_if_empty:NF \l_stex_key_style_clist {,
          data-ftml-styles={\l_stex_key_style_clist}
        }
      }
    }
    \tl_set:Nn\problem_scc_box_tl{}
    \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
      \cs_set:Nn \__problems_sccline:n {##1}
    }{
      \clist_if_in:NnT \l_stex_key_style_clist {dropdown} {
        \cs_set:Nn \__problems_sccline:n {##1}
      }
    }
  }
  \stex_deactivate_macro:Nn \mcb {sproblem~environments}
  \stex_deactivate_macro:Nn \scb {sproblem~environments}
  \stex_deactivate_macro:Nn \solution {sproblem~environments}
  \stex_deactivate_macro:Nn \hint {sproblem~environments}
  \stex_deactivate_macro:Nn \exnote {sproblem~environments}
  \stex_deactivate_macro:Nn \gnote {sproblem~environments}
  \stex_reactivate_macro:N \scc
  \stex_if_do_html:F \stex_style_apply:
}{
  \stex_if_do_html:TF{
    \__problems_maybe_inline:n { \endstex_annotate_env }
  }\stex_style_apply:
}{\par}{}{}

\stexstylescb[inline]{
  {\Large[}\cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
}{{\Large]}}

\stexstylescb[dropdown]{
  {\Large[}\cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
}{{\Large]}}

\stex_deactivate_macro:Nn \scb {sproblem~environments}
```

\scc

```

\stex_if_html_backend:TF{
  \tl_set:Nn\problem_scc_box_tl{$\bigcirc$}
  \newcommand\scc[2][]{
    \stex_keys_set:nn{mcc}{#1}
    \__problems_sccline:n{
      \stex_if_do_html:T{
        \stex_annotate:nn{data-ftml-problem-choice={
          \bool_if:NTF \l_stex_key_T_bool {true}{false}
        }}{
          #2~
          \stex_annotate:nn{data-ftml-problem-choice-verdict={}}{
            \bool_if:NTF \l_stex_key_T_bool {
              \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_t
```

```
            }{
              \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
            }
          }
          \tl_if_empty:NF \l_stex_key_feedback_tl {
            \stex_annotate:nn{data-ftml-problem-choice-feedback={}}{
              \l_stex_key_feedback_tl
            }
          }
        }
      }
    }
  }
}{
  \tl_set:Nn\problem_scc_box_default_tl{$\bigcirc$}
  \tl_set:Nn\problem_scc_box_tl{
    \ifsolutions
      \bool_if:NTF \l_stex_key_T_bool {
        \makebox[0pt][l]{\problem_scc_box_default_tl}
          \hspace{0.1em}$\cs_if_exist:NTF\checkmark{
            \raisebox{.15ex}{\checkmark}
          } X$
      }{\problem_scc_box_default_tl}
    \else
      \problem_scc_box_default_tl
    \fi
  }
  \newcommand\scc[2][]{
    \stex_keys_set:nn{mcc}{#1}
    \ifsolutions
      \tl_set:Nx \l_tmpa_tl{
        \bool_if:NTF \l_stex_key_T_bool {
          \tl_if_empty:NF \l_stex_key_Ttext_tl {\exp_not:N \l_stex_key_Ttext_tl}
        }{
          \tl_if_empty:NF \l_stex_key_Ftext_tl {\exp_not:N \l_stex_key_Ftext_tl}
        }
        \tl_if_empty:NF \l_stex_key_feedback_tl {
          \exp_not:n{ \\\emph{\l_stex_key_feedback_tl} }
        }
      }
    \fi

    \__problems_sccline:n{ #2
      \ifsolutions
        \tl_if_empty:NF \l_tmpa_tl { \footnote{\l_tmpa_tl} }
      \fi
    }
  }
}

\stex_deactivate_macro:Nn \scc {scb~environments}


\newcommand\yesTnoF{
```

```
  \begin{scb}[style=inline]
    \scc[T]{yes}~\scc[F]{no}
  \end{scb}
}
\newcommand\yesFnoT{
  \begin{scb}[style=inline]
    \scc[F]{yes}~\scc[T]{no}
  \end{scb}
}
\newcommand\trueTfalseF{
  \begin{scb}[style=inline]
    \scc[T]{true}~\scc[F]{false}
  \end{scb}
}
\newcommand\trueFfalseT{
  \begin{scb}[style=inline]
    \scc[F]{true}~\scc[T]{false}
  \end{scb}
}
```

(*End of definition for* \scc. *This function is documented on page* ??.)

\fillinsol

```
\stex_keys_define:nnnn{fillinsol}{
  \tl_clear:N \l__problems_fillin_solution_tl
  \dim_zero:N \l_stex_key_testspace_dim
}{
  testspace   .dim_set:N = \l_stex_key_testspace_dim,
  exact    .code:n = {\__problems_parse_fillin_arg:nnnn{exact}#1 },
  numrange .code:n = {\__problems_parse_fillin_arg:nnnn{numrange}#1 },
  regex    .code:n = {\__problems_parse_fillin_arg:nnnn{regex}#1 }
}{}

\tl_const:Nn \c__problems_true_tl{T}
\tl_const:Nn \c__problems_false_tl{F}

\msg_set:nnn{problem}{error/neither-true-nor-false}{
  Fillinsol~verdict~#1~is~neither~T~nor~F!
}

\stex_if_html_backend:TF{
  \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
    \tl_set:Nn \l_tmpa_tl{#3}
    \tl_if_eq:NNF \l_tmpa_tl\c__problems_true_tl{
      \tl_if_eq:NNF \l_tmpa_tl\c__problems_false_tl {
        \msg_error:nnn{problem}{error/neither-true-nor-false}{#3}
      }
    }
    \tl_put_right:Ne \l__problems_fillin_solution_tl {
      \stex_annotate:nn{
        data-ftml-fillin-case={#1},
        data-ftml-fillin-case-value={\tl_to_str:n{#2}},
        data-ftml-fillin-case-verdict={
          \tl_if_eq:NNTF\l_tmpa_tl\c__problems_true_tl{true}{false}
```

```
        },
      }{\_stex_annotate_force_break:n{\exp_not:n{#4}}}
    }
  }
}{
  \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
    \tl_put_right:Nn \l__problems_fillin_solution_tl {
      #1 & \tl_to_str:n{#2} & #3 & #4 \crcr
    }
  }
}


\newcommand\fillinsol[2][]{
  \stex_if_do_html:F \quad
  \mode_if_math:TF{
    \hbox{\__problems_fillinsol:nn{#1}{#2}}
  }{
    \__problems_fillinsol:nn{#1}{#2}
  }
  \stex_if_do_html:F \quad
}

\stex_if_html_backend:TF{
  \cs_new_protected:Nn \__problems_fillinsol:nn {
    \stex_keys_set:nn{fillinsol}{#1}
    \tl_if_empty:nF{#2}{
      \__problems_parse_fillin_arg:nnnn{exact}{#2}{T}{}
    }
    \hbox_set:Nn \l_tmpa_box{\fbox{\huge{\texttt{\tl_to_str:n{#2}}}\hskip\l_stex_key_testsp
    \exp_args:Ne \stex_annotate:nn{
      data-ftml-fillinsol={},
      data-ftml-fillinsol-width={\dim_to_decimal:n{1.5 \box_wd:N \l_tmpa_box }}
    }{
      \_stex_annotate_force_break:n{
        \l__problems_fillin_solution_tl
        #2
      }
    }
  }
}{
  \cs_new_protected:Nn \__problems_fillinsol:nn {
    \stex_keys_set:nn{fillinsol}{#1}
    \ifsolutions

      \cs_if_exist:NT\textcolor{\textcolor{red}}{\fbox{\texttt{\tl_to_str:n{#2}}}}
      \tl_if_empty:NF \l__problems_fillin_solution_tl {
        \footnote{
          \halign{ ~##~\hfil & ~##~\hfil & ~##~\hfil & ~##~\hfil \cr
            \textbf{type}&\textbf{case}&\textbf{verdict}&\textbf{feedback}\cr
            \tl_if_empty:nF{#2}{
              exact & #2 & T & \cr
            }
            \l__problems_fillin_solution_tl
```

```
            }
          }
        }
      \else
        \fbox{\dim_compare:nNnTF\l_stex_key_testspace_dim={0pt}{
          \phantom{\huge{\tl_to_str:n{#2}}}
        }{
          \hspace{\l_stex_key_testspace_dim}\vphantom{\huge{A \tl_to_str:n{#2}}}
        }}
      \fi
    }
}

\stex_deactivate_macro:Nn \fillinsol {sproblem~environments}
```

(*End of definition for* `\fillinsol`. *This function is documented on page 117.*)

`\testemptypage`

```
\newcommand\testemptypage[1][]{%
 \bool_if:NT \c__problems_test_bool {\ \vfill\begin{center}\hwexam@kw@testemptypage\end{cen
}
```

(*End of definition for* `\testemptypage`. *This function is documented on page* **??**.)

`\testspace`

```
\newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}
\newcommand\testsmallspace{\testspace{1cm}}
\newcommand\testmedspace{\testspace{2cm}}
\newcommand\testbigspace{\testspace{3cm}}
```

(*End of definition for* `\testspace`. *This function is documented on page* **??**.)

`\testnewpage`

```
\newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}
```

(*End of definition for* `\testnewpage`. *This function is documented on page* **??**.)

`\testnewpage`

```
\newcommand{\testnewpageInProblem}%
{\ifintest\vfill\begin{center}\emph{continued~on~next~page}\end{center}\testnewpage\fi}%
```

(*End of definition for* `\testnewpage`. *This function is documented on page* **??**.)

```
⟨/package⟩
```

## 39.3 Implementation: The hwexam Package

### 39.3.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
⟨*package⟩
\ProvidesExplPackage{hwexam}{2026/06/24}{4.1.0}{homework assignments and exams}
```

```
\RequirePackage{l3keys2e}

\keys_define:nn {hwexam / pkg}{
 multiple  .default:n   = { false },
 multiple .bool_set:N  = \c_hwexam_multiple_bool,
 qrcode  .default:n    = { false },
 qrcode  .bool_set:N  = \c_hwexam_qrcode_bool,
 unknown   .code:n    = {
  \PassOptionsToPackage{\CurrentOption}{problem}
 }
}
\ProcessKeysOptions{ hwexam /pkg }
\RequirePackage{problem}
```

`\hwexam_kw_*` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
\AddToHook{begindocument}{
 \ExplSyntaxOn\makeatletter
 \input{hwexam-english.ldf}
 \ltx@ifpackageloaded{babel}{
   \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
    \input{hwexam-ngerman.ldf}
   }
   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
    \input{hwexam-finnish.ldf}
   }
   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
    \input{hwexam-french.ldf}
   }
   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
    \input{hwexam-russian.ldf}
   }
 }{}
 \makeatother\ExplSyntaxOff
}
```

(*End of definition for* `\hwexam_kw_*`. *This function is documented on page* **??**.)

### 39.3.2 QR Codes

```
\group_begin:
  \escapechar=-1
  \xdef\_@@_qr_backslash{\string\\}
\group_end:

\bool_if:NT \c_hwexam_qrcode_bool {
 \RequirePackage{qrcode}
 \RequirePackage{marginnote}
 \str_new:N \g_@@_qr_json_str
 \bool_new:N \g_@@_qr_in_problems_json_bool
 \bool_set_false:N \g_@@_qr_in_problems_json_bool
 \bool_new:N \g_@@_qr_in_subproblems_json_bool
 \bool_set_false:N \g_@@_qr_in_subproblems_json_bool
```

```
\bool_new:N \g_@@_qr_in_anscls_json_bool
\bool_set_false:N \g_@@_qr_in_anscls_json_bool
\bool_if:NTF \c__problems_gnotes_bool {
  \gdef \qrjson { \str_gput_right:Nx \g_@@_qr_json_str}
}{
 \gdef \qrjson #1 {}
}
\qrjson{[}
\AtEndDocument{\qrjson{]}}
\def\_@@_qr_escape_char:n #1 {
 #1\exp_args:Nno\use:nn{\if_charcode:w#1}\_@@_qr_backslash#1\fi
}

\gdef\_@@_qr_escape:n #1{\exp_args:Ne\str_map_function:nN{\tl_to_str:n{#1}}\_@@_qr_escape_
\gdef \_@@_qr_escape:o #1{\exp_args:No\_@@_qr_escape:n#1}

\def\qrschema{TO:DO:\examnumber}

\bool_if:NTF \c__problems_test_bool {
 \def\doproblemqr{
  \ifstexhtml\else{
   \reversemarginpar\marginnote{\qrcode[height=1.5cm]{\qrschema}}
   \normalmarginpar
  }\fi
 }
 \def\insertexamnumber{
  \ifstexhtml\else
   \tl_if_exist:NTF \examnumber {
    {\Large\bfseries ID:~\examnumber}
   }{
    {\color{red}\Large\bfseries!!!~ WARNING:~NO~{\string\examnumber}~SET~!!!}\gdef\examnum
   }
   \global\def\insertexamnumber{}
  \fi
 }
}{
 \def\doproblemqr{}
 \def\insertexamnumber{}
}

\stexstyleproblem[noqr]{
 \par\noindent\problemheader \xdef\qrid{\thesproblem}
 \bool_if:NT \c__problems_pts_bool {
  \tl_if_eq:NnF \l__problems_pts_tl {0}{
   \marginpar{\l__problems_pts_tl{}~\problem@kw@points\smallskip}
  }
 }
 \bool_if:NT \c__problems_min_bool {
  \tl_if_eq:NnF \l__problems_min_tl {0} {
   \marginpar{\l__problems_min_tl{}~\problem@kw@minutes\smallskip}
  }
 }

 \bool_if:NTF \g_@@_qr_in_problems_json_bool {
```

```
    \qrjson{,}
   }{
    \bool_gset_true:N \g_@@_qr_in_problems_json_bool
   }

   \qrjson {
    \c_left_brace_str
     "id":"\_@@_qr_escape:o\l_stex_key_id_str","title":"\_@@_qr_escape:o\l_stex_key_title_tl
     "number":"\thesproblem"
   }

   \tl_if_eq:NnF \l__problems_pts_tl {0}{
    \qrjson {
     ,"pts":\l__problems_pts_tl
    }
   }
   \par
   \stex_ignore_spaces_and_pars:
  }{
   \bool_if:NT \g_@@_qr_in_subproblems_json_bool {
    \qrjson {]}
   }
   \bool_gset_false:N \g_@@_qr_in_subproblems_json_bool
   \qrjson {\c_right_brace_str}
   \par\bigskip
  }

  \stexstyleproblem{
   \par\noindent\problemheader \xdef\qrid{\thesproblem}
   \doproblemqr
   \bool_if:NT \c__problems_pts_bool {
    \tl_if_eq:NnF \l__problems_pts_tl {0}{
     \marginpar{\l__problems_pts_tl{}~\problem@kw@points\smallskip}
    }
   }
   \bool_if:NT \c__problems_min_bool {
    \tl_if_eq:NnF \l__problems_min_tl {0} {
     \marginpar{\l__problems_min_tl{}~\problem@kw@minutes\smallskip}
    }
   }

   \bool_if:NTF \g_@@_qr_in_problems_json_bool {
    \qrjson{,}
   }{
    \bool_gset_true:N \g_@@_qr_in_problems_json_bool
   }

   \qrjson {
    \c_left_brace_str
     "id":"\_@@_qr_escape:o\l_stex_key_id_str","title":"\_@@_qr_escape:o\l_stex_key_title_tl
     "number":"\thesproblem"
   }

   \tl_if_eq:NnF \l__problems_pts_tl {0}{
```

```
   \qrjson {
    ,"pts":\l__problems_pts_tl
   }
  }
  \par
  \stex_ignore_spaces_and_pars:
 }{
  \bool_if:NT \g_@@_qr_in_subproblems_json_bool {
   \qrjson {]}
  }
  \bool_gset_false:N \g_@@_qr_in_subproblems_json_bool
  \qrjson {\c_right_brace_str}
  \par\bigskip
 }

 \stexstylesubproblem[noqr]{
  \begin{list}{}{
   \setlength\topsep{0pt}
   \setlength\parsep{0pt}
   \setlength\rightmargin{0pt}
  }\item[\int_use:N \g__problems_subproblem_int .]
  \xdef\qrid{\thesproblem.\int_use:N \g__problems_subproblem_int}
  \bool_if:NT \c__problems_pts_bool {
   \bool_if:NF \l__problems_has_pts_bool {
    \marginpar{\smallskip\l_stex_key_pts_tl{}~\problem@kw@points}
   }
  }
  \bool_if:NT \c__problems_min_bool {
   \bool_if:NF \l__problems_has_min_bool{
    \marginpar{\smallskip\l_stex_key_min_tl{}~\problem@kw@minutes}
   }
  }
  \bool_if:NTF \g_@@_qr_in_subproblems_json_bool {
   \qrjson {,}
  }{
   \qrjson {
    ,"subproblems":[
   }
   \bool_gset_true:N \g_@@_qr_in_subproblems_json_bool
  }
  \qrjson {
   \c_left_brace_str
    "id":"\_@@_qr_escape:o\l_stex_key_id_str","title":"\_@@_qr_escape:o\l_stex_key_title_tl
    "number":"\thesproblem.\int_use:N \g__problems_subproblem_int"
  }
  \bool_if:NF \l__problems_has_pts_bool {
   \qrjson {
    ,"pts":\l_stex_key_pts_tl
   }
  }
 }{
  \qrjson {\c_right_brace_str}
  \end{list}
 }
```

```
\stexstylesubproblem{
 \begin{list}{}{
  \setlength\topsep{0pt}
  \setlength\parsep{0pt}
  \setlength\rightmargin{0pt}
 }\item[\int_use:N \g__problems_subproblem_int .]
 \xdef\qrid{\thesproblem.\int_use:N \g__problems_subproblem_int}
 \doproblemqr
 \bool_if:NT \c__problems_pts_bool {
  \bool_if:NF \l__problems_has_pts_bool {
   \marginpar{\smallskip\l_stex_key_pts_tl{}~\problem@kw@points}
  }
 }
 \bool_if:NT \c__problems_min_bool {
  \bool_if:NF \l__problems_has_min_bool{
   \marginpar{\smallskip\l_stex_key_min_tl{}~\problem@kw@minutes}
  }
 }
 \bool_if:NTF \g_@@_qr_in_subproblems_json_bool {
  \qrjson {,}
 }{
  \qrjson {
   ,"subproblems":[
  }
  \bool_gset_true:N \g_@@_qr_in_subproblems_json_bool
 }
 \qrjson {
  \c_left_brace_str
   "id":"\_@@_qr_escape:o\l_stex_key_id_str","title":"\_@@_qr_escape:o\l_stex_key_title_tl
   "number":"\thesproblem.\int_use:N \g__problems_subproblem_int"
 }
 \bool_if:NF \l__problems_has_pts_bool {
  \qrjson {
   ,"pts":\l_stex_key_pts_tl
  }
 }
}{
 \qrjson {\c_right_brace_str}
 \end{list}
}

\stexstylegnote{
 \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
 \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
  {~}\l_stex_key_title_tl
 } :~}
 \qrjson {
  ,"gnote": \c_left_brace_str
  "id":"\_@@_qr_escape:o\l_stex_key_id_str","title":"\_@@_qr_escape:o\l_stex_key_title_tl"
  "anscls":[
 }
}{
 \qrjson {
```

```
  ]\c_right_brace_str
 }
 \par\rule[.3em]{\linewidth}{0.4pt}\newline
}

\renewcommand \anscls [2][] {
 \stex_keys_set:nn{ anscls }{#1}
 \str_if_empty:NT \l_stex_key_id_str {
  \int_incr:N \l__problems_anscls_int
  \str_set:Nx \l_stex_key_id_str {
   AC\int_use:N \l__problems_anscls_int
  }
 }
 \bool_if:NTF \g_@@_qr_in_anscls_json_bool {
  \qrjson{,}
 }{
  \bool_set_true:N \g_@@_qr_in_anscls_json_bool
 }
 \qrjson{
  \c_left_brace_str
   "id":"\_@@_qr_escape:o\l_stex_key_id_str","description":"\_@@_qr_escape:n{#2}",
   "feedback":"\_@@_qr_escape:o\l_stex_key_feedback_tl",
   "pts":"\l_stex_key_pts_str"
  \c_right_brace_str
 }
 \begin{list}{}{
  \setlength\topsep{0pt}
  \setlength\parsep{0pt}
  \setlength\rightmargin{0pt}
 }\item[\l_stex_key_id_str]
  \stex_if_do_html:TF{
   \exp_args:Ne \stex_annotate:nn{
    data-ftml-answerclass={\l_stex_key_id_str}
    \str_if_empty:NF \l_stex_key_pts_str{
     ,data-ftml-answerclass-pts={\l_stex_key_pts_str}
    }
   }{
    #2
    \tl_if_empty:NF \l_stex_key_feedback_tl{
     \stex_annotate_invisible:nn{
      data-ftml-answerclass-feedback={true}
     }{\l_stex_key_feedback_tl}
    }
   }
  }{#2}
  \str_if_empty:NF \l_stex_key_pts_str {\par
   ~ \problem@kw@points :~\l_stex_key_pts_str
  }
  \str_if_empty:NF \l_stex_key_feedback_tl {\par
   ~ \problem@kw@feedback :~\l_stex_key_feedback_tl
  }
 \end{list}
}
\stex_deactivate_macro:Nn \anscls {gnote~environments}
```

```
\AtEndDocument{
 \message{^^J^^J\g_@@_qr_json_str^^J^^J}
 \bool_if:NT \c__problems_gnotes_bool {
  \iow_new:N \c_@@_qr_json_iow
  \iow_open:Nn \c_@@_qr_json_iow {\jobname-vollkorn.json}
  \iow_now:Nx \c_@@_qr_json_iow {\g_@@_qr_json_str}
  \iow_close:N \c_@@_qr_json_iow
 }
}

\renewcommand\testemptypage[1][]{%
 \bool_if:NT \c__problems_test_bool {\
 \xdef\qrid{P\thepage}
 \doproblemqr
 \vfill\begin{center}\hwexam@kw@testemptypage\end{center}\eject
 }
}
}
```

### 39.3.3 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

`assignment` (*env.*)

```
\stex_keys_define:nnnn{ assignment }{
  \tl_clear:N \l_stex_key_number_tl
 \tl_clear:N \l_stex_key_given_tl
 \tl_clear:N \l_stex_key_due_tl
}{
  number  .tl_set:N     = \l_stex_key_number_tl,
  given   .tl_set:N     = \l_stex_key_given_tl,
  due     .tl_set:N     = \l_stex_key_due_tl,
 unknown .code:n = {}
}{id,title,style}

\newcounter{assignment}
\stex_new_stylable_env:nnnnnnn {assignment}{O{}}{
  \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
    \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {#1,}
    \exp_args:Nno \stex_keys_set:nn{assignment}{
      \l_hwexam_includeassignment_keys_tl
    }
  }{
    \stex_keys_set:nn{assignment}{#1}
  }
 \tl_if_empty:NF \l_stex_key_number_tl {
  \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
 }
 \global\refstepcounter{assignment}
 \setcounter{sproblem}{0}
 \def\thesproblem{\theassignment.\arabic{sproblem}}
```

```
 \stex_if_do_html:T{
  \tl_if_empty:NF \l_stex_key_title_tl {
   \stexdoctitle \l_stex_key_title_tl
  }
 }
 \stex_style_apply:
 \_stex_do_id:
}{
 \stex_style_apply:
}{
 \par\begin{center}
 \textbf{\Large\assignmentautorefname~\theassignment
  \tl_if_empty:NF \l_stex_key_title_tl {
   {~}---~\l_stex_key_title_tl
  }
 }\par\smallskip
 \textbf{
  \tl_if_empty:NF \l_stex_key_given_tl {
   \hwexam@kw@given :~\l_stex_key_given_tl\quad
  }
  \tl_if_empty:NF \l_stex_key_due_tl {
   \hwexam@kw@due :~\l_stex_key_due_tl\quad
  }
 }
 \end{center}
 \par\bigskip
}{
 \par\pagebreak
}{}
```

\includeassignment

```
\NewDocumentCommand\includeassignment{O{} m}{
 \group_begin:
 \tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
 \stex_keys_set:nn{includeproblem}{#1}
 \exp_args:Nno \use:nn{\inputref[}\l_stex_key_mhrepos_str]{#2}
 \group_end:
}
```

(*End of definition for* \includeassignment*. This function is documented on page 76.*)

Restoring information about problems:

```
\prop_new:N \c_@@_problems_prop
\tl_set:Nn \c_@@_total_mins_tl {0pt}
\tl_set:Nn \c_@@_total_pts_tl {0pt}
\int_new:N \c_@@_total_problems_int
\cs_set_protected:Npn \problem@restore #1 #2 #3 {
 \int_gincr:N \c_@@_total_problems_int
 \prop_gput:Nnn \c_@@_problems_prop {#1}{{#2}{#3}}
 \tl_gset:Ne \c_@@_total_pts_tl { \dim_eval:n { \c_@@_total_pts_tl + #2pt }}
 \tl_gset:Ne \c_@@_total_mins_tl { \dim_eval:n { \c_@@_total_mins_tl + #2pt }}
}
```

\correction@table This macro generates the correction table

```
\newcommand\correction@table{
```

```
 \int_compare:nNnT \c_@@_total_problems_int = 0 {
  \int_incr:N \c_@@_total_problems_int
  \prop_put:Nnn \c_@@_problems_prop {~}{{~}{~}}
 }
 \tl_clear:N \l_tmpa_tl
 \tl_clear:N \l_tmpb_tl
 \tl_clear:N \l_tmpc_tl
 \prop_map_inline:Nn \c_@@_problems_prop {
  \tl_put_right:Nn \l_tmpa_tl { ##1 & }
  \tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
  \tl_put_right:Nn \l_tmpc_tl { & }
 }
 \resizebox{\textwidth}{!}{%
\exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c|}c||l|}\hline
&\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c||}
{\footnotesize\hwexam@kw@forgrading} &\\\hline
\hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\\\hline
\hwexam@kw@pts & \l_tmpb_tl \dim_to_decimal:n\c_@@_total_pts_tl & \\\hline
\hwexam@kw@reached & \l_tmpc_tl & \\[.7cm]\hline
\end{tabular}}}
```

(*End of definition for* `\correction@table`. *This function is documented on page* **??**.)

`\testheading`

```
\def\hwexamheader{\input{hwexam-default.header}}

\def\hwexamminutes{
 \tl_if_empty:NTF \hwexam@duration {
  {\hwexam@min}~\hwexam@minutes@kw
 }{
  \hwexam@duration
 }
}

\stex_keys_define:nnnn{ hwexam / testheading }{
 \tl_clear:N \hwexam@min
 \tl_clear:N \hwexam@duration
 \tl_clear:N \hwexam@reqpts
 \tl_clear:N \hwexam@tools
}{
 min   .tl_set:N  = \hwexam@min,
 duration .tl_set:N  = \hwexam@duration,
 reqpts  .tl_set:N  = \hwexam@reqpts,
 tools  .tl_set:N  = \hwexam@tools
}{}

\newenvironment{testheading}[1][]{
 \stex_keys_set:nn { hwexam / testheading}{#1}

 \tl_set:Nx \hwexam@totalpts {\dim_to_decimal:n \c_@@_total_pts_tl}
 \tl_set:Nx \hwexam@totalmin {\dim_to_decimal:n \c_@@_total_mins_tl}
 \tl_set:Nx \hwexam@checktime {\dim_to_decimal:n { \hwexam@min pt - \hwexam@totalmin pt }}

 \newif\if@bonuspoints
```

```
 \tl_if_empty:NTF \hwexam@reqpts {
  \@bonuspointsfalse
 }{
  \tl_set:Nx \hwexam@bonuspts {
   \dim_to_decimal:n{\hwexam@totalpts pt - \hwexam@reqpts pt}
  }
  \@bonuspointstrue
 }

 \makeatletter\hwexamheader\makeatother
}{
 \newpage
}
```

(*End of definition for* `\testheading`. *This function is documented on page* **??**.)

examdata
quizdata
homeworkdata

```
\bool_new:N \g_@@_allow_data_bool
\bool_gset_true:N \g_@@_allow_data_bool
\AtBeginDocument{
 \bool_gset_false:N \g_@@_allow_data_bool
}

\msg_set:nnn{stex}{hwexam/doubledata}{
  Exam/Homework/Quiz~Data~already~set
}

\msg_set:nnn{stex}{hwexam/nodate}{
  Exam/Homework/Quiz~Data~missing~date~value
}

\msg_set:nnn{stex}{hwexam/nocourse}{
  Exam/Homework/Quiz~Data~missing~course~value
}

\stex_keys_define:nnnn{ hwexam / commondata }{
 % https://docs.rs/dateparser/latest/dateparser/#accepted-date-formats
 \str_clear:N \l_@@_key_exam_date_str
 \str_clear:N \l_@@_key_exam_course_id_str
 \str_clear:N \l_@@_key_exam_term_str
 \int_set:Nn \l_@@_key_exam_num_int {1}
}{
 date     .str_set:N = \l_@@_key_exam_date_str,
 course   .str_set:N = \l_@@_key_exam_course_id_str,
 term     .str_set:N = \l_@@_key_exam_term_str,
 num      .int_set:N = \l_@@_key_exam_num_int
}{}

\stex_keys_define:nnnn{ hwexam / examdata }{
 \bool_set_false:N \l_@@_key_exam_retake_bool
}{
 retake   .bool_set:N = \l_@@_key_exam_retake_bool
}{hwexam/commondata}
```

```
\cs_set_protected:Nn \_@@_do_metadata:nnnn {
 \bool_if:NF \g_@@_allow_data_bool {
  \msg_fatal:nn{stex}{hwexam/doubledata}
 }
 \bool_gset_false:N \g_@@_allow_data_bool
 \stex_keys_set:nn { hwexam / #1}{#2}
 \str_set:Nn \g_stex_document_kind_str{#4}
 \str_if_empty:NT\l_@@_key_exam_date_str{
  \str_set:Ne \l_@@_key_exam_date_str \today
  %\msg_fatal:nn{stex}{hwexam/nodate}
 }
 \str_if_empty:NT\l_@@_key_exam_course_id_str{
  \msg_fatal:nn{stex}{hwexam/nocourse}
 }
 \cs_if_exist:NT \date {
  \exp_args:No \date \l_@@_key_exam_date_str
 }

 \tl_set:Ne \g_stex_document_kind_parameters_tl{
  ,data-ftml-document-kind-date={\l_@@_key_exam_date_str}
  ,data-ftml-document-kind-num=\int_use:N \l_@@_key_exam_num_int
  ,data-ftml-document-kind-course=\l_@@_key_exam_course_id_str
  \str_if_empty:NF \l_@@_key_exam_term_str{
   ,data-ftml-document-kind-term=\l_@@_key_exam_term_str
  }
  #3
 }
}

\newcommand\examdata[1]{
 \_@@_do_metadata:nnnn{examdata}{#1}{
  ,data-ftml-document-kind-retake=\bool_if:NTF \l_@@_key_exam_retake_bool{true}{false}
 }{exam}
}

\newcommand\homeworkdata[1]{
 \_@@_do_metadata:nnnn{commondata}{#1}{}{homework}
}

\newcommand\quizdata[1]{
 \_@@_do_metadata:nnnn{commondata}{#1}{}{quiz}
}

```

(*End of definition for* `examdata`, `quizdata`, *and* `homeworkdata`. *These functions are documented on page* ??.)

⟨/package⟩

### 39.3.4 Leftovers

at some point, we may want to reactivate the logos font, then we use

```
here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
```

```
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}
```

## 39.4 Tikzinput Implementation

```
⟨@@=tikzinput⟩
⟨*package⟩

%%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%

\ProvidesExplPackage{tikzinput}{2025/11/11}{4.0.0}{tikzinput package}
\RequirePackage{l3keys2e}

\keys_define:nn { tikzinput } {
  image   .bool_set:N   = \c_tikzinput_image_bool,
  image   .default:n    = false ,
  unknown   .code:n       = {}
}

\ProcessKeysOptions { tikzinput }

\bool_if:NTF \c_tikzinput_image_bool {
  \RequirePackage{graphicx}

  \providecommand\usetikzlibrary[]{}
  \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
}{
  \RequirePackage{tikz}
  \RequirePackage{standalone}

  \newcommand \tikzinput [2] [] {
    \setkeys{Gin}{#1}
    \ifx \Gin@ewidth \Gin@exclamation
      \ifx \Gin@eheight \Gin@exclamation
        \input { #2 }
      \else
        \resizebox{!}{ \Gin@eheight }{
          \input { #2 }
        }
      \fi
    \else
```

```
      \ifx \Gin@eheight \Gin@exclamation
        \resizebox{ \Gin@ewidth }{!}{
          \input { #2 }
        }
      \else
        \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
          \input { #2 }
        }
      \fi
    \fi
  }
}

\newcommand \ctikzinput [2] [] {
  \begin{center}
    \tikzinput [#1] {#2}
  \end{center}
}
```

⟨/package⟩

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

**B**



**C**

## D



## E

**F**

## G



## H



## I

## N

## Q



## R

**S**

stex internal commands:

## T

**U**



**V**

**W**



**X**



**Y**