

The \TeX 3 Package *

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

2021-12-20

Abstract

TODO

*Version 3.0 (last revised 2021-12-20)

Contents

I	Manual	1
1	Stuff	2
1.1	Modules	2
1.1.1	Semantic Macros and Notations	2
	Other Argument Types	4
	Precedences	6
1.1.2	Archives and Imports	6
	Namespaces	6
	Paths in Import-Statements	7
II	Documentation	8
2	sTeX-Basics	9
2.1	Macros and Environments	9
3	sTeX-MathHub	11
3.1	Macros and Environments	11
3.1.1	Files, Paths, URIs	11
3.1.2	MathHub Archives	12
4	sTeX-References	14
4.1	Macros and Environments	14
5	sTeX-Modules	15
5.1	Macros and Environments	15
5.1.1	The module-environment	17
6	sTeX-Module Inheritance	20
6.1	Macros and Environments	20
6.1.1	SMS Mode	20
6.1.2	Imports and Inheritance	21
7	sTeX-Symbols	24
7.1	Macros and Environments	24
8	sTeX-Terms	27
8.1	Macros and Environments	27
9	sTeX-Structural Features	30
9.1	Macros and Environments	30
	Structures	30
10	sTeX-Statements	31
10.1	Macros and Environments	31
11	sTeX-Metatheory	32
11.1	Symbols	32

III Extensions	33
12 Tikzinput	34
12.1 Macros and Environments	34
13 document-structure.sty: Semantic Markup for Open Mathematical Documents in L^AT_EX	35
13.1 Introduction	35
13.2 The User Interface	36
13.2.1 Package and Class Options	36
13.2.2 Document Structure	36
13.2.3 Ignoring Inputs	37
13.2.4 Structure Sharing	38
13.2.5 Global Variables	38
13.2.6 Colors	39
13.3 Limitations	39
14 Slides and Course Notes	40
14.1 Introduction	40
14.2 The User Interface	40
14.2.1 Package Options	40
14.2.2 Notes and Slides	41
14.2.3 Header and Footer Lines of the Slides	42
14.2.4 Frame Images	42
14.2.5 Colors and Highlighting	43
14.2.6 Front Matter, Titles, etc.	43
14.2.7 Excursions	43
14.2.8 Miscellaneous	43
14.3 Limitations	43
15 problem.sty: An Infrastructure for formatting Problems	44
15.1 Introduction	44
15.2 The User Interface	44
15.2.1 Package Options	44
15.2.2 Problems and Solutions	45
15.2.3 Multiple Choice Blocks	46
15.2.4 Including Problems	46
15.2.5 Reporting Metadata	46
15.3 Limitations	46
16 hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	48
16.1 Introduction	49
16.2 The User Interface	49
16.2.1 Package and Class Options	49
16.2.2 Assignments	49
16.2.3 Typesetting Exams	49
16.2.4 Including Assignments	50
16.3 Limitations	50

IV	Implementation	52
17	$\text{\texttt{STeX}}$-Basics Implementation	53
17.1	The $\text{\texttt{STeX}}$ Document Class	53
17.2	Preliminaries	53
17.3	Messages and logging	54
17.4	Persistence	55
17.5	HTML Annotations	55
17.6	Languages	58
17.7	Activating/Deactivating Macros	59
18	$\text{\texttt{STeX}}$-MathHub Implementation	60
18.1	Generic Path Handling	60
18.2	PWD and <code>kpsewhich</code>	62
18.3	File Hooks and Tracking	63
18.4	MathHub Repositories	64
19	$\text{\texttt{STeX}}$-References Implementation	70
19.1	Document URIs and URLs	70
19.2	Setting Reference Targets	72
19.3	Using References	73
20	$\text{\texttt{STeX}}$-Modules Implementation	74
20.1	The module environment	77
20.2	Invoking modules	82
21	$\text{\texttt{STeX}}$-Module Inheritance Implementation	84
21.1	SMS Mode	84
21.2	Inheritance	88
22	$\text{\texttt{STeX}}$-Symbols Implementation	93
22.1	Symbol Declarations	93
22.2	Notations	99
23	$\text{\texttt{STeX}}$-Terms Implementation	107
23.1	Symbol Invocations	107
23.2	Terms	109
23.3	Notation Components	116
24	$\text{\texttt{STeX}}$-Structural Features Implementation	118
24.1	The feature environment	118
24.2	Features	120
25	$\text{\texttt{STeX}}$-Statements Implementation	125
25.1	Definitions	126
25.2	Assertions	127
25.3	Examples	129
26	$\text{\texttt{STeX}}$-Others Implementation	130
27	$\text{\texttt{STeX}}$-Metatheory Implementation	131

28 Tikzinput Implementation	134
29 document-structure.sty Implementation	136
29.1 The OMDoc Class	136
29.2 Class Options	136
29.3 Beefing up the <code>document</code> environment	137
29.4 Implementation: OMDoc Package	137
29.5 Package Options	137
29.6 Document Structure	139
29.7 Front and Backmatter	142
29.8 Global Variables	144
30 MiKoSlides – Implementation	145
30.1 Class and Package Options	145
30.2 Notes and Slides	147
30.3 Header and Footer Lines	151
30.4 Frame Images	152
30.5 Colors and Highlighting	153
30.6 Sectioning	154
30.7 Excursions	156
31 The Implementation	158
31.1 Package Options	158
31.2 Problems and Solutions	159
31.3 Multiple Choice Blocks	164
31.4 Including Problems	165
31.5 Reporting Metadata	166
32 Implementation: The hwexam Class	168
32.1 Class Options	168
33 Implementation: The hwexam Package	170
33.1 Package Options	170
33.2 Assignments	171
33.3 Including Assignments	174
33.4 Typesetting Exams	175
33.5 Leftovers	177

Part I
Manual

Chapter 1

Stuff

1.1 Modules

`\sTeX` Both print this \TeX logo.
`\stex`

1.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

To declare a symbol (in a module), we use `\symdecl`, which takes as argument the name of the corresponding semantic macro, e.g. `\symdecl{foo}` introduces the macro `\foo`. Additionally, `\symdecl` takes several options, the most important one being its arity. `foo` as declared above yields a *constant* symbol. To introduce an *operator* which takes arguments, we have to specify which arguments it takes.

For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

Example 1

```
\symdecl[args=2]{mult}
\notation{mult}{#1 #2}
 $\mult{a}{b}$ 
```

ab

Since usually, a freshly introduced symbol also comes with a notation from the start, the `\symdef` command combines `\symdecl` and `\notation`. So instead of the above, we could have also written

```
\symdef[args=2]{mult}{#1 #2}
```

Adding more notations like `\notation[cdot]{mult}{#1 \comp{\cdot} #2}` or `\notation[times]{mult}{#1 \comp{\times} #2}` allows us to write $\mult[cdot]{a}{b}$ and $\mult[times]{a}{b}$:

Example 2

```
\notation[cdot]{mult}{#1 \comp{\cdot} #2}
\notation[times]{mult}{#1 \comp{\times} #2}
 $\mult[cdot]{a}{b}$  and  $\mult[times]{a}{b}$ 
```

$a \cdot b$ and $a \times b$

.

Not using an explicit option with a semantic macro yields the first declared notation, unless changed¹.

Outside of math mode, or by using the starred variant `\foo*`, allows to provide a custom notation, where notational (or textual) components can be given explicitly in square brackets.

Example 3

```
 $\mult*{a}[\comp{\ast}]{b}$  is the
\mult[\comp{product of}][ $\$a$ ][\comp{and}][ $\$b$ ]
```

$a * b$ is the product of a and b

.

In custom mode, prefixing an argument with a star will not print that argument, but still export it to OMDoc:

Example 4

```
\mult[\comp{Multiplying}]* $\mult{a}{b}$ [ again by  $\$b$  yields ...
```

Multiplying again by b yields...

The syntax `*[int]` allows switching the order of arguments. For example, given a 2-ary semantic macro `\forevery` with exemplary notation `\forall #1. #2`, we can write

Example 5

```
\symdecl[ args=2]{forevery}
\forevery* [2]{The proposition  $\$P$ [\comp{holds for every} ]*[1]{ $\$x$  in  $A$ }}
```

The proposition P holds for every $x \in A$

¹EdNOTE: TODO

When using `*[n]`, after reading the provided (n th) argument, the “argument counter” automatically continues where we left off, so the `*[1]` in the above example can be omitted.

For a macro with `arity > 0`, we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point `!` in either text or math mode. For that reason `\notation` (and thus `\symdef`) take an additional optional argument `op=`, which allows to assign a notation for the operator itself. e.g.

Example 6

```
\symdef[ args=2,op={+}]{add}{#1 \comp+ #2}
The operator  $\textcolor{teal}{\$}\textcolor{teal}{\text{add}}\textcolor{teal}{\$}$  adds two elements, as in  $\textcolor{teal}{\$}\textcolor{teal}{\text{add}}\textcolor{teal}{ab}\textcolor{teal}{\$}$ .
```

The operator $+$ adds two elements, as in $a+b$.

`*` is composable with `!` for custom notations, as in:

Example 7

```
\mult![\comp{Multiplication}] (denoted by  $\textcolor{teal}{\$}\textcolor{teal}{\text{mult}}\textcolor{teal}{*}\textcolor{teal}{![\comp{\textcolor{teal}{\text{cdot}}}\textcolor{teal}{\$}]}$  is defined by...
```

$\textcolor{teal}{\text{Multiplication}}$ (denoted by \cdot) is defined by...

The macro `\comp` as used everywhere above is responsible for highlighting, linking, and tooltips, and should be wrapped around the notation (or text) components that should be treated accordingly. While it is attractive to just wrap a whole notation, this would also wrap around e.g. the arguments themselves, so instead, the user is tasked with marking the notation components themselves.

The precise behaviour of `\comp` is governed by the macro `\@comp`, which takes two arguments: The tex code of the text (unexpanded) to highlight, and the URI of the current symbol. `\@comp` can be safely redefined to customize the behaviour.

The starred variant `\symdecl*{foo}` does not introduce a semantic macro, but still declares a corresponding symbol. `foo` (like any other symbol, for that matter) can then be accessed via `\STEXsymbol{foo}` or (if `foo` was declared in a module `Foo`) via `\STEXModule{Foo}?{foo}`.

both `\STEXsymbol` and `\STEXModule` take any arbitrary ending segment of a full URI to determine which symbol or module is meant. e.g. `\STEXsymbol{Foo?foo}` is also valid, as are e.g. `\STEXModule{path?Foo}?{foo}` or `\STEXsymbol{path?Foo?foo}`

There’s also a convient shortcut `\symref{?foo}{some text}` for `\STEXsymbol{?foo}![some text]`

Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: *i*-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters

representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two i-type arguments.

Besides i-type arguments, \TeX has two other types, which we will discuss now.

The first are *binding* (b-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above `\forevery`-macro: The first argument is not actually an argument that the `forevery` “function” is “applied” to; rather, the first argument is a new variable (e.g. x) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef[args=bi]{forevery}{\forall #1.\; #2}
```

b-type arguments are indistinguishable from i-type arguments within \TeX , but are treated very differently in OMDOC and by MMT. More interesting *within* \TeX are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

Example 8

```
\symdef[ args=a]{mult}{#1}{#1 \comp\cdot #2}
$\mult{a,b,c,{d^e},f}$
```

$$a \cdot b \cdot c \cdot d^e \cdot f$$

As the example above shows, notations get a little more complicated for associative arguments. For every a-type argument, the `\notation`-macro takes an additional argument that declares how individual entries in an a-type argument list are aggregated. The first notation argument then describes how the aggregated expression is combined into the full representation.

For a more interesting example, consider a flexary operator for ordered sequences in ordered set, that taking arguments $\{a, b, c\}$ and `\mathbb{R}` prints $a \leq b \leq c \in \mathbb{R}$. This operator takes two arguments (an a-type argument and an i-type argument), aggregates the individuals of the associative argument using `\leq`, and combines the result with `\in` and the second argument thusly:

Example 9

```
\symdef[ args=ai]{numseq}{#1 \comp\in #2}{#1 \comp\leq #2}
$numseq{a,b,c}{\mathbb{R}}$
```

$$a \leq b \leq c \in \mathbb{R}$$

Finally, B-type arguments combine the functionalities of a and b, i.e. they represent flexary binding operator arguments.

2 3

²EDNOTE: what about e.g. $\int \int \int f \, dx \, dy \, dz$?

³EDNOTE: “decompose” a-type arguments into fixed-arity operators?

Precedences

Every notation has an (upwards) *operator precedence* and for each argument a (downwards) *argument precedence* used for automated bracketing. For example, a notation for a binary operator `\foo` could be declared like this:

```
\notation[prec=200;500x600]{foo}{#1 \comp{+} #2}
```

assigning an operator precedence of 200, an argument precedence of 500 for the first argument, and an argument precedence of 600 for the second argument.

\TeX insert brackets thusly: Upon encountering a semantic macro (such as `\foo`), its operator precedence (e.g. 200) is compared to the current downwards precedence (initially `\neginfprec`). If the operator precedence is *larger* than the current downwards precedence, parentheses are inserted around the semantic macro.

Notations for symbols of arity 0 have a default precedence of `\infprec`, i.e. by default, parentheses are never inserted around constants. Notations for symbols with arity > 0 have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

Consequently, if some operator A should bind stronger than some operator B , then A as operator precedence should be smaller than B 's argument precedences.

For example:

Example 10

```
\notation[prec=100]{plus}{#1 \comp{+} #2}
\notation[prec=50]{times}{#1 \comp{\cdot} #2}
 $\plus{a}{\times{b}{c}}$  and  $\times{a}{\plus{b}{c}}$ 
```

$a+b \cdot c$ and $a \cdot (b+c)$

1.1.2 Archives and Imports

Namespaces

Ideally, \TeX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like MMT does things.

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 \TeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

- If `\begin{module}{Foo}` occurs in a file `/path/to/file/Foo[.<lang>].tex` which does not belong to an 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 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 suffix¹.

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`-folder is replaced by the archive's namespace URI.

Paths in Import-Statements

Conversely, here is how namespaces/URIs and file paths are computed in import statements, exemplary `\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 document 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 document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the archive's `source`-folder.
- 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`-folder, 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.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

Since this is less compatible with a modular development, using full URIs directly is discouraged.

¹which is internally attached to the module name instead, but a user need not worry about that.

Part II

Documentation

Chapter 2

sTeX-Basics

Both the sTeX package and class offer the following package options:

debug ($\langle log-prefix \rangle *$) Logs debugging information with the given prefixes to the terminal, or all if **all** is given.

showmods ($\langle boolean \rangle$) Shows explicit module information at the document margins.

lang ($\langle language \rangle *$) Languages to load with the **babel** package.

mathhub ($\langle directory \rangle$) MathHub folder to search for repositories.

sms ($\langle boolean \rangle$) use *persisted* mode (see ???).

image ($\langle boolean \rangle$) passed on to tikzinput.

2.1 Macros and Environments

<code>\sTeX</code>	Both print this sTeX logo.
<code>\stex</code>	

<code>\stex_debug:nn</code>	<code>\stex_debug:nn {$\langle log-prefix \rangle$} {$\langle message \rangle$}</code>
-----------------------------	--

Logs $\langle message \rangle$, if the package option **debug** contains $\langle log-prefix \rangle$.

<code>\stex_add_to_sms:n</code>	Adds the provided code to the <code>.sms</code> -file of the document.
---------------------------------	--

<code>\if@latexml</code>	L ^A T _E X2e and L ^A T _E X3 conditionals for L ^A T _E XML.
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</code>	

We have four macros for annotating generated HTML (via L^AT_EXML or S^CA^LL^AT_EX) with attributes:

<code>\stex_annotate:nnn</code>	<code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

Annotates the HTML generated by $\langle content \rangle$ with

`property="stex:⟨property⟩", resource="⟨resource⟩".`

`\stex_annotate_invisible:n` adds the attributes

`stex:visible="false", style="display:none".`

`\stex_annotate_invisible:nnn` combines the functionality of both.

<code>stex_annotate_env</code>	<code>\begin{stex_annotate_env}{⟨property⟩}{⟨resource⟩}</code> $\langle content \rangle$ <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code> .
--------------------------------	---

<code>\c_stex_languages_prop</code>
<code>\c_stex_language_abbrevs_prop</code>

Map language abbreviations to their full babel names and vice versa. e.g. `\c_stex_languages_prop{en}` yields `english`, and `\c_stex_language_abbrevs_prop{english}` yields `en`.

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}</code>
<code>\stex_reactivate_macro:N</code>	

Makes the macro $\langle cs \rangle$ throw an error, indicating that it is only allowed in the context of $\langle environments \rangle$.

`\stex_reactivate_macro:N⟨cs⟩` reactivates it again, i.e. this happens ideally in the $\langle begin \rangle$ -code of the associated environments.

<code>\MSC</code>	<code>\MSC{⟨msc⟩}</code>
-------------------	--------------------------

Designates the *math subject classifier* of the current module / file.

Chapter 3

STEX-MathHub

Code related to managing and using MathHub repositories, files, paths and related hooks and methods.

3.1 Macros and Environments

<code>\stex_kpsewhich:n</code>	<code>\stex_kpsewhich:n</code> executes <code>kpsewhich</code> and stores the return in <code>\l_stex_kpsewhich_return_str</code> . This does not require shell escaping.
--------------------------------	---

3.1.1 Files, Paths, URIs

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{ \langle string \rangle \}$
<code>\stex_path_from_string:(NV cn cV)</code>	

turns the $\langle string \rangle$ into a path by splitting it at `/`-characters and stores the result in $\langle path-variable \rangle$. Also applies `\stex_path_canonicalize:N`.

<code>\stex_path_to_string:NN</code>	The inverse; turns a path into a string and stores it in the second argument variable, or
<code>\stex_path_to_string:N</code>	leaves it in the input stream.

<code>\stex_path_canonicalize:N</code>	Canonicalizes the path provided; in particular, resolves <code>.</code> and <code>..</code> path segments.
--	--

<code>\stex_path_if_absolute_p:N</code>	\star
<code>\stex_path_if_absolute:NTF</code>	\star

Checks whether the path provided is *absolute*, i.e. starts with an empty segment

<code>\c_stex_pwd_seq</code>	Store the current working directory as path-sequence and string, respectively, and the (heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</code> .
<code>\c_stex_pwd_str</code>	
<code>\c_stex_mainfile_seq</code>	
<code>\c_stex_mainfile_str</code>	

`\g_stex_currentfile_seq`

The file being currently processed (respecting `\input` etc.)

Test 1

```
\ExplSyntaxOn
\def\cpath@print#1{
\stex_path_from_string:Nn \l_tmpb_seq { #1 }
\stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
\str_use:N \l_tmpa_str
}
\ExplSyntaxOff
\begin{center}
\begin{tabular}{|l|l|l|}\hline
path & canonicalized path & expected\\\hline
aaa & \cpath@print{aaa} & aaa \\
.././aaa & \cpath@print{.././aaa} & & .././aaa \\
aaa/bbb & \cpath@print{aaa/bbb} & & aaa/bbb \\
aaa/. & \cpath@print{aaa/.} & & \\
.././aaa/bbb & \cpath@print{.././aaa/bbb} & & .././aaa/bbb \\
../aaa/./bbb & \cpath@print{../aaa/./bbb} & & ../bbb \\
../aaa/bbb & \cpath@print{../aaa/bbb} & & ../aaa/bbb \\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & & aaa/ddd \\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & & aaa/bbb/ddd \\
./ & \cpath@print{./} & & \\
aaa/bbb/./.. & \cpath@print{aaa/bbb/./..} & & \\
\end{tabular}
\end{center}
```

path	canonicalized path	expected
aaa	aaa	aaa
.././aaa	.././aaa	.././aaa
aaa/bbb	aaa/bbb	aaa/bbb
aaa/.		
.././aaa/bbb	.././aaa/bbb	.././aaa/bbb
../aaa/./bbb	../bbb	../bbb
../aaa/bbb	../aaa/bbb	../aaa/bbb
aaa/bbb/./ddd	aaa/ddd	aaa/ddd
aaa/bbb/./ddd	aaa/bbb/ddd	aaa/bbb/ddd
./		
aaa/bbb/./..		

3.1.2 MathHub Archives

`\mathhub`

`\c_stex_mathhub_seq`

`\c_stex_mathhub_str`

We determine the path to the local MathHub folder via one of three means, in order of precedence:

1. The `mathhub` package option, or
2. the `\mathhub`-macro, if it has been defined before the `\usepackage{stex}`-statement, or
3. the `MATHHUB` system variable.

In all three cases, `\c_stex_mathhub_seq` and `\c_stex_mathhub_str` are set accordingly.

`\l_stex_current_repository_prop`

Always points to the *current* MathHub repository (if we currently are in one). Has the fields `id`, `ns` (namespace), `narr` (narrative namespace; currently not in use) and `deps` (dependencies; currently not in use).

<hr/> <hr/> <code>\stex_set_current_repository:n</code>	Sets the current repository to the one with the provided ID. calls <code>__stex_mathhub_do_manifest:n</code> , so works whether this repository's MANIFEST.MF-file has already been read or not.
<hr/> <hr/> <code>\stex_require_repository:n</code>	Calls <code>__stex_mathhub_do_manifest:n</code> iff the corresponding archive property list does not already exist, and adds a corresponding definition to the <code>.sms</code> -file.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<repository-name>}{<code>}</code> Change the current repository to <code>{<repository-name>}</code> (or not, if <code>{<repository-name>}</code> is empty), and passes its ID on to <code>{<code>}</code> as #1. Switches back to the previous repository after executing <code>{<code>}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{<archive-ID>}{<filename>}</code> Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <code>\inputref:nn</code>	<code>\inputref[<archive-ID>]{<filename>}</code> <code>\inputs</code> the file <code><filename></code> in repository <code><archive-ID></code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```

\ExplSyntaxOn
\stex_require_repository:n { Foo/Bar }
id:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {id}\ \
narr:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {narr}\ \
ns:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {ns}\ \
deps:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {deps}\ \
\stex_require_repository:n { Bar/Foo }
\ExplSyntaxOff

```

```

id: Foo/Bar
narr:
ns: http://mathhub.info/tests/Foo/Bar
deps:

```

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\l_stex_current_module_prop`

All information of a module is stored as a property list. `\l_stex_current_module_prop` always points to the current module (if existent).

Most importantly, the `content`-field stores all the code to execute on activation; i.e. when this module is being included.

Additionally, it stores:

- The *name* in field `name`,
- the *namespace* in field `ns`,
- this module's *language* in field `lang`,
- if a language module that translates some other modules, the *original* module in field `sig` (for signature),
- the *metatheory* in field `meta`,
- the URIs of all *imported modules* in field `imports`,
- the names of all *declarations* in field `constants`,
- the *file* this module was declared in in field `file`,

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

A property list mapping file paths to the lists of all modules declared therein. `\g_stex_modules_in_file_seq` always points to the current file(-stream - `\inputs` are considered the same file).

```
\stex_if_in_module_p: * Conditional for whether we are currently in a module
\stex_if_in_module:TF *
```

```
\stex_if_module_exists_p:n *
\stex_if_module_exists:nTF *
```

Conditional for whether a module with the provided URI is already known.

```
\stex_add_to_current_module:n
\STEXexport
```

Adds the provided tokens to the `content` field of the current module.

```
\stex_add_constant_to_current_module:n
```

Adds the declaration with the provided name to the `constants` field of the current module.

```
\stex_add_import_to_current_module:n
```

Adds the module with the provided full URI to the `imports` field of the current module.

```
\stex_modules_compute_namespace:nN \stex_modules_compute_namespace:nN
{\<namespace>} {\<path>}
```

Computes the namespace for file `<path>` in repository with namespace `<namespace>` as follows:

If the file is `.../source/sub/file.tex` and the namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`.

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

```
\ExplSyntaxOn
\stex_modules_current_namespace:
Namespace~1:\\ \l_stex_modules_ns_str \\
Faking~a~repository:\\
\stex_set_current_repository:n{Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \testtemp
\edef\testtempb{\detokenize{source}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\edef\testtempb{\detokenize{test}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtemp }
\stex_modules_current_namespace:
Namespace~2:\\ \l_stex_modules_ns_str
\ExplSyntaxOff
```

```

Namespace 1:
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

.

5.1.1 The module-environment

`module` `\begin{module}[\langle options \rangle]{\langle name \rangle}`
 Opens a new module with name $\langle name \rangle$.
 TODO document options.

`\stex_module_setup:nn` `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`
 Sets up a new module with name $\langle name \rangle$ and optional parameters $\langle params \rangle$. In particular, sets `\l_stex_current_module_prop` appropriately.

`\stex_modules_heading:` Takes care of the module header, if the `showmods` package option is true. This macro can be overridden for customization.

`@module` `\begin{@module}[\langle options \rangle]{\langle name \rangle}`
 Core functionality of the `module-environment` without a header.

Test 4

```

\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\begin{@module}{Foo}
Module~path:-
\prop_item:Nn \l_stex_current_module_prop { ns }?
\prop_item:Nn \l_stex_current_module_prop { name }\\
Language:-\prop_item:Nn \l_stex_current_module_prop { lang }\\
Signature:-\prop_item:Nn \l_stex_current_module_prop { sig }\\
Metatheory:-\prop_item:Nn \l_stex_current_module_prop { meta }\\
\end{@module}
\ExplSyntaxOff

```

```

Module path: http://mathhub.info/tests/Foo/Bar?Foo
Language:
Signature:
Metatheory:

```

.

Test 5

```
\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\stex_debug:nn{modules}{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\stex_debug:nn{modules}{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\begin{module}[title=Foo Bar]{Bar}
Module-path:-
\prop_item:Nn \l_stex_current_module_prop { ns }?
\prop_item:Nn \l_stex_current_module_prop { name }\\
Language:-\prop_item:Nn \l_stex_current_module_prop { lang }\\
Signature:-\prop_item:Nn \l_stex_current_module_prop { sig }\\
Metatheory:-\prop_item:Nn \l_stex_current_module_prop { meta }\\
\end{module}
\ExplSyntaxOff
```

```
Module 5.1.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

`\STEXModule` `\STEXModule {⟨fragment⟩}`

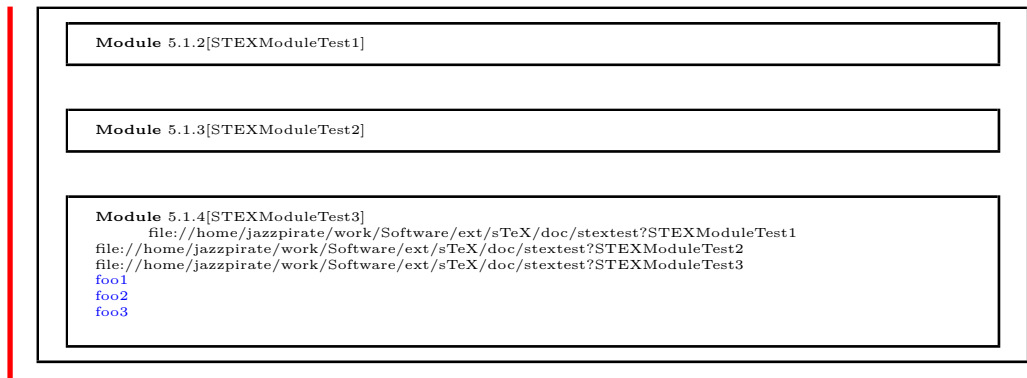
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

Invoked by `\STEXModule`. Needs to be followed either by `!⟨macro⟩` or `?{⟨symbolname⟩}`. In the first case, it stores the full URI in `⟨macro⟩`; in the second case, it invokes the symbol `⟨symbolname⟩` in the selected module.

Test 6

```
\begin{module}{STEXModuleTest1}
\symdecl{foo}
\end{module}
\begin{module}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\symdecl{foo}
\end{module}
\begin{module}{STEXModuleTest3}
\importmodule{STEXModuleTest2}
\symdecl{foo}
\STEXModule{STEXModuleTest1}!\teststring
\teststring\
\STEXModule{STEXModuleTest2}!\teststring
\teststring\
\STEXModule{STEXModuleTest3}!\teststring
\teststring\
\STEXModule{STEXModuleTest1}?{foo}[\comp{foo1}]\
\STEXModule{STEXModuleTest2}?{foo}[\comp{foo2}]\
\STEXModule{STEXModuleTest3}?{foo}[\comp{foo3}]\
\end{module}
```



`\stex_activate_module:n`

Activate the module with the provided URI; i.e. executes all macro code of the module's `content`-field (does nothing if the module is already activated in the current context) and adds the module to `\l_stex_all_modules_seq`.

Chapter 6

STEX-Module Inheritance

Code related to Module Inheritance, in particular *sms mode*.

6.1 Macros and Environments

6.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T_EX commands not explicitly allowed via the following lists:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. with the category code scheme used in SMS mode.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

Importantly, these macros need to call `\stex_smsmode_set_codes:` after reading all arguments. Note, that `\stex_smsmode_set_codes:` takes care of checking whether we are in SMS mode in the first place, so calling this function eagerly is unproblematic.

`\g_stex_smsmode_allowedenvs_seq`

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_escape_tl`, so `\stex_smsmode_set_codes:` should be called at the end of the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called with the SMS mode category code scheme active.

`\stex_if_smsmode_p: *`
`\stex_if_smsmode:TF *`

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

Sets the current category code scheme to that of the SMS mode, if SMS mode is currently active and if necessary.

This method should be called at the end of every macro or `\begin` environment code that are allowed in SMS mode.

`\stex_in_smsmode:nn`

`\stex_in_smsmode:nn {<name>} {<code>}`

Executes `<code>` in SMS mode. `<name>` can be arbitrary, but should be distinct, since it allows for nesting `\stex_in_smsmode:nn` without spuriously terminating SMS mode.

Test 7

```
\immediate\openout\testfile=./tests/sometest.tex
\immediate\write\testfile{\detokenize{\this is \a test}^J}
\immediate\write\testfile{\detokenize{this \is a \test}}
\immediate\closeout\testfile
\ExplSyntaxOn
\stex_in_smsmode:nn { foo } {
  \input{tests/sometest.tex}
}
\ExplSyntaxOff
```

6.1.2 Imports and Inheritance

`\importmodule`

`\importmodule[<archive-ID>]{<module-path>}`

Imports a module by reading it from a file and “activating” it. \TeX determines the module and its containing file by passing its arguments on to `\stex_import_module_path:nn`.

Test 8

```
\begin{module}{Foo}
\symdecl[name=foo, args=3]{bar}
\symdecl[ args=bai]{foobar}
Meaning:-\present\bar\
\end{module}
Meaning:-\present\bar\
\begin{module}{Importtest}
\importmodule{Foo}
Meaning:-\present\bar\
\end{module}
\begin{module}{Importtest2}
\importmodule{Importtest}
Meaning:-\present\bar\
\end{module}
```

Module 6.1.1[Foo]

Meaning: $\text{\macro:->\stex_invoke_symbol:n \{file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo\}}$

Meaning: $\text{\macro:->\protect \bar \ }$

Module 6.1.2[Importtest]

Meaning: $\text{\macro:->\stex_invoke_symbol:n \{file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo\}}$

Module 6.1.3[Importtest2]

Meaning: $\text{\macro:->\stex_invoke_symbol:n \{file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo\}}$

`\usemodule` `\importmodule[⟨archive-ID⟩]{⟨module-path⟩}`

Like `\importmodule`, but does not export its contents; i.e. including the current module will not activate the used module

Test 9

```

\begin{module}{UseTest1}
\symdecl{foo}
\end{module}
\begin{module}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:~\present\foo\\
\end{module}
\begin{module}{UseTest3}
\importmodule{UseTest2}
Meaning:~\present\foo\\
Meaning:~\present\bar\\

All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,~} \\
All~symbols:~
\seq_use:Nn \l_stex_all_symbols_seq {,~}
\ExplSyntaxOff
\end{module}

```

Module 6.1.4[UseTest1]

Module 6.1.5[UseTest2]
Meaning: `>macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest1?foo}<`

Module 6.1.6[UseTest3]
Meaning: `>undefined<`
Meaning: `>macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar}<`

All modules: `http://mathhub.info/sTeX?Metatheory`, `file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3`,
`file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2`
All symbols: `http://mathhub.info/sTeX?Metatheory?isa`, `http://mathhub.info/sTeX?Metatheory?bind`, `http://mathhub.info/sTeX?Metatheory?collec`,
`http://mathhub.info/sTeX?Metatheory?fromto`, `http://mathhub.info/sTeX?Metatheory?apply`, `http://mathhub.info/sTeX?Metatheory?collec`,
`http://mathhub.info/sTeX?Metatheory?seqtype`, `http://mathhub.info/sTeX?Metatheory?sequence-index`, `http://mathhub.info/sTeX?Metatheory?mathematical-structure`,
`http://mathhub.info/sTeX?Metatheory?aseqfromto`, `http://mathhub.info/sTeX?Metatheory?aseqfromtovia`, `http://mathhub.info/sTeX?Metatheory?module-type`, `http://mathhub.info/sTeX?Metatheory?mathematical-structure`,
`file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar`

Test 10

```

Circular dependencies:
\begin{module}{CircDep1}
\importmodule[Foo/Bar]{circular1?Circular1}
\importmodule[Bar/Foo]{circular2?Circular2}
\present\fooA\\
\present\fooB\\
\end{module}

```

Circular dependencies:

Module 6.1.7[CircDep1]
`>macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<`
`>macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo//circular2?Circular2?fooB}<`

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {<archive-ID>} {<module-path>}`

Determines the URI of a module by splitting `<module-path>` into `<path>?<name>`. If `<module-path>` does *not* contain a `?`-character, we consider it to be the `<name>`, and `<path>` to be empty.

If `<archive-ID>` is empty, it is automatically set to the ID of the current archive (if one exists).

1. If `<archive-ID>` is empty:

- (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`. That module should have the same namespace as the current one.

- (b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

- (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the top `source` folder of the archive, containing a module `<name>`.

That module should lie directly in the namespace of the archive.

- (b) If `<path>` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

If a module by that namespace exists, it is returned. Otherwise, we call `\stex_require_module:nn` on the `source` directory of the archive to find the file.

`\stex_import_require_module:nnnn`

`{<ns>} {<archive-ID>} {<path>} {<name>}`

Checks whether a module with URI `<ns>?<name>` already exists. If not, it looks for a plausible file that declares a module with that URI.

Finally, activates that module by executing its `content`-field.

Chapter 7

STEX-Symbols

Code related to symbol declarations and notations

7.1 Macros and Environments

<u><code>\symdecl</code></u>	<code>\symdecl[⟨args⟩]{⟨macroname⟩}</code>
------------------------------	--

Declares a new symbol with semantic macro `\macroname`. Optional arguments are:

- **name**: An (OMDOC) name. By default equal to `⟨macroname⟩`.
- **type**: An (ideally semantic) term. Not used by STEX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:
 - i a “normal” argument, e.g. `\symdecl[args=ii]{plus}` allows for `\plus{2}{2}`.
 - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl[args=a]{plus}` allows for `\plus{2,2,2}`.
 - b a *variable* argument. Is treated by STEX like an i-argument, but an application is turned into an OMBind in OMDoc, binding the provided variable in the subsequent arguments of the operator; e.g. `\symdecl[args=bi]{forall}` allows for `\forall{x\in\Nat}{x\geq0}`.

`\stex_symdecl_do:n`

Implements the core functionality of `\symdecl`, and is called by `\symdecl` and `\symdef`.

Ultimately stores the symbol $\langle URI \rangle$ in the property list `\g_stex_symdecl_⟨URI⟩_prop` with fields:

- `name` (string),
- `module` (string),
- `notations` (sequence of strings; initially empty),
- `local` (boolean),
- `type` (token list),
- `args` (string of is, as and bs),
- `arity` (integer string),
- `assocs` (integer string; number of associative arguments),

Test 11

```
\begin{module}{SymdeclTest}
\symdecl[name=foo, args=3]{bar}
\symdecl[name=foobar, args=iab]{bari}
\symdecl[def=\bar* abc]{bardef}
\ExplSyntaxOn
Meaning:~\present\bar\\
\stex_get_symbol:n { bar }
Result:~\l_stex_get_symbol_uri_str\\
Meaning:~\present\bardef\\
\ExplSyntaxOff
\end{module}
```

```
Module 7.1.1[SymdeclTest]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo}<
Result: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?bardef}<
```

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...

`\notation`

`\notation[⟨args⟩]{⟨symbol⟩}{⟨notations+⟩}`

Introduces a new notation for $\langle symbol \rangle$, see `\stex_notation_do:nn`

`\stex_notation_do:nn` `\stex_notation_do:nn{<URI>}{<notations+>}`

Implements the core functionality of `\notation`, and is called by `\notation` and `\symdef`.

Ultimately stores the notation in the property list `\g_stex_notation_<URI>#<variant>#<lang>_prop` with fields:

- symbol (URI string),
- language (string),
- variant (string),
- opprec (integer string),
- argprecs (sequence of integer strings)

Test 12

```
\begin{module}{NotationTest}
\importmodule{Foo}
\notation{foo, prec=500;20x20x20}{bar}{\comp\langle {#1} ^ {#2} _ {#3} \comp\rangle }
\notation{foo, prec=500;20x20x20}{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^ {#3} \comp\rangle }{ {#1}_\comp{#2}}
\end{module}
```

Module 7.1.2[NotationTest]

`\symdef` `\symdef[<args>]{<symbol>}{<notations+>}`

Combines `\symdecl` and `\notation` by introducing a new symbol and assigning a new notation for it.

Test 13

```
\begin{module}{SymdefTest}
\symdef[ args=a, prec=50]{plus}{ #1 }{#1 \comp+ #2}
$\plus{a,b,c}$
\end{module}
```

Module 7.1.3[SymdefTest]
 $a+b+c$

Chapter 8

STEX-Terms

Code related to symbolic expressions, typesetting notations, notation components, etc.

8.1 Macros and Environments

<hr/> <hr/> <code>\STEXsymbol</code>	Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>
<hr/> <hr/> <code>\symref</code>	<code>\symref{<symbol>}{<text>}</code> shortcut for <code>\STEXsymbol{<symbol>}! [<text>]</code>
<hr/> <hr/> <code>\stex_invoke_symbol:n</code>	Executes a semantic macro. Outside of math mode or if followed by <code>*</code> , it continues to <code>\stex_term_custom:nn</code> . In math mode, it uses the default or optionally provided notation of the associated symbol. If followed by <code>!</code> , it will invoke the symbol <i>itself</i> rather than its application (and continue to <code>\stex_term_custom:nn</code>), i.e. it allows to refer to <code>\plus!</code> [addition] as an operation, rather than <code>\plus[addition of]{some}{terms}</code> .
<hr/> <hr/> <code>_stex_term_math_oms:nnnn</code> <code>_stex_term_math_oma:nnnn</code> <code>_stex_term_math_omb:nnnn</code>	<code><URI><fragment><precedence><body></code> Annotates <code><body></code> as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol <code><URI></code> , generated by the specific notation <code><fragment></code> with (upwards) operator precedence <code><precedence></code> . Inserts parentheses according to the current downwards precedence and operator precedence.
<hr/> <hr/> <code>_stex_term_math_arg:nnn</code>	<code>\stex_term_arg:nnn<int><prec><body></code> Annotates <code><body></code> as the <code><int></code> th argument of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> .
<hr/> <hr/> <code>_stex_term_math_assoc_arg:nnnn</code>	<code>\stex_term_arg:nnn<int><prec><notation><body></code> Annotates <code><body></code> as the <code><int></code> th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> and associative notation <code><notation></code> .

<hr/> <code>\infprec</code> <code>\neginfprec</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code> Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \S I E X brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <code>\withbrackets</code> <hr/>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code> Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \S I E X for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$. Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.

Test 14

```

\begin{module}{MathTest1}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1} ^ {#2} _{#3} \comp\rangle }
$\bar{abc}$ and $\bar{foo} abc$.
\end{module}

```

Module 8.1.1[MathTest1]
 $\langle x20x20a^b{}_c \rangle$ and $\langle x20x20a^b{}_c \rangle$.

Test 15

```

\begin{module}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle }{ {#1}_{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle } }
$\foobar{a}{b,c,d,e,f}g$ and $\foobar{foo}{a}{b,c}g$ and $\foobar{abc}$

\symdecl[ args=a]{ plus }
\symdecl[ args=a]{ mult }
\notation[ prec=50]{ plus }{#1}{#1 \comp+ #2}
\notation[ prec=100]{ mult }{#1}{#1 \comp\cdot #2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac{ab}{c}}}$
\[\plus{a,\mult{b,c}}\text{ and }\mult{a,\plus{\frac{ab}{c}}}\]
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[]{$\displaystyle \mult{a,\plus{\frac{ab}{c}}}$}
\end{module}

```

Module 8.1.2[MathTest2]
 $\langle x20x20a|[b,c,d,e,f]{}^g \rangle$ and $\langle x20x20a|[b,c]{}^g \rangle$ and $\langle x20x20a|[b]{}^c \rangle$
 $a+(b\cdot c)$ and $a\cdot\frac{a}{b}+\frac{a}{c}$
 $a+(b\cdot c)$ and $a\cdot\frac{a}{b}+\frac{a}{c}$

`\stex_term_custom:nn`

`\stex_term_custom:nn{<URI>}{<args>}`

Implements custom one-time notation. Invoked by `\stex_invoke_symbol:n` in text mode, or if followed by `*` in math mode, or whenever followed by `!`.

Test 16

```
\begin{module}{TextTest}
\importmodule{Foo}

\bar[some ]a[ and some ]b[ and also some ]c[ here].

$\bar*[\text{some }]a[\text{ and some }]b[\text{ and also some }]c[\text{ here}]\$.

$\bar![\mathtt{bar}]$

\bar*{a}*{b}[or just some ]c

\bar![bar]

\bar[or first ]*[2]{b}[ , then ]*[3]{c}[ , and finally ]a

\end{module}
```

```
Module 8.1.3[TextTest]
  some a and some b and also some c here.
  some a and some b and also some c here.

  or just some c
  bar
  or first b, then c, and finally a
```

`\stex_highlight_term:nn`

`\stex_highlight_term:nn{<URI>}{<args>}`

Establishes a context for `\comp`. Stores the URI in a variable so that `\comp` knows which symbol governs the current notation.

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

Marks `<args>` as a notation component of the current symbol for highlighting, linking, etc.

The precise behavior is governed by `\@comp`, which takes as additional argument the URI of the current symbol. By default, `\@comp` adds the URI as a PDF tooltip and colors the highlighted part in blue.

`\@defemph` behaves like `\@comp`, and can be similarly redefined, but marks an expression as *definiendum* (used by `\definiendum`)

`\STEXinvisible`

Exports its argument as OMDOC (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.

`\ellipses`

TODO

Chapter 9

STEX-Structural Features

Code related to structural features

9.1 Macros and Environments

Structures

`mathstructure` TODO

Test 17

```
\begin{module}{StructureTest1}
\begin{mathstructure}[name=Magma]{magma}
\symdef{universe}{\comp M}
\symdef{args=2}{op}{#1 \comp\circ #2}
 $\text{\isa{\op ab}\universe}$ 
\end{mathstructure}

\ExplSyntaxOn
\prop_get:NnN \g_stex_last_feature_prop {fields} \l_tmpa_seq
\seq_use:Nn \l_tmpa_seq {,}
\ExplSyntaxOff

\present\magma

\instantiate{magma}[
universe ! {{\comp U}},
op ! {{#1 \comp+ #2 }}
]{mM}
\notation[op = U]{mM/universe}{\comp U}
\notation[op = +]{mM/op}{#1 \comp+ #2}

Test:  $\text{\mM{op}ab}$ 

Test2:  $\text{\mM{}}$ 
\end{module}
```

```
Module 9.1.1[StructureTest1]
aob:M
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/S
feature?op
>macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}<
Test: a+b
Test2: <U,+>
```

Chapter 10

sTeX-Statements

Code related to statements, e.g. definitions, theorems

10.1 Macros and Environments

`symboldoc` `\begin{<symboldoc>}{<symbols>} <text> \end{<symboldoc>}`
Declares *<text>* to be a (natural language, encyclopaedic) description of *{<symbols>}*
(a comma separated list of symbol identifiers).

Chapter 11

sTeX-Metatheory

The default meta theory for an sTeX module. Contains symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) “type”-annotations, or required for basic structuring principles (theorems, definitions).

Foundations should ideally instantiate these symbols with their formal counterparts, e.g. `isa` corresponds to a typing operation in typed setting, or the \in -operator in set-theoretic contexts; `bind` corresponds to a universal quantifier in (n th-order) logic, or a Π in dependent type theories.

11.1 Symbols

Part III
Extensions

Chapter 12

Tikzinput

12.1 Macros and Environments

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

Chapter 13

document-structure.sty: Semantic Markup for Open Mathematical Documents in L^AT_EX

The `omdoc` package is part of the $\S\mathrm{T}_\mathrm{E}\mathrm{X}$ collection, a version of $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ that allows to markup $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ documents semantically without leaving the document format, essentially turning $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in $\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$. This includes a simple structure sharing mechanism for $\S\mathrm{T}_\mathrm{E}\mathrm{X}$ that allows to move from a copy-and-paste document development model to a copy-and-reference model, which conserves space and simplifies document management. The augmented structure can be used by MKM systems for added-value services, either directly from the $\S\mathrm{T}_\mathrm{E}\mathrm{X}$ sources, or after translation.

13.1 Introduction

$\S\mathrm{T}_\mathrm{E}\mathrm{X}$ is a version of $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ that allows to markup $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ documents semantically without leaving the document format, essentially turning $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

The `omdoc` package supplies macros and environments that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the $\S\mathrm{T}_\mathrm{E}\mathrm{X}$ sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the $\S\mathrm{T}_\mathrm{E}\mathrm{X}$ collection.

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document

13.2 The User Interface

The `omdoc` package generates two files: `omdoc.cls`, and `omdoc.sty`. The `OMDOC` class is a minimally changed variant of the standard `article` class that includes the functionality provided by `omdoc.sty`. The rest of the documentation pertains to the functionality introduced by `omdoc.sty`.

13.2.1 Package and Class Options

The `omdoc` class accept the following options:

<code>class=<name></code>	load <code><name>.cls</code> instead of <code>article.cls</code>
<code>topsect=<sect></code>	The top-level sectioning level; the default for <code><sect></code> is <code>section</code>
<code>showignores</code>	show the the contents of the <code>ignore</code> environment after all
<code>showmeta</code>	show the metadata; see <code>metakeys.sty</code>
<code>showmods</code>	show modules; see <code>modules.sty</code>
<code>extrefs</code>	allow external references; see <code>sref.sty</code>
<code>defindex</code>	index definienda; see <code>statements.sty</code>
<code>minimal</code>	for testing; do not load any \TeX packages

The `omdoc` package accepts the same except the first two.

13.2.2 Document Structure

document

\documentkeys

id

The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble². This can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the \LaTeX ML transformation.

omgroup

id

creators

contributors

short

loadmodules

The structure of the document is given by the `omgroup` environment just like in `OMDOC`. In the \LaTeX route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the `omgroup`. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

```

\begin{module}{foo}
\symdef{bar}{B^a_r}
...
\begin{omgroup}[id=sec.barderiv,loadmodules]{Introducing $\protect\bar$ Derivations}

```

blindomgroup

\TeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a subsection* as an introduction for a chapter). Therefore the `omdoc` package provides a variant `blindomgroup`

⁴EDNOTE: integrate with latexml's XMRef in the Math mode.

²We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 1 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter³ and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

```
\begin{document}
\begin{blindomgroup}
\begin{blindomgroup}
\begin{frontmatter}
\maketitle\newpage
\begin{omgroup}[display=flow]{Preface}
... <<preface>> ...
\end{omgroup}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindomgroup}
... <<introductory remarks>> ...
\end{blindomgroup}
\begin{omgroup}{Introduction}
... <<intro>> ...
\end{omgroup}
... <<more chapters>> ...
\bibliographystyle{alpha}\bibliography{kwarc}
\end{document}
```

Example 1: A typical Document Structure of a Book

`\skipomgroup`

The `\skipomgroup` “skips an `omgroup`”, 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 `\skipomgroup`.

`\currentsectionlevel`

`\CurrentSectionLevel`

The `\currentsectionlevel` macro supplies the name of the current sectioning level, e.g. “chapter”, or “subsection”. `\CurrentSectionLevel` is the capitalized variant. They are useful to write something like “In this `\currentsectionlevel`, we will...” in an `omgroup` environment, where we do not know which sectioning level we will end up.

13.2.3 Ignoring Inputs

`ignore`
`showignores`

The `ignore` environment can be used for hiding text parts from the document structure. The body of the environment is not PDF or DVI output unless the `showignores` option

³We shied away from redefining the `frontmatter` to induce a `blindomgroup`, but this may be the “right” way to go in the future.

is given to the `omdoc` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

editing One may want to hide unfinished or obsolete parts of a document

narrative/content markup In \LaTeX we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh20d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an `ignore` and referenced by the `verbalizes` key in `\inlinedef`.

For prematurely stopping the formatting of a document, \LaTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `omgroup` environment 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 [LMH].

13.2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy` [`\URL`]{`\label`}, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL [`\URL`] that lets \LaTeX ML generate the correct reference.

`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in \LaTeX . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.⁵

13.2.5 Global Variables

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 \LaTeX preamble of the course notes file. `\setSGvar`{`\vname`}{`\text`} to set the global variable `\vname` to `\text` and `\useSGvar`{`\vname`} to reference it.

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.

⁵EdNOTE: document LMID und LMXRef here if we decide to keep them.

13.2.6 Colors

For convenience, the `omdoc` package defines a couple of color macros for the `color` package: For instance `\blue` abbreviates `\textcolor{blue}`, so that `\blue{<something>}` writes *<something>* in blue. The macros `\red`, `\green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

13.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `TeX` GitHub repository [\[sTeX\]](#).

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made.

Chapter 14

Slides and Course Notes

We present a document class from which we can generate both course slides and course notes in a transparent way.

14.1 Introduction

The `mikoslides` document class is derived from `beamer.cls` [Tana], it adds a “notes version” for course notes derived from the `omdoc` class [Kohlhase:smomdl] that is more suited to printing than the one supplied by `beamer.cls`.

14.2 The User Interface

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

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the `mikoslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

14.2.1 Package Options

The `mikoslides` class takes a variety of class options:⁶

- | | |
|---------------------------|---|
| <code>slides</code> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code> | Section 14.2.2). |
| <code>sectocframes</code> | • If the option <code>sectocframes</code> is given, then for the <code>omgroups</code> , special frames with the <code>omgroup</code> title (and number) are generated. |

<code>showmeta</code>	<ul style="list-style-type: none"> • <code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> • If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 14.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.
<code>topsect</code>	<ul style="list-style-type: none"> • <code>topsect=<sect></code> can be used to specify the top-level sectioning level; the default for <code><sect></code> is <code>section</code>.

14.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.
`note` The `mikoslides` class adds the `note` environment for encapsulating the course note fragments.⁴

⚠ 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 L^AT_EX becomes confused and throws error messages that are difficult to decipher.

```
\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}
...
```

Example 2: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 2.

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

⁶EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

⁴MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L^AT_EX trickery. Hints to the author are welcome.

⚠: 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 `mikoslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

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 from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

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

14.2.3 Header and Footer Lines of the Slides

The default logo provided by the `mikoslides` package is the \LaTeX logo it can be customized using `\setslidelogo{<logo name>}`.

The default footer line of the `mikoslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder. By default it is *Michael Kohlhase* in the `mikoslides` package since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name. 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.

14.2.4 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 \LaTeX notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] 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.⁷

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

⁷EdNOTE: MK: the `hyperref` link does not seem to work yet. I wonder why but do not have the time to fix it.

`\mhframeimage{baz/foobar}`

14.2.5 Colors and Highlighting

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

14.2.6 Front Matter, Titles, etc.

14.2.7 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}{../ex/founif}{We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

`\excursion` The `\excursion{<ref>}{<path>}{<text>}` is syntactic sugar for
`\activateexcursion` `\begin{nomtext}[title=Excursion]`
 `\activateexcursion{founif}{../ex/founif}`
 We will cover first-order unification in `\sref{founif}`.
 `\end{nomtext}`

`\activateexcursion` where `\activateexcursion{<path>}` augments the `\printexcursions` macro by a
`\printexcursions` 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` `\excursionref{<label>}` for that.

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

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

14.2.8 Miscellaneous

14.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the [sTeXGitHub](#) repository [\[sTeX\]](#).

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` 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 `omdoc` package.

Chapter 15

problem.sty: An Infrastructure for formatting Problems

The `problem` package supplies an infrastructure that allows specify problems and to reuse them efficiently in multiple environments.

15.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁵. 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.

15.2 The User Interface

15.2.1 Package Options

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> 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.
<code>mh</code>	The <code>mh</code> option turns on MathHub support; see [<code>Kohlhase:mss</code>].
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [<code>Kohlhase:metakeys</code>]
	for details and customization options).

⁵for the moment multiple choice problems are not supported, but may well be in a future version

15.2.2 Problems and Solutions

problem The main environment provided by the **problem** package 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. For an example of a marked up problem see Figure 3 and the resulting markup see Figure 4.

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{problem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants 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}[for=elephants,height=3cm]
    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{problem}
\end{document}
```

Example 3: A marked up Problem

solution The **solution** environment can be to specify a solution to a problem. If the **solutions** option 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).

```
Problem0.0 ()
How many Elephants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


Solution: Four, two in the front seats, and two in the back.


---


```

Example 4: The Formatted Problem from Figure 3

hint The **hint** and **exnote** environments can be used in a **problem** environment to give hints and to make notes that elaborate certain aspects of the problem.

gnote The **gnote** (grading notes) environment can be used to document situations that

may arise in grading.

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.

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.

15.2.3 Multiple Choice Blocks

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc[⟨keyvals⟩]{⟨text⟩}` macro, which 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` • `T` for true answers, `F` for false ones,
- `F` • `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `Ttext` • `feedback` for a short feedback text given to the student.
- `Ftext`
- `feedback`

See Figure ?? for an example

15.2.4 Including Problems

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.

15.2.5 Reporting Metadata

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` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

15.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `STEXGitHub` repository [[sTeX](#)].

1. none reported yet

```

\begin{problem}[title=Functions]
  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=Noooooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{problem}

```

Problem0.0 ()

What is the keyword to introduce a function definition in python?

1. def
2. function
3. fun
4. public static void

Problem0.0 ()

What is the keyword to introduce a function definition in python?

1. def
!
2. function
that is for C and C++
3. fun
that is for Standard ML
4. public static void
that is for Java

Example 5: A Problem with a multiple choice block

Chapter 16

`hwexam.sty/cls`: An Infrastructure for formatting Assignments and Exams

The `hwexam` package and class allows individual course assignment sheets and compound assignment documents using problem files marked up with the `problem` package.

Contents

16.1 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 [Kohlhase:problem]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

16.2 The User Interface

16.2.1 Package and Class Options

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).

`showmeta` If the `showmeta` option is set, then the metadata keys are shown (see [Kohlhase:metakeys] for details and customization options).

The `hwexam` class additionally accepts the options `report`, `book`, `chapter`, `part`, and `showignores`, of the `omdoc` package [Kohlhase:smomdl] on which it is based and passes them on to that. For the `extrefs` option see [Kohlhase:sref].

16.2.2 Assignments

`assignment` 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).

16.2.3 Typesetting Exams

`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 L^AT_EX source.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. `\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.

`testheading` Finally, 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.

16.2.4 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.

16.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `STEX`GitHub repository [\[sTeX\]](#).

1. none reported yet.

Name:
MatriculationNumber:

2021-12-20

You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

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

[illegible]

Example 6: A generated test heading.

Part IV

Implementation

Chapter 17

STEX -Basics Implementation

17.1 The STEXDocument Class

The `stex` document class is pretty straight-forward: It largely extends the `standalone` package and loads the `stex` package, passing all provided options on to the package.

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

17.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-compatible}
22 \RequirePackage{morewrites}
23
24 Package options:
25 \keys_define:nn { stex } {
26   debug      .clist_set:N = \c_stex_debug_clist ,
27   showmods   .bool_set:N = \c_stex_showmods_bool ,
```

```

26 lang      .clist_set:N = \c_stex_languages_clist ,
27 mathhub   .tl_set_x:N  = \mathhub ,
28 sms       .bool_set:N  = \c_stex_persist_mode_bool ,
29 image     .bool_set:N  = \c_tikzinput_image_bool
30 }
31 \ProcessKeysOptions { stex }

\stex The sTeXlogo:
\TeX
32 \protected\def\stex{%
33   \ifundefined{texorpdfstring}%
34   {\let\texorpdfstring\@firstoftwo}%
35   }%
36   \texorpdfstring{\raisebox{-.5ex}{S\kern-.5ex\TeX}{sTeX}\xspace}%
37 }
38 \def\sTeX{\stex}

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 9.)

17.3 Messages and logging

```

39 <@@=stex_log>
   Warnings and error messages
40 \msg_new:nnn{stex}{error/unknownlanguage}{
41   Unknown~language:~#1
42 }
43 \msg_new:nnn{stex}{warning/nomathhub}{
44   MATHHUB~system~variable~not~found~and~no~
45   \detokenize{\mathhub}~value~set!
46 }
47 \msg_new:nnn{stex}{error/deactivated-macro}{
48   The~\detokenize{#1}~command~is~only~allowed~in~#2!
49 }

\stex_debug:nn A simple macro issuing package messages with subpath.
50 \cs_new_protected:Nn \stex_debug:nn {
51   \clist_if_in:NnTF \c_stex_debug_clist { all } {
52     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
53       \Debug~#1:~#2\\
54     }
55     \msg_none:nn{stex}{debug / #1}
56   }{
57     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
58       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
59         \Debug~#1:~#2\\
60       }
61       \msg_none:nn{stex}{debug / #1}
62     }
63   }
64 }

```

(End definition for `\stex_debug:nn`. This function is documented on page 9.)

Redirecting messages:

```

65 \clist_if_in:NnTF \c_stex_debug_clist {all} {

```

```

66     \msg_redirect_module:nnn{ stex }{ none }{ term }
67   }{
68     \clist_map_inline:Nn \c_stex_debug_clist {
69       \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
70     }
71   }
72
73   \stex_debug:nn{log}{debug~mode~on}

```

17.4 Persistence

```

74 <@@=stex_persist>

```

`\c_stex_persist_sms_iow` File variable used for the sms-File

```

75 \iow_new:N \c_stex_persist_sms_iow
76 \AddToHook{begindocument}{
77   \bool_if:NTF \c_stex_persist_mode_bool {
78     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
79   } {
80     \iow_open:Nn \c_stex_persist_sms_iow {\jobname.sms}
81   }
82 }
83 \AddToHook{enddocument}{
84   \bool_if:NF \c_stex_persist_mode_bool {
85     \iow_close:N \c_stex_persist_sms_iow
86   }
87 }

```

(End definition for \c_stex_persist_sms_iow.)

`\stex_add_to_sms:n` Adds the provided code to the .sms-file of the document.

```

88 \cs_new_protected:Nn \stex_add_to_sms:n {
89   \bool_if:NF \c_stex_persist_mode_bool {
90     \iow_now:Nn \c_stex_persist_sms_iow { #1 }
91   }
92 }

```

(End definition for \stex_add_to_sms:n. This function is documented on page 9.)

17.5 HTML Annotations

```

93 <@@=stex_annotate>
94 \RequirePackage{scalatex}

```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to SCALATEX:

```

95 \scalatex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}

```

`\if@latexml` Conditionals for LATEXML:

```

\latexml_if_p:
\latexml_if:TF
96 \ifcsname if@latexml\endcsname\else
97   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
98 \fi
99

```

```

100 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
101   \if@latexml
102     \prg_return_true:
103   \else:
104     \prg_return_false:
105   \fi:
106 }

```

(End definition for \if@latexml and \latexml_if:TF. These functions are documented on page 9.)

\l_stex_annotate_arg_tl Used by annotation macros to ensure that the HTML output to annotate is not empty.

```

\c_stex_annotate_emptyarg_tl
107 \tl_new:N \l__stex_annotate_arg_tl
108 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
109   \scalatex_if:TF {
110     \scalatex_direct_HTML:n { \c_ampersand_str lrm; }
111   }{-}
112 }

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

```

\__stex_annotate_checkempty:n
113 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
114   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
115   \tl_if_empty:NT \l__stex_annotate_arg_tl {
116     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
117   }
118 }

```

(End definition for __stex_annotate_checkempty:n.)

\l_stex_html_do_output_bool Whether to (locally) produce HTML output

```

\stex_if_do_html:
119 \bool_new:N \l_stex_html_do_output_bool
120 \bool_set_true:N \l_stex_html_do_output_bool
121 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
122   \bool_if:nTF \l_stex_html_do_output_bool
123     \prg_return_true: \prg_return_false:
124 }

```

(End definition for \l_stex_html_do_output_bool and \stex_if_do_html:. These functions are documented on page ??.)

\stex_suppress_html:n Whether to (locally) produce HTML output

```

125 \cs_new_protected:Nn \stex_suppress_html:n {
126   \exp_args:Nne \use:nn {
127     \bool_set_false:N \l_stex_html_do_output_bool
128     #1
129   }{
130     \stex_if_do_html:T {
131       \bool_set_true:N \l_stex_html_do_output_bool
132     }
133   }
134 }

```

(End definition for \stex_suppress_html:n. This function is documented on page ??.)

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EXML, S^CA^LT_EX, p^DF^LA^TE_X).

The p^DF^LA^TE_X-macros largely do nothing; the S^CA^LT_EX-implementations are pretty clear in what they do, the L^AT_EXML-implementations resort to perl bindings.

```
135 \scalatex_if:TF{
136   \cs_new_protected:Nn \stex_annotate:nnn {
137     \__stex_annotate_checkempty:n { #3 }
138     \scalatex_annotate_HTML:nn {
139       property="stex:#1" ~
140       resource="#2"
141     } {
142       \tl_use:N \l__stex_annotate_arg_tl
143     }
144   }
145   \cs_new_protected:Nn \stex_annotate_invisible:n {
146     \__stex_annotate_checkempty:n { #1 }
147     \scalatex_annotate_HTML:nn {
148       stex:visible="false" ~
149       style:display="none"
150     } {
151       \tl_use:N \l__stex_annotate_arg_tl
152     }
153   }
154   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
155     \__stex_annotate_checkempty:n { #3 }
156     \scalatex_annotate_HTML:nn {
157       property="stex:#1" ~
158       resource="#2" ~
159       stex:visible="false" ~
160       style:display="none"
161     } {
162       \tl_use:N \l__stex_annotate_arg_tl
163     }
164   }
165   \NewDocumentEnvironment{stex_annotate_env} { m m } {
166     \par
167     \scalatex_annotate_HTML_begin:n {
168       property="stex:#1" ~
169       resource="#2"
170     }
171   }{
172     \scalatex_annotate_HTML_end:
173   }
174 }{
175   \latexml_if:TF {
176     \cs_new_protected:Nn \stex_annotate:nnn {
177       \__stex_annotate_checkempty:n { #3 }
178       \mode_if_math:TF {
179         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
180           \tl_use:N \l__stex_annotate_arg_tl
181         }
182       }{
183         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
```

```

184         \tl_use:N \l__stex_annotate_arg_tl
185     }
186 }
187 }
188 \cs_new_protected:Nn \stex_annotate_invisible:n {
189     \__stex_annotate_checkempty:n { #1 }
190     \mode_if_math:TF {
191         \cs:w latexml@invisible@math\cs_end:{
192             \tl_use:N \l__stex_annotate_arg_tl
193         }
194     } {
195         \cs:w latexml@invisible@text\cs_end:{
196             \tl_use:N \l__stex_annotate_arg_tl
197         }
198     }
199 }
200 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
201     \__stex_annotate_checkempty:n { #3 }
202     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
203         \tl_use:N \l__stex_annotate_arg_tl
204     }
205 }
206 \NewDocumentEnvironment{stex_annotate_env} { m m } {
207     \par\begin{latexml@annotateenv}{#1}{#2}
208 }{
209     \end{latexml@annotateenv}
210 }
211 }{
212     \cs_new_protected:Nn \stex_annotate:nnn {#3}
213     \cs_new_protected:Nn \stex_annotate_invisible:n {}
214     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
215     \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{
216 }
217 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`.
These functions are documented on page 10.)

17.6 Languages

```

218 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

We store language abbreviations in two (mutually inverse) property lists:

```

219 \prop_const_from_keyval:Nn \c_stex_languages_prop {
220     en = english ,
221     de = ngerman ,
222     ar = arabic ,
223     bg = bulgarian ,
224     ru = russian ,
225     fi = finnish ,
226     ro = romanian ,
227     tr = turkish ,
228     fr = french
229 }

```

```

230
231 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
232   english   = en ,
233   ngerman   = de ,
234   arabic    = ar ,
235   bulgarian = bg ,
236   russian   = ru ,
237   finnish   = fi ,
238   romanian  = ro ,
239   turkish   = tr ,
240   french    = fr
241 }
242 % todo: chinese simplified (zhs)
243 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 10.)

we use the `lang`-package option to load the corresponding babel languages:

```

244 \clist_if_empty:NF \c_stex_languages_clist {
245   \clist_clear:N \l_tmpa_clist
246   \clist_map_inline:Nn \c_stex_languages_clist {
247     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
248       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
249     } {
250       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
251     }
252   }
253   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
254   \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
255 }

```

17.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

256 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
257   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
258   \def#1{
259     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
260   }
261 }

```

(End definition for `\stex_deactivate_macro:Nn`. This function is documented on page 10.)

`\stex_reactivate_macro:N`

```

262 \cs_new_protected:Nn \stex_reactivate_macro:N {
263   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
264 }

```

(End definition for `\stex_reactivate_macro:N`. This function is documented on page 10.)

```

265 \</package>

```


Chapter 18

STEX -MathHub Implementation

```
266 <*package>
267
268 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
269
270 <@@=stex_path>
271
272 Warnings and error messages
273 \msg_new:nnn{stex}{error/norepository}{
274   No~archive~#1~found~in~#2
275 }
276 \msg_new:nnn{stex}{error/notinarchive}{
277   Not~currently~in~an~archive,~but~\detokenize{#1}~
278   needs~one!
279 }
280 \msg_new:nnn{stex}{error/nofile}{
281   \detokenize{#1}~could~not~find~file~#2
282 }
```

18.1 Generic Path Handling

We treat paths as L^AT_EX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

```
\stex_path_from_string:Nn
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV
281 \cs_new_protected:Nn \stex_path_from_string:Nn {
282   \str_set:Nx \l_tmpa_str { #2 }
283   \str_if_empty:NTF \l_tmpa_str {
284     \seq_clear:N #1
285   }{
286     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
287     \sys_if_platform_windows:T{
288       \seq_clear:N \l_tmpa_tl
289       \seq_map_inline:Nn #1 {
290         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
291         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl

```

```

292     }
293     \seq_set_eq:NN #1 \l_tmpa_tl
294   }
295   \stex_path_canonicalize:N #1
296 }
297 }
298 \cs_generate_variant:Nn \stex_path_from_string:Nn
299 { NV, cn, cV }

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 11.)

```

\stex_path_to_string:NN
\stex_path_to_string:N
300 \cs_new_protected:Nn \stex_path_to_string:NN {
301   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
302 }
303
304 \cs_new:Nn \stex_path_to_string:N {
305   \seq_use:Nn #1 /
306 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 11.)

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
307 \str_const:Nn \c__stex_path_dot_str {.}
308 \str_const:Nn \c__stex_path_up_str {...}

```

(End definition for `\c__stex_path_dot_str` and `\c__stex_path_up_str`.)

`\stex_path_canonicalize:N` Canonicalizes the path provided; in particular, resolves . and .. path segments.

```

309 \cs_new_protected:Nn \stex_path_canonicalize:N {
310   \seq_if_empty:NF #1 {
311     \seq_clear:N \l_tmpa_seq
312     \seq_get_left:NN #1 \l_tmpa_tl
313     \str_if_empty:NT \l_tmpa_tl {
314       \seq_put_right:Nn \l_tmpa_seq {}
315     }
316     \seq_map_inline:Nn #1 {
317       \str_set:Nn \l_tmpa_tl { ##1 }
318       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
319         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
320           \seq_if_empty:NTF \l_tmpa_seq {
321             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
322               \c__stex_path_up_str
323             }
324           }{
325             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
326             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
327               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
328                 \c__stex_path_up_str
329               }
330             }{
331               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
332             }

```

```

333     }
334   }{
335     \str_if_empty:NF \l_tmpa_tl {
336       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
337     }
338   }
339 }
340 }
341 \seq_gset_eq:NN #1 \l_tmpa_seq
342 }
343 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 11.)

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:N \underline{TF}`

```

344 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
345   \seq_if_empty:NTF #1 {
346     \prg_return_false:
347   }{
348     \seq_get_left:NN #1 \l_tmpa_tl
349     \str_if_empty:NTF \l_tmpa_tl {
350       \prg_return_true:
351     }{
352       \prg_return_false:
353     }
354   }
355 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 11.)

18.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

356 \str_new:N\l_stex_kpsewhich_return_str
357 \cs_new_protected:Nn \stex_kpsewhich:n {
358   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
359   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
360   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
361 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 11.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

362 \sys_if_platform_windows:TF{
363   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
364 }{
365   \stex_kpsewhich:n{-var-value~PWD}
366 }
367
368 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
369 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
370 \stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 11.)

18.3 File Hooks and Tracking

371 `<@@=stex_files>`

We introduce hooks for file inputs that keep track of the absolute paths of files used. This will be useful to keep track of modules, their archives, namespaces etc.

Note that the absolute paths are only accurate in `\input`-statements for paths relative to the PWD, so they shouldn't be relied upon in any other setting than for \TeX -purposes.

`\g__stex_files_stack` keeps track of file changes

372 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

373 `\str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}`

374 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

375 `\c_stex_mainfile_str`

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 11.)

`\g_stex_currentfile_seq` Hooks for file inputs that push/pop `\g__stex_files_stack` to update `\c_stex_mainfile_seq`.

```

376 \seq_gclear_new:N\g_stex_currentfile_seq
377 \AddToHook{file/before}{
378   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
379   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
380     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
381   }{
382     \stex_path_from_string:Nn\g_stex_currentfile_seq{
383       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
384     }
385   }
386   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
387   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
388 }
389 \AddToHook{file/after}{
390   \seq_if_empty:NF\g__stex_files_stack{
391     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
392   }
393   \seq_if_empty:NTF\g__stex_files_stack{
394     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
395   }{
396     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
397     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
398   }
399 }

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 12.)

18.4 MathHub Repositories

```

400 <@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
401 \str_if_empty:NTF\mathhub{
402   \stex_kpsewhich:n{-var-value~MATHHUB}
403   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
404
405   \str_if_empty:NTF\c_stex_mathhub_str{
406     \msg_warning:nn{stex}{warning/nomathhub}
407   }{
408     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
409     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
410   }
411 }{
412   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
413   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
414     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
415       \c_stex_pwd_str/\mathhub
416     }
417   }
418   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
419   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
420 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 12.)

```

\__stex_mathhub_do_manifest:n
421 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
422   \str_set:Nx \l_tmpa_str { #1 }
423   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
424     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
425     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
426     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
427     \__stex_mathhub_find_manifest:N \l_tmpa_seq
428     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
429       \msg_error:nnnn{stex}{error/norepository}{#1}{
430         \stex_path_to_string:N \c_stex_mathhub_str
431       }
432     } {
433       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
434     }
435   }
436 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
437 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

`_stex_mathhub_find_manifest:N` Attempts to find the MANIFEST.MF in some file path and stores its path in `\l__stex_mathhub_manifest_file_seq`:

```

438 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
439   \seq_set_eq:NN \l_tmpa_seq #1
440   \bool_set_true:N \l_tmpa_bool
441   \bool_while_do:Nn \l_tmpa_bool {
442     \seq_if_empty:NTF \l_tmpa_seq {
443       \bool_set_false:N \l_tmpa_bool
444     }{
445       \file_if_exist:nTF{
446         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
447       }{
448         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
449         \bool_set_false:N \l_tmpa_bool
450       }{
451         \file_if_exist:nTF{
452           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
453         }{
454           \seq_put_right:Nn \l_tmpa_seq{META-INF}
455           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
456           \bool_set_false:N \l_tmpa_bool
457         }{
458           \file_if_exist:nTF{
459             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
460           }{
461             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
462             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
463             \bool_set_false:N \l_tmpa_bool
464           }{
465             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
466           }
467         }
468       }
469     }
470   }
471   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
472 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

473 \ior_new:N \c_stex_mathhub_manifest_ior

```

(End definition for `\c_stex_mathhub_manifest_ior`.)

`_stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```

474 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
475   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
476   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
477   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
478     \str_set:Nn \l_tmpa_str {##1}
479     \exp_args:NNoo \seq_set_split:Nnn
480       \l_tmpb_seq \c_colon_str \l_tmpa_str
481     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

482 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
483 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
484 }
485 \exp_args:No \str_case:nnTF \l_tmpa_tl {
486 {id} {
487 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
488 { id } \l_tmpb_tl
489 }
490 {narration-base} {
491 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
492 { narr } \l_tmpb_tl
493 }
494 {url-base} {
495 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
496 { docurl } \l_tmpb_tl
497 }
498 {source-base} {
499 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
500 { ns } \l_tmpb_tl
501 }
502 {ns} {
503 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
504 { ns } \l_tmpb_tl
505 }
506 {dependencies} {
507 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
508 { deps } \l_tmpb_tl
509 }
510 }{}{}
511 }{}
512 }
513 \ior_close:N \c__stex_mathhub_manifest_ior
514 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

515 \cs_new_protected:Nn \stex_set_current_repository:n {
516 \stex_require_repository:n { #1 }
517 \prop_set_eq:Nc \l_stex_current_repository_prop {
518 c_stex_mathhub_#1_manifest_prop
519 }
520 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 13.)

`\stex_require_repository:n`

```

521 \cs_new_protected:Nn \stex_require_repository:n {
522 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
523 \stex_debug:nn{mathhub}{Opening~archive:~#1}
524 \__stex_mathhub_do_manifest:n { #1 }
525 \exp_args:Nx \stex_add_to_sms:n {
526 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
527 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
528 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

529     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
530     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
531   }
532 }
533 }
534 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 13.)

`\l_stex_current_repository_prop` Current MathHub repository

```

535 \prop_new:N \l_stex_current_repository_prop
536
537 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
538 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
539   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
540 } {
541   \__stex_mathhub_parse_manifest:n { main }
542   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
543   \l_tmpa_str
544   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
545   \c_stex_mathhub_main_manifest_prop
546   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
547   \stex_debug:nn{mathhub}{Current~repository:~
548     \prop_item:Nn \l_stex_current_repository_prop {id}
549   }
550 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 12.)

`\stex_in_repository:nn` Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```

551 \cs_new_protected:Nn \stex_in_repository:nn {
552   \str_set:Nx \l_tmpa_str { #1 }
553   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
554   \str_if_empty:NTF \l_tmpa_str {
555     \exp_args:Ne \l_tmpa_cs{
556       \prop_item:Nn \l_stex_current_repository_prop { id }
557     }
558   }{
559     \stex_require_repository:n \l_tmpa_str
560     \str_set:Nx \l_tmpa_str { #1 }
561     \exp_args:Nne \use:nn {
562       \stex_set_current_repository:n \l_tmpa_str
563       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
564     }{
565       \stex_set_current_repository:n {
566         \prop_item:Nn \l_stex_current_repository_prop { id }
567       }
568     }
569   }
570 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 13.)

\inputref
\inputref:nn

```

571 \newif \ifinputref \inputreffalse
572
573 \cs_new_protected:Nn \inputref:nn {
574   \stex_in_repository:nn {#1} {
575     \ifinputref
576       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
577     \else
578       \inputreftrue
579       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
580       \inputreffalse
581     \fi
582   }
583 }
584 \NewDocumentCommand \inputref { 0{} m}{
585   \inputref:nn{ #1 }{ #2 }
586 }

```

(End definition for \inputref and \inputref:nn. These functions are documented on page 13.)

\mhp

```

587 \def \mhp #1 #2 {
588   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
589     \c_stex_mathhub_str /
590     \prop_item:Nn \l_stex_current_repository_prop { id }
591     / source / #2
592   }{
593     \c_stex_mathhub_str / #1 / source / #2
594   }
595 }

```

(End definition for \mhp. This function is documented on page 13.)

\libinput

```

596 \cs_new_protected:Npn \libinput #1 {
597   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
598     \msg_error:nnn{stex}{error/notinarchive}\libinput
599   }
600   \bool_set_false:N \l_tmpa_bool
601   \tl_clear:N \l_tmpa_tl
602   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
603   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
604   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
605   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
606     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
607     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
608       / meta-inf / lib / #1.tex}{
609       \bool_set_true:N \l_tmpa_bool
610       \tl_put_right:Nx \l_tmpa_tl {
611         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
612           / meta-inf / lib / #1.tex}
613       }
614     }{}
615   }

```

```

616 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
617 / \l_tmpa_str / lib / #1.tex
618 }{
619   \bool_set_true:N \l_tmpa_bool
620   \tl_put_right:Nx \l_tmpa_tl {
621     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
622       / \l_tmpa_str / lib / #1.tex}
623   }
624 }{}
625 \bool_if:NF \l_tmpa_bool {
626   \msg_error:nnnn{stex}{error/nofile}\libinput{#1.tex}
627 }
628 \l_tmpa_tl
629 }

```

(End definition for `\libinput`. This function is documented on page [13](#).)

```

630 </package>

```

Chapter 19

STEX -References Implementation

```
631 <*package>
632
633 %%%%%%%%%% references.dtx %%%%%%%%%%
634
635 %\RequirePackage{hyperref}
636 %\RequirePackage{cleveref}
637 <@@=stex_refs>
638
639 \iow_new:N \c__stex_refs_refs_iow
640 \AddToHook{begindocument}{
641   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
642 }
643 \AddToHook{enddocument}{
644   \iow_close:N \c__stex_refs_refs_iow
645 }
646
647 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
648
649 \NewDocumentCommand \STEXreftitle { m } {
650   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
651 }
```

19.1 Document URIs and URLs

```
652 \seq_new:N \g__stex_refs_all_refs_seq
653
654 \str_new:N \l_stex_current_docns_str
655
656 \cs_new_protected:Nn \stex_get_document_uri: {
657   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
658   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
659   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
660   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
```

```

661 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
662
663 \str_clear:N \l_tmpa_str
664 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
665   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
666 }
667
668 \str_if_empty:NTF \l_tmpa_str {
669   \str_set:Nx \l_stex_current_docns_str {
670     file:/\stex_path_to_string:N \l_tmpa_seq
671   }
672 }{
673   \bool_set_true:N \l_tmpa_bool
674   \bool_while_do:Nn \l_tmpa_bool {
675     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
676     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
677       {source} { \bool_set_false:N \l_tmpa_bool }
678     }{}{
679       \seq_if_empty:NT \l_tmpa_seq {
680         \bool_set_false:N \l_tmpa_bool
681       }
682     }
683   }
684
685   \seq_if_empty:NTF \l_tmpa_seq {
686     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
687   }{
688     \str_set:Nx \l_stex_current_docns_str {
689       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
690     }
691   }
692 }
693 }
694
695 \str_new:N \l_stex_current_docurl_str
696 \cs_new_protected:Nn \stex_get_document_url: {
697   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
698   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
699   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
700   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
701   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
702
703   \str_clear:N \l_tmpa_str
704   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
705     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
706       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
707     }
708   }
709
710   \str_if_empty:NTF \l_tmpa_str {
711     \str_set:Nx \l_stex_current_docurl_str {
712       file:/\stex_path_to_string:N \l_tmpa_seq
713     }
714   }{
715     \bool_set_true:N \l_tmpa_bool

```

```

715 \bool_while_do:Nn \l_tmpa_bool {
716   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
717   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
718     {source} { \bool_set_false:N \l_tmpa_bool }
719   }{}{
720     \seq_if_empty:NT \l_tmpa_seq {
721       \bool_set_false:N \l_tmpa_bool
722     }
723   }
724 }
725
726 \seq_if_empty:NTF \l_tmpa_seq {
727   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
728 }{
729   \str_set:Nx \l_stex_current_docurl_str {
730     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
731   }
732 }
733 }
734 }

```

19.2 Setting Reference Targets

```

735 \str_const:Nn \c__stex_refs_url_str{URL}
736 \str_const:Nn \c__stex_refs_ref_str{REF}
737 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
738   \stex_get_document_uri:
739   \str_set:Nx \l_tmpa_str { #1 }
740   \str_if_empty:NT \l_tmpa_str {
741     \int_zero:N \l_tmpa_int
742     \bool_set_true:N \l_tmpa_bool
743     \bool_while_do:Nn \l_tmpa_bool {
744       \cs_if_exist:cTF {
745         sref_\l_stex_current_docns_str\c_hash_str REF\int_use:N \l_tmpa_int _type
746       }{
747         \int_incr:N \l_tmpa_int
748       }{}
749       \str_set:Nx \l_tmpa_str { REF\int_use:N \l_tmpa_int }
750       \bool_set_false:N \l_tmpa_bool
751     }
752   }
753 }
754 \str_set:Nx \l_tmpa_str {
755   \l_stex_current_docns_str\c_hash_str\l_tmpa_str
756 }
757 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
758 \stex_if_smsmode:TF {
759   \stex_get_document_url:
760   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
761   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
762 }{
763   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~::~~\expandafter{\@currentlabel~in~\exp_a
764   \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
765   \str_gset:cn {sref_\l_tmpa_str _type}\c__stex_refs_ref_str

```

```

766 }
767 }
768 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
769   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
770 }

```

19.3 Using References

```

771 \keys_define:nn { stex / sref } {
772   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
773   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
774   pre           .tl_set:N = \l__stex_refs_pre_tl ,
775   post          .tl_set:N = \l__stex_refs_post_tl ,
776   indoc         .str_set_x:N = \l__stex_refs_repo_str ,
777 }
778
779 \cs_new_protected:Nn \__stex_refs_args:n {
780   \tl_clear:N \l__stex_refs_linktext_tl
781   \tl_clear:N \l__stex_refs_fallback_tl
782   \tl_clear:N \l__stex_refs_pre_tl
783   \tl_clear:N \l__stex_refs_post_tl
784   \str_clear:N \l__stex_refs_repo_str
785   \keys_set:nn { stex / sref } { #1 }
786 }
787
788 \</package>

```

Chapter 20

STEX -Modules Implementation

```
789 <*package>
790
791 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
792
793 <@@=stex_modules>
794
795     Warnings and error messages
796 \msg_new:nnn{stex}{error/unknownmodule}{
797     No~module~#1~found
798 }
799 \msg_new:nnn{stex}{error/syntax}{
800     Syntax~error:~#1
801 }
802 \msg_new:nnn{stex}{error/siglanguage}{
803     Module~#1~declares~signature~#2,~but~does~not~
804     declare~its~language
805 }
```

`\l_stex_current_module_prop` The current module:

```
804 \prop_new:N \l_stex_current_module_prop
```

(End definition for `\l_stex_current_module_prop`. This variable is documented on page 15.)

`\l_stex_all_modules_seq` Stores all available modules

```
805 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 15.)

`\g_stex_modules_in_file_seq` All modules sorted by containing file; used e.g. in `\importmodule`
`\g_stex_module_files_prop`

```
806 \seq_new:N \g_stex_modules_in_file_seq
807 \prop_new:N \g_stex_module_files_prop
```

(End definition for `\g_stex_modules_in_file_seq` and `\g_stex_module_files_prop`. These variables are documented on page 16.)

```

\stex_if_in_module_p:
\stex_if_in_module:TF
808 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
809   \prop_if_empty:NTF \l_stex_current_module_prop
810   \prg_return_false: \prg_return_true:
811 }

```

(End definition for \stex_if_in_module:TF. This function is documented on page 16.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
812 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
813   \prop_if_exist:cTF { c_stex_module_#1_prop }
814   \prg_return_true: \prg_return_false:
815 }

```

(End definition for \stex_if_module_exists:nTF. This function is documented on page 16.)

```

\stex_add_to_current_module:n
\STEXexport
816 \cs_new_protected:Nn \stex_add_to_current_module:n {
817   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
818   \tl_put_right:Nn \l_tmpa_tl { #1 }
819   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
820 }
821 \cs_new_protected:Npn \STEXexport {
822   \begingroup
823   \newlinechar=-1\relax
824   \endlinechar=-1\relax
825   %\catcode'\ = 9\relax
826   \expandafter\endgroup\STEXexport:n
827 }
828 \cs_new_protected:Nn \STEXexport:n {
829   \ignorespaces #1
830   \stex_add_to_current_module:n { \ignorespaces #1 }
831   \stex_smsmode_set_codes:
832 }
833 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented on page 16.)

```

\stex_add_constant_to_current_module:n
834 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
835   \str_set:Nx \l_tmpa_str { #1 }
836   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
837   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
838   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
839 }

```

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page 16.)

```

\stex_add_import_to_current_module:n
840 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
841   \str_set:Nx \l_tmpa_str { #1 }
842   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
843   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
844   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
845 }

```


(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 16.)

`\stex_modules_compute_namespace:nN` Computer the appropriate namespace from the top-level namespace of a repository (#1) and a file path (#2).

```

846 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
847   \str_set:Nx \l_tmpa_str { #1 }
848   \seq_set_eq:NN \l_tmpa_seq #2
849   % split off file extension
850   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
851   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
852   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
853   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
854
855   \bool_set_true:N \l_tmpa_bool
856   \bool_while_do:Nn \l_tmpa_bool {
857     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
858     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
859       {source} { \bool_set_false:N \l_tmpa_bool }
860     }{}{
861       \seq_if_empty:NT \l_tmpa_seq {
862         \bool_set_false:N \l_tmpa_bool
863       }
864     }
865   }
866
867   \seq_if_empty:NTF \l_tmpa_seq {
868     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
869   }{
870     \str_set:Nx \l_stex_modules_ns_str {
871       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
872     }
873   }
874 }

```

(End definition for `\stex_modules_compute_namespace:nN`. This function is documented on page 16.)

Stores its return values in:

`\l_stex_modules_ns_str`

```

875 \str_new:N \l_stex_modules_ns_str

```

(End definition for `\l_stex_modules_ns_str`. This variable is documented on page ??.)

`\stex_modules_current_namespace:` Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```

876 \cs_new_protected:Nn \stex_modules_current_namespace: {
877   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
878     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
879   }{
880     % split off file extension
881     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
882     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
883     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
884     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
885     \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

886     \str_set:Nx \l_stex_modules_ns_str {
887         file:/\stex_path_to_string:N \l_tmpa_seq
888     }
889 }
890 }

```

(End definition for `\stex_modules_current_namespace:`. This function is documented on page 16.)

20.1 The module environment

module arguments:

```

891 \keys_define:nn { stex / module } {
892     title          .str_set_x:N = \l_stex_module_title_str ,
893     ns             .str_set_x:N = \l_stex_module_ns_str ,
894     lang           .str_set_x:N = \l_stex_module_lang_str ,
895     sig            .str_set_x:N = \l_stex_module_sig_str ,
896     creators       .str_set_x:N = \l_stex_module_creators_str ,
897     contributors   .str_set_x:N = \l_stex_module_contributors_str ,
898     meta           .str_set_x:N = \l_stex_module_meta_str
899 }
900
901 \cs_new_protected:Nn \__stex_modules_args:n {
902     \str_clear:N \l_stex_module_title_str
903     \str_clear:N \l_stex_module_ns_str
904     \str_clear:N \l_stex_module_lang_str
905     \str_clear:N \l_stex_module_sig_str
906     \str_clear:N \l_stex_module_creators_str
907     \str_clear:N \l_stex_module_contributors_str
908     \str_clear:N \l_stex_module_meta_str
909     \keys_set:nn { stex / module } { #1 }
910 }
911
912 % module parameters here? In the body?
913

```

`\stex_module_setup:nn` Sets up a new module property list:

```

914 \cs_new_protected:Nn \stex_module_setup:nn {
915     \str_set:Nx \l_stex_module_name_str { #2 }
916     \__stex_modules_args:n { #1 }
917
918     First, we set up the name and namespace of the module.
919     Are we in a nested module?
920
921     \stex_if_in_module:TF {
922         % Nested module
923         \prop_get:NnN \l_stex_current_module_prop
924         { ns } \l_stex_module_ns_str
925         \str_set:Nx \l_stex_module_name_str {
926             \prop_item:Nn \l_stex_current_module_prop
927             { name } / \l_stex_module_name_str
928         }
929     }{
930         % not nested:
931         \str_if_empty:NT \l_stex_module_ns_str {

```

```

928     \stex_modules_current_namespace:
929     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
930     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
931       / {\l_stex_module_ns_str}
932     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
933     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
934       \str_set:Nx \l_stex_module_ns_str {
935         \stex_path_to_string:N \l_tmpa_seq
936       }
937     }
938   }
939 }

```

Next, we determine the language of the module:

```

940 \str_if_empty:NT \l_stex_module_lang_str {
941   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
942   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
943   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
944   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
945   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
946     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
947       inferred~from~file~name}
948     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
949   }
950 }
951
952 \str_if_empty:NF \l_stex_module_lang_str {
953   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
954   \l_tmpa_str {
955     \ltx@ifpackageloaded{babel}{
956       \exp_args:Nx \selectlanguage { \l_tmpa_str }
957     }{}
958   } {
959     \msg_error:nnn{stex}{error/unknownlanguage}{\l_tmpa_str}
960   }
961 }

```

We check if we need to extend a signature module, and set `\l_stex_current_module_prop` accordingly:

```

962 \str_if_empty:NTF \l_stex_module_sig_str {
963   \str_clear:N \l_tmpa_str
964   \seq_clear:N \l_tmpa_seq
965   \tl_clear:N \l_tmpa_tl
966   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
967     name      = \l_stex_module_name_str ,
968     ns        = \l_stex_module_ns_str ,
969     imports   = \exp_not:o { \l_tmpa_seq } ,
970     constants = \exp_not:o { \l_tmpa_seq } ,
971     content   = \exp_not:o { \l_tmpa_tl } ,
972     file      = \exp_not:o { \g_stex_currentfile_seq } ,
973     lang      = \l_stex_module_lang_str ,
974     sig       = \l_stex_module_sig_str ,
975     meta      = \l_stex_module_meta_str
976   }

```

```

977 }{
978   \str_if_empty:NT \l_stex_module_lang_str {
979     \msg_error:nnnn{stex}{error/siglanguage}{
980       \l_stex_module_ns_str?\l_stex_module_name_str
981     }\l_stex_module_sig_str}
982   }
983
984   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
985   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
986   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
987   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
988   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
989   \str_set:Nx \l_tmpa_str {
990     \stex_path_to_string:N \l_tmpa_seq /
991     \l_tmpa_str . \l_stex_module_sig_str .tex
992   }
993   \IfFileExists \l_tmpa_str {
994     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
995       \seq_clear:N \l_stex_all_modules_seq
996       \prop_clear:N \l_stex_current_module_prop
997       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
998       \input { \l_tmpa_str }
999     }
1000   }{
1001     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1002   }
1003   \stex_activate_module:n {
1004     \l_stex_module_ns_str ? \l_stex_module_name_str
1005   }
1006   \prop_set_eq:Nc \l_stex_current_module_prop {
1007     c_stex_module_
1008     \l_stex_module_ns_str ?
1009     \l_stex_module_name_str
1010     _prop
1011   }
1012 }

```

We load the metatheory:

```

1013 \str_if_empty:NT \l_stex_module_meta_str {
1014   \str_set:Nx \l_stex_module_meta_str {
1015     \c_stex_metatheory_ns_str ? Metatheory
1016   }
1017 }
1018 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1019   \exp_args:Nx \stex_add_to_current_module:n {
1020     \stex_activate_module:n {\l_stex_module_meta_str}
1021   }
1022   \stex_activate_module:n {\l_stex_module_meta_str}
1023 }
1024 }

```

(End definition for \stex_module_setup:nn. This function is documented on page 17.)

module The module environment.

```

\__stex_modules_begin_module:nn implements \begin{module}

1025 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1026   \stex_reactivate_macro:N \STEXexport
1027   \stex_reactivate_macro:N \importmodule
1028   \stex_reactivate_macro:N \symdecl
1029   \stex_reactivate_macro:N \notation
1030   \stex_reactivate_macro:N \symdef
1031   \stex_module_setup:nn{#1}{#2}
1032
1033   \stex_debug:nn{modules}{
1034     New~module:\\
1035     Namespace:~\l_stex_module_ns_str\\
1036     Name:~\l_stex_module_name_str\\
1037     Language:~\l_stex_module_lang_str\\
1038     Signature:~\l_stex_module_sig_str\\
1039     Metatheory:~\l_stex_module_meta_str\\
1040     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1041   }
1042
1043   \seq_put_right:Nx \l_stex_all_modules_seq {
1044     \l_stex_module_ns_str ? \l_stex_module_name_str
1045   }
1046
1047   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1048     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1049
1050   \stex_if_smsmode:TF {
1051     \stex_smsmode_set_codes:
1052   } {
1053     \begin{stex_annotate_env} {theory} {
1054       \l_stex_module_ns_str ? \l_stex_module_name_str
1055     }
1056
1057     \stex_annotate_invisible:nnn{header}{} {
1058       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1059       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1060       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1061         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1062       }
1063     }
1064   }
1065   % TODO: Inherit metatheory for nested modules?
1066 }
1067 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

(End definition for \__stex_modules_begin_module:nn.)

```

```

\__stex_modules_end_module: implements \end{module}

1068 \cs_new_protected:Nn \__stex_modules_end_module: {
1069   \str_set:Nx \l_tmpa_str {
1070     c_stex_module_
1071     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1072     \prop_item:Nn \l_stex_current_module_prop { name }
1073     _prop

```

```

1074 }
1075 %^^A \prop_new:c { \l_tmpa_str }
1076 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1077 \stex_debug:nn{modules}{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
1078 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1079 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1080 \NewDocumentEnvironment { @module } { 0{} m } {
1081   \par
1082   \_stex_modules_begin_module:nn{#1}{#2}
1083 } {
1084   \_stex_modules_end_module:
1085   \stex_if_smsmode:TF {
1086     \exp_args:Nx \stex_add_to_sms:n {
1087       \prop_gset_from_keyval:cn {
1088         c_stex_module_
1089         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1090         \prop_item:Nn \l_stex_current_module_prop { name }
1091         _prop
1092       } {
1093         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1094         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1095         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1096         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1097         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1098         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1099         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1100         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1101         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1102       }
1103     }
1104   }{
1105     \end{stex_annotate_env}
1106   }
1107 }

```

\stex_modules_heading: Code for document headers

```

1108 \cs_if_exist:NTF \thesection {
1109   \newcounter{module}[section]
1110 }{
1111   \newcounter{module}
1112 }
1113
1114 \bool_if:NT \c_stex_showmods_bool {
1115   \latexml_if:F { \RequirePackage{mdframed} }
1116 }
1117
1118 \cs_new_protected:Nn \stex_modules_heading: {
1119   \stepcounter{module}
1120   \par
1121   \bool_if:NT \c_stex_showmods_bool {

```

```

1122     \noindent{\textbf{Module} ~
1123     \cs_if_exist:NT \thesection {\thesection.}
1124     \themodule ~ [\l_stex_module_name_str]
1125   }
1126   \str_if_empty:NTF \l_stex_module_title_str {
1127   }{
1128     \quad(\l_stex_module_title_str)\hfill
1129   }\par
1130 }
1131 \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1132 % TODO
1133 \stex_ref_new_doc_target:n \l_stex_module_name_str
1134 }

```

(End definition for `\stex_modules_heading:`. This function is documented on page 17.)

Finally:

```

1135 \NewDocumentEnvironment { module } { 0 } { m } {
1136   \bool_if:NT \c_stex_showmods_bool {
1137     \begin{mdframed}
1138   }
1139   \begin{@module}[#1]{#2}
1140   \stex_modules_heading:
1141 }{
1142   \end{@module}
1143   \bool_if:NT \c_stex_showmods_bool {
1144     \end{mdframed}
1145   }
1146 }

```

20.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```

1147 \NewDocumentCommand \STEXModule { m } {
1148   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1149   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1150   \tl_set:Nn \l_tmpa_tl {
1151     \msg_error:nnn{stex}{error/unknownmodule}{#1}
1152   }
1153   \seq_map_inline:Nn \l_stex_all_modules_seq {
1154     \str_set:Nn \l_tmpb_str { ##1 }
1155     \str_if_eq:eeT { \l_tmpa_str } {
1156       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1157     } {
1158       \seq_map_break:n {
1159         \tl_set:Nn \l_tmpa_tl {
1160           \stex_invoke_module:n { ##1 }
1161         }
1162       }
1163     }
1164   }
1165   \l_tmpa_tl
1166 }
1167

```

```

1168 \cs_new_protected:Nn \stex_invoke_module:n {
1169   \stex_debug:nn{modules}{Invoking~module~#1}
1170   \peek_charcode_remove:NTF ! {
1171     \__stex_modules_invoke_uri:nN { #1 }
1172   } {
1173     \peek_charcode_remove:NTF ? {
1174       \__stex_modules_invoke_symbol:nn { #1 }
1175     } {
1176       \msg_error:nnn{stex}{error/syntax}{
1177         ?~or~!~expected~after~
1178         \c_backslash_str STEXModule{#1}
1179       }
1180     }
1181   }
1182 }
1183
1184 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1185   \str_set:Nn #2 { #1 }
1186 }
1187
1188 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1189   \stex_invoke_symbol:n{#1?#2}
1190 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 18.)

`\stex_activate_module:n`

```

1191 \cs_new_protected:Nn \stex_activate_module:n {
1192   \stex_debug:nn{modules}{Activating~module~#1}
1193   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1194     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1195     \prop_item:cn { c_stex_module_#1_prop } { content }
1196   }
1197 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page 19.)

```

1198 </package>

```


Chapter 21

sTeX -Module Inheritance Implementation

```
1199 <*package>
1200
1201 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1202
```

21.1 SMS Mode

```
1203 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1204 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1205 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1206 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1207
1208 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1209   \makeatletter
1210   \makeatother
1211   \ExplSyntaxOn
1212   \ExplSyntaxOff
1213 }
1214
1215 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1216   \symdef
1217   \importmodule
1218   \notation
1219   \symdecl
1220   \STEXexport
1221 }
1222
1223 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1224   \tl_to_str:n {
1225     module,
1226     @module
```

```

1227 }
1228 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 20.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF

```

```

1229 \bool_new:N \g__stex_smsmode_bool
1230 \bool_set_false:N \g__stex_smsmode_bool
1231 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1232   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1233 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 20.)

```

\__stex_smsmode_if_catcodes_p:

```

Checks whether the SMS mode category code scheme is active.

```

\__stex_smsmode_if_catcodes:TF

```

```

1234 \bool_new:N \g__stex_smsmode_catcode_bool
1235 \bool_set_false:N \g__stex_smsmode_catcode_bool
1236 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1237   \bool_if:NTF \g__stex_smsmode_catcode_bool
1238   \prg_return_true: \prg_return_false:
1239 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1240 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1241   \stex_if_smsmode:T {
1242     \__stex_smsmode_if_catcodes:F {
1243       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1244       \exp_after:wN \char_gset_active_eq:NN
1245       \c_backslash_str \__stex_smsmode_cs:
1246       \tex_global:D \char_set_catcode_active:N \
1247       \tex_global:D \char_set_catcode_other:N $
1248       \tex_global:D \char_set_catcode_other:N ^
1249       \tex_global:D \char_set_catcode_other:N _
1250       \tex_global:D \char_set_catcode_other:N &
1251       \tex_global:D \char_set_catcode_other:N ##
1252     }
1253   }
1254 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\stex_smsmode_set_codes:.` This function is documented on page 20.)

```

\__stex_smsmode_unset_codes:

```

Sets category code scheme back from the one used in SMS mode.

```

1255 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1256   \__stex_smsmode_if_catcodes:T {
1257     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1258     \exp_after:wN \tex_global:D \exp_after:wN
1259     \char_set_catcode_escape:N \c_backslash_str
1260     \tex_global:D \char_set_catcode_math_toggle:N $
1261     \tex_global:D \char_set_catcode_math_superscript:N ^
1262     \tex_global:D \char_set_catcode_math_subscript:N _
1263     \tex_global:D \char_set_catcode_alignment:N &
1264     \tex_global:D \char_set_catcode_parameter:N ##
1265   }
1266 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1267 \cs_new_protected:Nn \stex_in_smsmode:nn {
1268   \vbox_set:Nn \l_tmpa_box {
1269     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1270     \bool_gset_true:N \g__stex_smsmode_bool
1271     \stex_smsmode_set_codes:
1272     #2
1273     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1274     \stex_if_smsmode:F {
1275       \__stex_smsmode_unset_codes:
1276     }
1277   }
1278   \box_clear:N \l_tmpa_box
1279 }

```

(End definition for `\stex_in_smsmode:nn`. This function is documented on page 21.)

`_stex_smsmode_cs:` is executed on encountering `\` in `smsmode`. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```

1280 \cs_new_protected:Nn \_stex_smsmode_cs: {
1281   \str_clear:N \l_tmpa_str
1282   \peek_analysis_map_inline:n {
1283     % #1: token (one expansion)
1284     % #2: charcode
1285     % #3 catcode
1286     \token_if_eq_charcode:NNTF ##3 B {
1287       % token is a letter
1288       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1289     } {
1290       \str_if_empty:NTF \l_tmpa_str {
1291         % we don't allow (or need) single non-letter CSs
1292         % for now
1293         \peek_analysis_map_break:
1294       }{
1295         \str_if_eq:onTF \l_tmpa_str { begin } {
1296           \peek_analysis_map_break:n {
1297             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1298           }
1299         } {
1300           \str_if_eq:onTF \l_tmpa_str { end } {
1301             \peek_analysis_map_break:n {
1302               \exp_after:wN \_stex_smsmode_checkend:n ##1
1303             }
1304           } {
1305             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1306             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1307               \g_stex_smsmode_allowedmacros_tl
1308               { \use:c{\l_tmpa_str} } {
1309               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1310               \peek_analysis_map_break:n {
1311                 \exp_after:wN \l_tmpa_tl ##1
1312               }

```

```

1313     } {
1314         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1315         \g_stex_smsmode_allowedmacros_escape_tl
1316         { \use:c{\l_tmpa_str} } {
1317             \__stex_smsmode_unset_codes:
1318             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1319             % TODO \__stex_smsmode_rescan_cs:
1320             % \int_compare:nNnTF {##2} = {92} {
1321             %     \peek_analysis_map_break:n {
1322             %         \__stex_smsmode_unset_codes:
1323             %         \__stex_smsmode_rescan_cs:
1324             %     }
1325             % } {
1326             %     \peek_analysis_map_break:n {
1327             %         \exp_after:wN \l_tmpa_tl ##1
1328             %     }
1329             % }
1330         } {
1331             \int_compare:nNnTF {##2} = {92} {
1332                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1333             }{
1334                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1335             }
1336         }
1337     }
1338 }
1339 }
1340 }
1341 }
1342 }
1343 }

```

(End definition for __stex_smsmode_cs:.)

__stex_smsmode_rescan_cs: If the last token gobbled by \stex_smsmode_cs: happened to be a \, we need to rescan the cs name and reinsert it into the input stream:

```

1344 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1345     \str_clear:N \l_tmpb_str
1346     \peek_analysis_map_inline:n {
1347         \token_if_eq_charcode:NNTF ##3 B {
1348             % token is a letter
1349             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1350         } {
1351             \peek_analysis_map_break:n {
1352                 \exp_after:wN \use:c \exp_after:wN {
1353                     \exp_after:wN \l_tmpa_str\exp_after:wN
1354                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1355             }
1356         }
1357     }
1358 }

```

(End definition for __stex_smsmode_rescan_cs:.)

`__stex_smsmode_checkbegin:n` called on `\begin`; checks whether the environment being opened is allowed in SMS mode.

```

1359 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1360   \str_set:Nn \l_tmpa_str { #1 }
1361   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1362     \__stex_smsmode_unset_codes:
1363     \begin{#1}
1364   }
1365 }
```

(End definition for `__stex_smsmode_checkbegin:n`.)

`__stex_smsmode_checkend:n` called on `\end`; checks whether the environment being opened is allowed in SMS mode.

```

1366 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1367   \str_set:Nn \l_tmpa_str { #1 }
1368   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1369     \end{#1}
1370   }
1371 }
```

(End definition for `__stex_smsmode_checkend:n`.)

21.2 Inheritance

1372 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1373 \cs_new_protected:Nn \stex_import_module_uri:nn {
1374   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1375   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1376   \str_if_empty:NT \l__stex_importmodule_archive_str {
1377     \prop_if_empty:NF \l_stex_current_repository_prop {
1378       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1379     }
1380   }
1381
1382   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1383   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1384   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1385
1386   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1387     \stex_modules_current_namespace:
1388     \str_if_empty:NF \l__stex_importmodule_path_str {
1389       \str_set:Nx \l_stex_module_ns_str {
1390         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1391       }
1392     }
1393   }{
1394     \stex_require_repository:n \l__stex_importmodule_archive_str
1395     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1396     \l_stex_module_ns_str
1397     \str_if_empty:NF \l__stex_importmodule_path_str {
1398       \str_set:Nx \l_stex_module_ns_str {
1399         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1400       }
1401     }
```

```

1401     }
1402   }
1403 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 23.)

```

\l_stex_importmodule_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_importmodule_archive_str 1404 \str_new:N \l__stex_importmodule_name_str
\l_stex_importmodule_path_str 1405 \str_new:N \l__stex_importmodule_archive_str
\l_stex_importmodule_file_str 1406 \str_new:N \l__stex_importmodule_path_str
1407 \str_new:N \g__stex_importmodule_file_str

```

(End definition for `\l__stex_importmodule_name_str` and others.)

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1408 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1409   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1410
1411     % archive
1412     \str_set:Nx \l_tmpa_str { #2 }
1413     \str_if_empty:NTF \l_tmpa_str {
1414       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1415     } {
1416       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1417       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1418       \seq_put_right:Nn \l_tmpa_seq { source }
1419     }
1420
1421     % path
1422     \str_set:Nx \l_tmpb_str { #3 }
1423     \str_if_empty:NTF \l_tmpb_str {
1424       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1425
1426       \ltx@ifpackageloaded{babel} {
1427         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1428           { \language } \l_tmpb_str {
1429           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1430         }
1431       } {
1432         \str_clear:N \l_tmpb_str
1433       }
1434
1435       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1436       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1437         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1438       }{
1439         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1440         \IfFileExists{ \l_tmpa_str.tex }{
1441           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1442         }{
1443           % try english as default
1444           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1445           \IfFileExists{ \l_tmpa_str.en.tex }{
1446             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }

```

```

1447         }{
1448             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1449         }
1450     }
1451 }
1452
1453 } {
1454     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1455     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1456
1457     \ltx@ifpackageloaded{babel} {
1458         \exp_args:Nnx \prop_get:NnNF \c_stex_language_abbrevs_prop
1459             { \language } \l_tmpb_str {
1460             \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1461         }
1462     } {
1463         \str_clear:N \l_tmpb_str
1464     }
1465
1466     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1467
1468     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1469     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1470         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1471     }{
1472         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1473         \IfFileExists{ \l_tmpa_str/#4.tex }{
1474             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1475         }{
1476             % try english as default
1477             \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1478             \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1479                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1480             }{
1481                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1482                 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1483                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1484                 }{
1485                     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1486                     \IfFileExists{ \l_tmpa_str.tex }{
1487                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1488                     }{
1489                         % try english as default
1490                         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1491                         \IfFileExists{ \l_tmpa_str.en.tex }{
1492                             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1493                         }{
1494                             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1495                         }
1496                     }
1497                 }
1498             }
1499         }
1500     }

```

```

1501     }
1502
1503     \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1504     \seq_clear:N \g_stex_modules_in_file_seq
1505     % \exp_args:Nnx \use:nn {
1506         \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1507             \seq_clear:N \l_stex_all_modules_seq
1508             \prop_clear:N \l_stex_current_module_prop
1509             \str_set:Nx \l_tmpb_str { #2 }
1510             \str_if_empty:NF \l_tmpb_str {
1511                 \stex_set_current_repository:n { #2 }
1512             }
1513             \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1514             \input { \g__stex_importmodule_file_str }
1515         }
1516     % }{
1517
1518     % }
1519     \prop_gput:Noo \g_stex_module_files_prop
1520     \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1521     \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1522
1523     \stex_if_module_exists:nF { #1 ? #4 } {
1524         \msg_error:nnn{stex}{error/unknownmodule}{
1525             #1?#4~(in~file~\g__stex_importmodule_file_str)
1526         }
1527     }
1528 }
1529 \stex_activate_module:n { #1 ? #4 }
1530 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 23.)

`\importmodule`

```

1531 \NewDocumentCommand \importmodule { O{} m } {
1532     \stex_import_module_uri:nn { #1 } { #2 }
1533     \stex_debug:nn{modules}{Importing~module:~
1534         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1535     }
1536     \stex_if_smsmode:F {
1537         \stex_import_require_module:nnnn
1538         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1539         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1540         \stex_annotate_invisible:nnn
1541         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1542     }
1543     \exp_args:Nx \stex_add_to_current_module:n {
1544         \stex_import_require_module:nnnn
1545         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1546         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1547     }
1548     \exp_args:Nx \stex_add_import_to_current_module:n {
1549         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1550     }

```



```

1551 \stex_smsmode_set_codes:
1552 }
1553 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

(End definition for \importmodule. This function is documented on page 21.)

\usemodule

```

1554 \NewDocumentCommand \usemodule { 0{} m } {
1555   \stex_if_smsmode:F {
1556     \stex_import_module_uri:nn { #1 } { #2 }
1557     \stex_import_require_module:nnnn
1558     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1559     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1560     \stex_annotate_invisible:nnn
1561     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1562   }
1563   \stex_smsmode_set_codes:
1564 }

```

(End definition for \usemodule. This function is documented on page 22.)

```

1565 \endpackage

```

Chapter 22

STEX -Symbols Implementation

```
1566 <*package>
1567
1568 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1569
Warnings and error messages
1570
```

22.1 Symbol Declarations

```
1571 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1572 \seq_new:N \l_stex_all_symbols_seq

(End definition for \l_stex_all_symbols_seq. This variable is documented on page 25.)

\STEXsymbol
1573 \NewDocumentCommand \STEXsymbol { m } {
1574   \stex_get_symbol:n { #1 }
1575   \exp_args:No
1576   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1577 }

(End definition for \STEXsymbol. This function is documented on page 27.)
symdecl arguments:
1578 \keys_define:nn { stex / symdecl } {
1579   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1580   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1581   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1582   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1583   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1584   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1585   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1586   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1587 }
```

```

1588
1589 \bool_new:N \l_stex_symdecl_make_macro_bool
1590
1591 \cs_new_protected:Nn \__stex_symdecl_args:n {
1592   \str_clear:N \l_stex_symdecl_name_str
1593   \str_clear:N \l_stex_symdecl_args_str
1594   \bool_set_false:N \l_stex_symdecl_local_bool
1595   \tl_clear:N \l_stex_symdecl_type_tl
1596   \tl_clear:N \l_stex_symdecl_definiens_tl
1597
1598   \keys_set:nn { stex / symdecl } { #1 }
1599 }

```

\symdecl Parses the optional arguments and passes them on to `\stex_symdecl_do:` (so that `\symdef` can do the same)

```

1600
1601 \NewDocumentCommand \symdecl { s O{} m } {
1602   \__stex_symdecl_args:n { #2 }
1603   \IfBooleanTF #1 {
1604     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1605   } {
1606     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1607   }
1608   \stex_symdecl_do:n { #3 }
1609   \stex_smsmode_set_codes:
1610 }
1611 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for `\symdecl`. This function is documented on page 24.)

\stex_symdecl_do:n

```

1612 \cs_new_protected:Nn \stex_symdecl_do:n {
1613   \stex_if_in_module:F {
1614     % TODO throw error? some default namespace?
1615   }
1616
1617   \str_if_empty:NT \l_stex_symdecl_name_str {
1618     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1619   }
1620
1621   \prop_if_exist:cT { g_stex_symdecl_
1622     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1623     \prop_item:Nn \l_stex_current_module_prop {name} ?
1624     \l_stex_symdecl_name_str
1625     _prop
1626   }{
1627     % TODO throw error (beware of circular dependencies)
1628   }
1629
1630   \prop_clear:N \l_tmpa_prop
1631   \prop_put:Nnx \l_tmpa_prop { module } {
1632     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1633     \prop_item:Nn \l_stex_current_module_prop {name}
1634   }

```

```

1635 \seq_clear:N \l_tmpa_seq
1636 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1637 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1638 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1639 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1640
1641 \exp_args:No \stex_add_constant_to_current_module:n {
1642   \l_stex_symdecl_name_str
1643 }
1644
1645 % arity/args
1646 \int_zero:N \l_tmpb_int
1647
1648 \bool_set_true:N \l_tmpa_bool
1649 \str_map_inline:Nn \l_stex_symdecl_args_str {
1650   \token_case_meaning:NnF ##1 {
1651     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1652     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1653     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1654     {\tl_to_str:n a} {
1655       \bool_set_false:N \l_tmpa_bool
1656       \int_incr:N \l_tmpb_int
1657     }
1658     {\tl_to_str:n B} {
1659       \bool_set_false:N \l_tmpa_bool
1660       \int_incr:N \l_tmpb_int
1661     }
1662   }{
1663     \msg_set:nnn{stex}{error/wrongargs}{
1664       args~value~in~symbol~declaration~for~
1665       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1666       \prop_item:Nn \l_stex_current_module_prop {name} ?
1667       \l_stex_symdecl_name_str ~
1668       needs~to~be~
1669       i,~a,~b~or~B,~but~##1~given
1670     }
1671     \msg_error:nn{stex}{error/wrongargs}
1672   }
1673 }
1674 \bool_if:NTF \l_tmpa_bool {
1675   % possibly numeric
1676   \str_if_empty:NTF \l_stex_symdecl_args_str {
1677     \prop_put:Nnn \l_tmpa_prop { args } {}
1678     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1679   }{
1680     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1681     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1682     \str_clear:N \l_tmpa_str
1683     \int_step_inline:nn \l_tmpa_int {
1684       \str_put_right:Nn \l_tmpa_str i
1685     }
1686     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1687   }
1688 } {

```

```

1689 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1690 \prop_put:Nnx \l_tmpa_prop { arity }
1691 { \str_count:N \l_stex_symdecl_args_str }
1692 }
1693 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1694
1695
1696 % semantic macro
1697
1698 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1699   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1700     \prop_item:Nn \l_tmpa_prop { module } ?
1701     \prop_item:Nn \l_tmpa_prop { name }
1702   } }
1703
1704   \bool_if:NF \l_stex_symdecl_local_bool {
1705     \exp_args:Nx \stex_add_to_current_module:n {
1706       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1707         \prop_item:Nn \l_tmpa_prop { module } ?
1708         \prop_item:Nn \l_tmpa_prop { name }
1709       } }
1710     }
1711   }
1712 }
1713
1714 % add to all symbols
1715
1716 \bool_if:NF \l_stex_symdecl_local_bool {
1717   \exp_args:Nx \stex_add_to_current_module:n {
1718     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1719       \prop_item:Nn \l_tmpa_prop { module } ?
1720       \prop_item:Nn \l_tmpa_prop { name }
1721     }
1722   }
1723 }
1724
1725 \stex_debug:nn{symbols}{New~symbol:~
1726   \prop_item:Nn \l_tmpa_prop { module } ?
1727   \prop_item:Nn \l_tmpa_prop { name } ^^J
1728   Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1729   Args:~\prop_item:Nn \l_tmpa_prop { args }
1730 }
1731
1732 % circular dependencies require this:
1733
1734 \prop_if_exist:cF {
1735   g_stex_symdecl_
1736   \prop_item:Nn \l_tmpa_prop { module } ?
1737   \prop_item:Nn \l_tmpa_prop { name }
1738   _prop
1739 } {
1740   \prop_gset_eq:cN {
1741     g_stex_symdecl_
1742     \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1743     \prop_item:Nn \l_tmpa_prop { name }
1744     _prop
1745   } \l_tmpa_prop
1746 }
1747
1748 \stex_if_smsmode:TF {
1749   \bool_if:NF \l_stex_symdecl_local_bool {
1750     \exp_args:Nx \stex_add_to_sms:n {
1751       \prop_gset_from_keyval:cn {
1752         g_stex_symdecl_
1753         \prop_item:Nn \l_tmpa_prop { module } ?
1754         \prop_item:Nn \l_tmpa_prop { name }
1755         _prop
1756       } {
1757         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1758         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1759         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1760         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1761         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1762         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1763         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1764         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1765       }
1766       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1767         \prop_item:Nn \l_tmpa_prop { module } ?
1768         \prop_item:Nn \l_tmpa_prop { name }
1769       }
1770     }
1771   }
1772 }{
1773   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1774     \prop_item:Nn \l_tmpa_prop { module } ?
1775     \prop_item:Nn \l_tmpa_prop { name }
1776   }
1777   \stex_if_do_html:T {
1778     \stex_annotate_invisible:nnn {symdecl} {
1779       \prop_item:Nn \l_tmpa_prop { module } ?
1780       \prop_item:Nn \l_tmpa_prop { name }
1781     } {
1782       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1783       \stex_annotate_invisible:nnn{args}{}{
1784         \prop_item:Nn \l_tmpa_prop { args }
1785       }
1786       \stex_annotate_invisible:nnn{macroname}{}{#1}
1787       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1788         \stex_annotate_invisible:nnn{definiens}{}
1789         {\l_stex_symdecl_definiens_tl$}
1790       }
1791     }
1792   }
1793 }
1794 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 25.)

`\stex_get_symbol:n`

```
1795 \str_new:N \l_stex_get_symbol_uri_str
1796
1797 \cs_new_protected:Nn \stex_get_symbol:n {
1798   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1799     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1800   }{
1801     % argument is a string
1802     % is it a command name?
1803     \cs_if_exist:cTF { #1 }{
1804       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1805       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1806       \str_if_empty:NTF \l_tmpa_str {
1807         \exp_args:Nx \cs_if_eq:NNTF {
1808           \tl_head:N \l_tmpa_tl
1809         } \stex_invoke_symbol:n {
1810           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1811         }{
1812           \__stex_symdecl_get_symbol_from_string:n { #1 }
1813         }
1814       } {
1815         \__stex_symdecl_get_symbol_from_string:n { #1 }
1816       }
1817     }{
1818       % argument is not a command name
1819       \__stex_symdecl_get_symbol_from_string:n { #1 }
1820       % \l_stex_all_symbols_seq
1821     }
1822   }
1823 }
1824
1825 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1826   \str_set:Nn \l_tmpa_str { #1 }
1827   \bool_set_false:N \l_tmpa_bool
1828   \stex_if_in_module:T {
1829     \prop_get:NnN \l_stex_current_module_prop
1830     { constants } \l_tmpa_seq
1831     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1832       \bool_set_true:N \l_tmpa_bool
1833       \str_set:Nx \l_stex_get_symbol_uri_str {
1834         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1835         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1836       }
1837     }
1838   }
1839   \bool_if:NF \l_tmpa_bool {
1840     \tl_set:Nn \l_tmpa_tl {
1841       \msg_set:nnn{stex}{error/unknownsymbol}{
1842         No~symbol~#1~found!
1843       }
1844     }
1845     \msg_error:nn{stex}{error/unknownsymbol}
1846   }
1847   \str_set:Nn \l_tmpa_str { #1 }
1848   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1848 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1849   \str_set:Nn \l_tmpb_str { ##1 }
1850   \str_if_eq:eeT { \l_tmpa_str } {
1851     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1852   } {
1853     \seq_map_break:n {
1854       \tl_set:Nn \l_tmpa_tl {
1855         \str_set:Nn \l_stex_get_symbol_uri_str {
1856           ##1
1857         }
1858       }
1859     }
1860   }
1861 }
1862 \l_tmpa_tl
1863 }
1864 }
1865
1866 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1867   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1868   { \tl_tail:N \l_tmpa_tl }
1869   \tl_if_single:NTF \l_tmpa_tl {
1870     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1871       \exp_after:wN \str_set:Nn \exp_after:wN
1872       \l_stex_get_symbol_uri_str \l_tmpa_tl
1873     }{
1874       % TODO
1875       % tail is not a single group
1876     }
1877   }{
1878     % TODO
1879     % tail is not a single group
1880   }
1881 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page [25](#).)

22.2 Notations

```

1882 <@@=stex_notation>
1883 notation arguments:
1884 \keys_define:nn { stex / notation } {
1885   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1886   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1887   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1888   op .tl_set:N = \l__stex_notation_op_tl ,
1889   unknown .code:n = \str_set:Nx
1890     \l__stex_notation_variant_str \l_keys_key_str
1891 }
1892
1893 \cs_new_protected:Nn \__stex_notation_args:n {
1894   \str_clear:N \l__stex_notation_lang_str
1895   \str_clear:N \l__stex_notation_variant_str

```



```

1895 \str_clear:N \l__stex_notation_prec_str
1896 \tl_clear:N \l__stex_notation_op_tl
1897
1898 \keys_set:nn { stex / notation } { #1 }
1899 }

```

\notation

```

1900 \NewDocumentCommand \notation { 0{ } m } {
1901   \__stex_notation_args:n { #1 }
1902   \tl_clear:N \l_stex_symdecl_definiens_tl
1903   \stex_get_symbol:n { #2 }
1904   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1905 }
1906 \stex_deactivate_macro:Nn \notation {module~environments}

```

(End definition for \notation. This function is documented on page 25.)

\stex_notation_do:nn

```

1907 \cs_new_protected:Nn \stex_notation_do:nn {
1908   \prop_set_eq:Nc \l_tmpa_prop {
1909     g_stex_symdecl_ #1 _prop
1910   }
1911
1912   \prop_clear:N \l_tmpb_prop
1913   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1914   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1915   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1916
1917   % precedences
1918   \seq_clear:N \l_tmpb_seq
1919   \exp_args:NNno
1920   \str_if_empty:NTF \l__stex_notation_prec_str {
1921     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1922     \int_compare:nNnTF \l_tmpa_str = 0 {
1923       \exp_args:NNnx
1924       \prop_put:Nno \l_tmpb_prop { opprec }
1925       { \neginfprec }
1926     }{
1927       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1928     }
1929   } {
1930     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1931       \exp_args:NNnx
1932       \prop_put:Nno \l_tmpb_prop { opprec }
1933       { \neginfprec }
1934       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1935       \int_step_inline:nn { \l_tmpa_str } {
1936         \exp_args:NNx
1937         \seq_put_right:Nn \l_tmpb_seq { \infprec }
1938       }
1939     }{
1940       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1941       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1942         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1943         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

1944         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1945         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
1946         \seq_map_inline:Nn \l_tmpa_seq {
1947             \seq_put_right:Nn \l_tmpb_seq { ##1 }
1948         }
1949     }
1950     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1951 }{
1952     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1953     \int_compare:nNnTF \l_tmpa_str = 0 {
1954         \exp_args:NNnx
1955         \prop_put:Nno \l_tmpb_prop { opprec }
1956         { \infprec }
1957     }{
1958         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1959     }
1960 }
1961 }
1962 }
1963
1964 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1965 \int_step_inline:nn { \l_tmpa_str } {
1966     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
1967         \exp_args:NNx
1968         \seq_put_right:Nn \l_tmpb_seq {
1969             \prop_item:Nn \l_tmpb_prop { opprec }
1970         }
1971     }
1972 }
1973
1974 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1975 \tl_clear:N \l_tmpa_tl
1976
1977 \int_compare:nNnTF \l_tmpa_str = 0 {
1978     \exp_args:NNe
1979     \cs_set:Npn \l__stex_notation_macrocode_cs {
1980         \_stex_term_math_oms:nnnn { #1 }
1981         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1982         { \prop_item:Nn \l_tmpb_prop { opprec } }
1983         { \exp_not:n { #2 } }
1984     }
1985     \__stex_notation_final:
1986 }{
1987     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1988     \str_if_in:NnTF \l_tmpb_str b {
1989         \exp_args:Nne \use:nn
1990         {
1991             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1992             \cs_set:Npn \l_tmpa_str { {
1993                 \_stex_term_math_omb:nnnn { #1 }
1994                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1995                 { \prop_item:Nn \l_tmpb_prop { opprec } }
1996                 { \exp_not:n { #2 } }
1997             }}

```

```

1998   }{
1999     \str_if_in:NnTF \l_tmpb_str B {
2000       \exp_args:Nne \use:nn
2001       {
2002         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2003         \cs_set:Npn \l_tmpa_str } { {
2004           \_stex_term_math_omb:nnnn { #1 }
2005           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2006           { \prop_item:Nn \l_tmpb_prop { opprec } }
2007           { \exp_not:n { #2 } }
2008         } }
2009       }{
2010         \exp_args:Nne \use:nn
2011         {
2012           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2013           \cs_set:Npn \l_tmpa_str } { {
2014             \_stex_term_math_oma:nnnn { #1 }
2015             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2016             { \prop_item:Nn \l_tmpb_prop { opprec } }
2017             { \exp_not:n { #2 } }
2018           } }
2019         }
2020       }
2021
2022       \int_zero:N \l_tmpa_int
2023       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2024       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2025       \__stex_notation_arguments:
2026     }
2027   }

```

(End definition for `\stex_notation_do:nn`. This function is documented on page 26.)

`__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```

2028 \cs_new_protected:Nn \__stex_notation_arguments: {
2029   \int_incr:N \l_tmpa_int
2030   \str_if_empty:NnTF \l_tmpa_str {
2031     \__stex_notation_final:
2032   }{
2033     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2034     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2035     \str_if_eq:NnTF \l_tmpb_str a {
2036       \__stex_notation_argument_assoc:n
2037     }{
2038       \str_if_eq:NnTF \l_tmpb_str B {
2039         \__stex_notation_argument_assoc:n
2040       }{
2041         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2042         \tl_put_right:Nx \l_tmpa_tl {
2043           { \_stex_term_math_arg:nnn
2044             { \int_use:N \l_tmpa_int }
2045             { \l_tmpb_str }
2046             { ####\int_use:N \l_tmpa_int }
2047           }

```

```

2048     }
2049     \__stex_notation_arguments:
2050   }
2051 }
2052 }
2053 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2054 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2055   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2056   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2057   \tl_put_right:Nx \l_tmpa_tl {
2058     { \stex_term_math_assoc_arg:nnnn
2059       { \int_use:N \l_tmpa_int }
2060       { \l_tmpb_str }
2061       \exp_args:No \exp_not:n
2062       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2063       { ####\int_use:N \l_tmpa_int }
2064     }
2065   }
2066   \__stex_notation_arguments:
2067 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2068 \cs_new_protected:Nn \__stex_notation_final: {
2069   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2070   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2071   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2072   \exp_args:Nne \use:nn
2073   {
2074     \cs_generate_from_arg_count:cNnn {
2075       stex_notation_ \l_tmpa_str \c_hash_str
2076       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2077       _cs
2078     }
2079     \cs_gset:Npn \l_tmpb_str } { {
2080       \exp_after:wN \exp_after:wN \exp_after:wN
2081       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2082       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2083     } }
2084
2085   \tl_if_empty:NF \l__stex_notation_op_tl {
2086     \cs_gset:cpx {
2087       stex_op_notation_ \l_tmpa_str \c_hash_str
2088       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2089       _cs
2090     } {
2091       \stex_term_oms:nnn {
2092         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2093         \l__stex_notation_lang_str

```

```

2094     }{
2095         \l_tmpa_str
2096     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2097 }
2098 }
2099
2100
2101
2102 \stex_debug:nn{symbols}{
2103     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2104     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2105     Operator~precedence:~
2106     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2107     Argument~precedences:~
2108     \seq_use:Nn \l_tmpa_seq {,~}^^J
2109     Notation: \cs_meaning:c {
2110         stex_notation_ \l_tmpa_str \c_hash_str
2111         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2112         _cs
2113     }
2114 }
2115
2116 \prop_gset_eq:cN {
2117     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2118     \c_hash_str \l__stex_notation_lang_str _prop
2119 } \l_tmpb_prop
2120
2121 \exp_args:Nx
2122 \stex_add_to_current_module:n {
2123     \prop_get:cnN {
2124         g_stex_symdecl_
2125         \prop_item:Nn \l_tmpb_prop { symbol }
2126         _prop
2127     } { notations } \exp_not:N \l_tmpa_seq
2128     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2129         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2130     }
2131     \prop_put:cno {
2132         g_stex_symdecl_
2133         \prop_item:Nn \l_tmpb_prop { symbol }
2134         _prop
2135     } { notations } \exp_not:N \l_tmpa_seq
2136 }
2137
2138 \stex_if_smsmode:TF {
2139     \stex_smsmode_set_codes:
2140     \exp_args:Nx \stex_add_to_sms:n {
2141         \prop_gset_from_keyval:cn {
2142             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2143             \c_hash_str \l__stex_notation_lang_str _prop
2144         } {
2145             symbol = \prop_item:Nn \l_tmpb_prop { symbol } ,
2146             language = \prop_item:Nn \l_tmpb_prop { language } ,
2147             variant = \prop_item:Nn \l_tmpb_prop { variant } ,

```

```

2148         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2149         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2150     }
2151 }
2152 }{
2153   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2154   \seq_put_right:Nx \l_tmpa_seq {
2155     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2156   }
2157   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2158   \prop_set_eq:cN {
2159     g_stex_symdecl_ \l_tmpa_str _prop
2160   } \l_tmpa_prop
2161
2162   % HTML annotations
2163   \stex_if_do_html:T {
2164     \stex_annotate_invisible:nnn { notation }
2165     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2166       \stex_annotate_invisible:nnn { notationfragment }
2167       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2168     }
2169     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2170     \stex_annotate_invisible:nnn { precedence }
2171     { \prop_item:Nn \l_tmpb_prop { opprec } ;
2172       \seq_use:Nn \l_tmpa_seq { x }
2173     }{}
2174
2175     \int_zero:N \l_tmpa_int
2176     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2177     \tl_clear:N \l_tmpa_tl
2178     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2179       \int_incr:N \l_tmpa_int
2180       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2181       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2182       \str_if_eq:VnTF \l_tmpb_str a {
2183         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2184           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2185           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2186         } }
2187       }{
2188         \str_if_eq:VnTF \l_tmpb_str B {
2189           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2190             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2191             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2192           } }
2193         }{
2194           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2195             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2196           } }
2197         }
2198       }
2199     }
2200     \stex_annotate_invisible:nnn { notationcomp }{}{
2201       $ \exp_args:Nno \use:nn { \use:c {
2202         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2202         \c_hash_str \l__stex_notation_variant_str
2203         \c_hash_str \l__stex_notation_lang_str _cs
2204     } } { \l_tmpa_tl } $
2205   }
2206 }
2207 }
2208 }
2209 }

```

(End definition for _stex_notation_final:.)

\symdef

```

2210 \keys_define:nn { stex / symdef } {
2211   name .str_set_x:N = \l_stex_symdecl_name_str ,
2212   local .bool_set:N = \l_stex_symdecl_local_bool ,
2213   args .str_set_x:N = \l_stex_symdecl_args_str ,
2214   type .tl_set:N = \l_stex_symdecl_type_tl ,
2215   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2216   op .tl_set:N = \l__stex_notation_op_tl ,
2217   lang .str_set_x:N = \l__stex_notation_lang_str ,
2218   variant .str_set_x:N = \l__stex_notation_variant_str ,
2219   prec .str_set_x:N = \l__stex_notation_prec_str ,
2220   unknown .code:n = \str_set:Nx
2221     \l__stex_notation_variant_str \l_keys_key_str
2222 }
2223
2224 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2225   \str_clear:N \l_stex_symdecl_name_str
2226   \str_clear:N \l_stex_symdecl_args_str
2227   \bool_set_false:N \l_stex_symdecl_local_bool
2228   \tl_clear:N \l_stex_symdecl_type_tl
2229   \tl_clear:N \l_stex_symdecl_definiens_tl
2230   \str_clear:N \l__stex_notation_lang_str
2231   \str_clear:N \l__stex_notation_variant_str
2232   \str_clear:N \l__stex_notation_prec_str
2233   \tl_clear:N \l__stex_notation_op_tl
2234
2235   \keys_set:nn { stex / symdef } { #1 }
2236 }
2237
2238 \NewDocumentCommand \symdef { 0{} m } {
2239   \_stex_notation_symdef_args:n { #1 }
2240   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2241   \stex_symdecl_do:n { #2 }
2242   \exp_args:Nx \stex_notation_do:nn {
2243     \prop_item:Nn \l_tmpa_prop { module } ?
2244     \prop_item:Nn \l_tmpa_prop { name }
2245   }
2246 }
2247 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for \symdef. This function is documented on page 26.)

```

2248 \</package>

```

Chapter 23

STEX -Terms Implementation

```
2249 <*package>
2250
2251 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2252
2253 <@@=stex_terms>
2254
2255 Warnings and error messages
2256 \msg_new:nnn{stex}{error/nonotation}{
2257   Symbol~#1~invoked,~but~has~no~notation#2!
2258 }
2259 \msg_new:nnn{stex}{error/notationarg}{
2260   Error~in~parsing~notation~#1
2261 }
```

23.1 Symbol Invocations

Arguments:

```
2261 \keys_define:nn { stex / terms } {
2262   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2263   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2264   unknown .code:n = \str_set:Nx
2265     \l__stex_terms_variant_str \l_keys_key_str
2266 }
2267
2268 \cs_new_protected:Nn \__stex_terms_args:n {
2269   \str_clear:N \l__stex_terms_lang_str
2270   \str_clear:N \l__stex_terms_variant_str
2271   \str_clear:N \l__stex_terms_prec_str
2272   \tl_clear:N \l__stex_terms_op_tl
2273
2274   \keys_set:nn { stex / terms } { #1 }
2275 }
```

`\stex_invoke_symbol:n` Invokes a semantic macro


```

2276 \cs_new_protected:Nn \stex_invoke_symbol:n {
2277   \if_mode_math:
2278     \exp_after:wN \__stex_terms_invoke_math:n
2279   \else:
2280     \exp_after:wN \__stex_terms_invoke_text:n
2281   \fi: { #1 }
2282 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 27.)

`__stex_terms_invoke_math:n`

```

2283 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2284   \peek_charcode_remove:NTF ! {
2285     \peek_charcode:NTF [ {
2286       \__stex_terms_invoke_op:nw { #1 }
2287     }{
2288       \__stex_terms_invoke_op:nw { #1 } []
2289     }
2290   }{
2291     \peek_charcode_remove:NTF * {
2292       \__stex_terms_invoke_text:n { #1 }
2293     }{
2294       \peek_charcode:NTF [ {
2295         \__stex_terms_invoke_math:nw { #1 }
2296       }{
2297         \__stex_terms_invoke_math:nw { #1 } []
2298       }
2299     }
2300   }
2301 }

```

(End definition for `__stex_terms_invoke_math:n`.)

`__stex_terms_invoke_op:nw`

```

2302 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2303   \__stex_terms_args:n { #2 }
2304   \cs_if_exist:cTF {
2305     stex_op_notation_ #1 \c_hash_str
2306     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2307   }{
2308     \csname stex_op_notation_ #1 \c_hash_str
2309     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2310   \endcsname
2311   }{
2312     % TODO throw error
2313   }
2314 }

```

(End definition for `__stex_terms_invoke_op:nw`.)

`__stex_terms_invoke_math:nw`

```

2315 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2316   \__stex_terms_args:n { #2 }
2317   \prop_set_eq:Nc \l_tmpa_prop {
2318     g_stex_symdecl_ #1 _prop

```

```

2319 }
2320 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2321 \seq_if_empty:NTF \l_tmpa_seq {
2322   \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2323 } {
2324   \seq_if_in:NxTF \l_tmpa_seq
2325     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2326     \use:c{
2327       stex_notation_ #1 \c_hash_str
2328       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2329       _cs
2330     }
2331   }{
2332     \str_if_empty:NTF \l__stex_terms_variant_str {
2333       \str_if_empty:NTF \l__stex_terms_lang_str {
2334         \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2335         \use:c{
2336           stex_notation_ #1 \c_hash_str \l_tmpa_str
2337           _cs
2338         }
2339       }{
2340         \msg_error:nn{stex}{error/nonotation}{#1}{
2341           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2342         }
2343       }
2344     }{
2345       \msg_error:nn{stex}{error/nonotation}{#1}{
2346         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2347       }
2348     }
2349   }
2350 }
2351 }

```

(End definition for `__stex_terms_invoke_math:nw`.)

`__stex_terms_invoke_text:n`

```

2352 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2353   \peek_charcode_remove:NTF ! {
2354     \stex_term_custom:nn { #1 } { }
2355   }{
2356     \prop_set_eq:Nc \l_tmpa_prop {
2357       g_stex_symdecl_ #1 _prop
2358     }
2359     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2360     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2361   }
2362 }

```

(End definition for `__stex_terms_invoke_text:n`.)

23.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2363 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2364 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2365 \int_new:N \l__stex_terms_downprec
2366 \int_set_eq:NN \l__stex_terms_downprec \infprec

(End definition for \infprec, \neginfprec, and \l__stex_terms_downprec. These variables are docu-
mented on page 28.)

```

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
2367 \tl_set:Nn \l__stex_terms_left_bracket_str (
2368 \tl_set:Nn \l__stex_terms_right_bracket_str )

(End definition for \l__stex_terms_left_bracket_str and \l__stex_terms_right_bracket_str.)

```

_stex_terms_maybe_brackets:nn Compares precedences and insert brackets accordingly

```

2369 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2370 \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2371 \bool_if:NNTF \l_stex_inarray_bool { #2 }{
2372 \dobrackets { #2 }
2373 }
2374 }{ #2 }
2375 }

(End definition for \_stex_terms_maybe_brackets:nn.)

```

\dobrackets

```

2376 %\RequirePackage{scalerel}
2377 \cs_new_protected:Npn \dobrackets #1 {
2378 %\ThisStyle{\if D\m@switch
2379 % \exp_args:Nnx \use:nn
2380 % { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2381 % { \exp_not:N\right\l__stex_terms_right_bracket_str }
2382 % \else
2383 \exp_args:Nnx \use:nn
2384 { \l__stex_terms_left_bracket_str #1 }
2385 { \l__stex_terms_right_bracket_str }
2386 %\fi}
2387 }

(End definition for \dobrackets. This function is documented on page 28.)

```

\withbrackets

```

2388 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2389 \exp_args:Nnx \use:nn
2390 {
2391 \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2392 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2393 #3
2394 }
2395 {
2396 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2397 {\l__stex_terms_left_bracket_str}
2398 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str

```

```

2399     {\l__stex_terms_right_bracket_str}
2400   }
2401 }

```

(End definition for `\withbrackets`. This function is documented on page 28.)

`\STEXinvisible`

```

2402 \cs_new_protected:Npn \STEXinvisible #1 {
2403   \stex_annotate_invisible:n { #1 }
2404 }

```

(End definition for `\STEXinvisible`. This function is documented on page 29.)

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2405 \cs_new_protected:Nn \_stex_term_oms:nnn {
2406   \stex_annotate:nnn{ OMID }{ #2 }{
2407     \stex_highlight_term:nn { #1 } { #3 }
2408   }
2409 }
2410
2411 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2412   \__stex_terms_maybe_brackets:nn { #3 }{
2413     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2414   }
2415 }

```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 27.)

`_stex_term_math_oma:nnnn`

```

2416 \cs_new_protected:Nn \_stex_term_oma:nnn {
2417   \stex_annotate:nnn{ OMA }{ #2 }{
2418     \stex_highlight_term:nn { #1 } { #3 }
2419   }
2420 }
2421
2422 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2423   \__stex_terms_maybe_brackets:nn { #3 }{
2424     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2425   }
2426 }

```

(End definition for `_stex_term_math_oma:nnnn`. This function is documented on page 27.)

`_stex_term_math_omb:nnnn`

```

2427 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2428   \stex_annotate:nnn{ OMBIND }{ #2 }{
2429     \stex_highlight_term:nn { #1 } { #3 }
2430   }
2431 }
2432
2433 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2434   \__stex_terms_maybe_brackets:nn { #3 }{
2435     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2436   }
2437 }

```

(End definition for `_stex_term_math_omb:nnnn`. This function is documented on page 27.)

`_stex_term_math_arg:nnn`

```

2438 \cs_new_protected:Nn \_stex_term_arg:nn {
2439   \stex_unhighlight_term:n {
2440     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2441   }
2442 }
2443 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2444   \exp_args:Nnx \use:nn
2445     { \int_set:Nn \l__stex_terms_downprec { #2 }
2446       \_stex_term_arg:nn { #1 }{ #3 }
2447     }
2448   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2449 }

```

(End definition for `_stex_term_math_arg:nnn`. This function is documented on page 27.)

`_stex_term_math_assoc_arg:nnnn`

```

2450 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2451   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2452   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2453     \tl_set:Nn \l_tmpa_tl { #4 }
2454   }{
2455     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2456     \seq_reverse:N \l_tmpa_seq
2457     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2458     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2459
2460     \seq_map_inline:Nn \l_tmpa_seq {
2461       \exp_args:NNo \tl_set:No \l_tmpa_tl {
2462         \exp_args:Nno
2463           \l_tmpa_cs { ##1 } \l_tmpa_tl
2464       }
2465     }
2466   }
2467 }
2468 \exp_args:Nnno
2469 \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2470 }

```

(End definition for `_stex_term_math_assoc_arg:nnnn`. This function is documented on page 27.)

`\stex_term_custom:nn`

```

2471 \cs_new_protected:Nn \stex_term_custom:nn {
2472   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2473   \str_set:Nn \l_tmpa_str { #2 }
2474   \tl_clear:N \l_tmpa_tl
2475   \int_zero:N \l_tmpa_int
2476   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2477   \__stex_terms_custom_loop:
2478 }

```

(End definition for `\stex_term_custom:nn`. This function is documented on page 29.)

_stex_terms_custom_loop:

```

2479 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2480   \bool_set_false:N \l_tmpa_bool
2481   \bool_while_do:nn {
2482     \str_if_eq_p:ee X {
2483       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2484     }
2485   }{
2486     \int_incr:N \l_tmpa_int
2487   }
2488
2489   \peek_charcode:NTF [ {
2490     % notation/text component
2491     \_stex_terms_custom_component:w
2492   } {
2493     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2494       % all arguments read => finish
2495       \_stex_terms_custom_final:
2496     } {
2497       % arguments missing
2498       \peek_charcode_remove:NTF * {
2499         % invisible, specific argument position or both
2500         \peek_charcode:NTF [ {
2501           % visible specific argument position
2502           \_stex_terms_custom_arg:wn
2503         } {
2504           % invisible
2505           \peek_charcode_remove:NTF * {
2506             % invisible specific argument position
2507             \_stex_terms_custom_arg_inv:wn
2508           } {
2509             % invisible next argument
2510             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2511           }
2512         }
2513       } {
2514         % next normal argument
2515         \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2516       }
2517     }
2518   }
2519 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```

2520 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2521   \bool_set_true:N \l_tmpa_bool
2522   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2523 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

2524 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2525   \str_set:Nx \l_tmpb_str {
2526     \str_item:Nn \l_tmpa_str { #1 }
2527   }
2528   \str_case:VnTF \l_tmpb_str {
2529     { X } {
2530       \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2531     }
2532     { i } { \__stex_terms_custom_set_X:n { #1 } }
2533     { b } { \__stex_terms_custom_set_X:n { #1 } }
2534     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2535     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2536   }{}{
2537     \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2538   }
2539
2540   \bool_if:nTF \l_tmpa_bool {
2541     \tl_put_right:Nx \l_tmpa_tl {
2542       \stex_annotate_invisible:n {
2543         \stex_term_arg:nn { \int_eval:n { #1 } }
2544         \exp_not:n { { #2 } }
2545       }
2546     }
2547   } {
2548     \tl_put_right:Nx \l_tmpa_tl {
2549       \stex_term_arg:nn { \int_eval:n { #1 } }
2550       \exp_not:n { { #2 } }
2551     }
2552   }
2553
2554   \__stex_terms_custom_loop:
2555 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

2556 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2557   \str_set:Nx \l_tmpa_str {
2558     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2559     X
2560     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2561   }
2562 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

2563 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2564   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2565   \__stex_terms_custom_loop:
2566 }

```

(End definition for __stex_terms_custom_component:.)

`_stex_terms_custom_final:`

```

2567 \cs_new_protected:Nn \_stex_terms_custom_final: {
2568   \int_compare:nNnTF \l_tmpb_int = 0 {
2569     \exp_args:Nnno \_stex_term_oms:nnn
2570   }{
2571     \str_if_in:NnTF \l_tmpa_str {b} {
2572       \exp_args:Nnno \_stex_term_ombind:nnn
2573     } {
2574       \exp_args:Nnno \_stex_term_oma:nnn
2575     }
2576   }
2577   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2578 }

```

(End definition for _stex_terms_custom_final:.)

`\symref`

`\symname`

```

2579 \NewDocumentCommand \symref { m m }{
2580   \let\compemph_uri_prev:\compemph@uri
2581   \let\compemph@uri\symrefemph@uri
2582   \STEXsymbol{#1}![#2]
2583   \let\compemph@uri\compemph_uri_prev:
2584 }
2585
2586 \keys_define:nn { stex / symname } {
2587   post      .str_set_x:N      = \l_stex_symname_post_str
2588 }
2589
2590 \cs_new_protected:Nn \stex_symname_args:n {
2591   \str_clear:N \l_stex_symname_post_str
2592   \keys_set:nn { stex / symname } { #1 }
2593 }
2594
2595 \NewDocumentCommand \symname { 0{} m }{
2596   \stex_symname_args:n { #1 }
2597   \stex_get_symbol:n { #2 }
2598   \str_set:Nx \l_tmpa_str {
2599     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2600   }
2601   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2602
2603   \let\compemph_uri_prev:\compemph@uri
2604   \let\compemph@uri\symrefemph@uri
2605   \exp_args:NNx \use:nn
2606   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2607     \l_tmpa_str \l_stex_symname_post_str
2608   ] }
2609   \let\compemph@uri\compemph_uri_prev:
2610 }

```

(End definition for \symref and \symname. These functions are documented on page 27.)

23.3 Notation Components

2611 $\langle @@=\text{stex_notationcomps} \rangle$

$\backslash\text{stex_highlight_term:nn}$

```

2612
2613 \str_new:N \l__stex_notationcomps_highlight_uri_str
2614 \cs_new_protected:Nn \stex_highlight_term:nn {
2615   \exp_args:Nnx
2616   \use:nn {
2617     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2618     #2
2619   } {
2620     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2621     { \l__stex_notationcomps_highlight_uri_str }
2622   }
2623 }
2624
2625 \cs_new_protected:Nn \stex_unhighlight_term:n {
2626   % \latexml_if:TF {
2627   %   #1
2628   % } {
2629   %   \scalatex_if:TF {
2630   %     #1
2631   %   } {
2632     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2633   % }
2634   % }
2635 }
```

(End definition for $\backslash\text{stex_highlight_term:nn}$. This function is documented on page 29.)

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
2636 \cs_new_protected:Npn \comp #1 {
2637   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2638     \scalatex_if:TF {
2639       \stex_annotate:nnn { comp } { \l__stex_notationcomps_highlight_uri_str } { #1 }
2640     } {
2641       \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2642     }
2643   }
2644 }
2645
2646 \cs_new_protected:Npn \compemph@uri #1 #2 {
2647   \compemph{ #1 }
2648 }
2649
2650
2651 \cs_new_protected:Npn \compemph #1 {
2652   \textcolor{blue}{#1}
2653 }
2654
2655 \cs_new_protected:Npn \defemph@uri #1 #2 {
2656   \defemph{#1}
2657 }
```

```

2658
2659 \cs_new_protected:Npn \defemph #1 {
2660     \textbf{#1}
2661 }
2662
2663 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2664     \symrefemph{#1}
2665 }
2666
2667 \cs_new_protected:Npn \symrefemph #1 {
2668     \textbf{#1}
2669 }

```

(End definition for `\comp` and others. These functions are documented on page 29.)

`\ellipses`

```

2670 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 29.)

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
2671 \bool_new:N \l_stex_inparray_bool
2672 \bool_set_false:N \l_stex_inparray_bool
2673 \NewDocumentCommand \parray { m m } {
2674     \begingroup
2675     \bool_set_true:N \l_stex_inparray_bool
2676     \begin{array}{#1}
2677         #2
2678     \end{array}
2679 \endgroup
2680 }
2681
2682 \NewDocumentCommand \prmatrix { m } {
2683     \begingroup
2684     \bool_set_true:N \l_stex_inparray_bool
2685     \begin{matrix}
2686         #1
2687     \end{matrix}
2688 \endgroup
2689 }
2690
2691 \def \parrayline #1 #2 {
2692     #1 #2 \bool_if:NT \l_stex_inparray_bool {\}
2693 }
2694
2695 \def \parraylineh #1 #2 {
2696     #1 #2 \bool_if:NT \l_stex_inparray_bool {\hline}
2697 }
2698
2699 \def \parraycell #1 {
2700     #1 \bool_if:NT \l_stex_inparray_bool {\&}
2701 }

```

(End definition for `\parray` and others. These functions are documented on page ??.)

```

2702 \endpackage

```

Chapter 24

STEX -Structural Features Implementation

```
2703 <*package>
2704
2705 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2706
2707 <@@=stex_features>
      Warnings and error messages
2708
```

24.1 The feature environment

structural@feature

```
2709
2710 \NewDocumentEnvironment{structural@feature}{ m m m }{
2711   \stex_if_in_module:F {
2712     \msg_set:nnn{stex}{error/nomodule}{
2713       Structural~Feature~has~to~occur~in~a~module:\\
2714       Feature~#2~of~type~#1\\
2715       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2716     }
2717     \msg_error:nn{stex}{error/nomodule}
2718   }
2719
2720   \str_set:Nx \l_stex_module_name_str {
2721     \prop_item:Nn \l_stex_current_module_prop
2722       { name } / #2 - feature
2723   }
2724
2725   \str_set:Nx \l_stex_module_ns_str {
2726     \prop_item:Nn \l_stex_current_module_prop
2727       { ns }
2728   }
2729
```

```

2730
2731 \str_clear:N \l_tmpa_str
2732 \seq_clear:N \l_tmpa_seq
2733 \tl_clear:N \l_tmpa_tl
2734 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2735     origname = #2,
2736     name      = \l_stex_module_name_str ,
2737     ns        = \l_stex_module_ns_str ,
2738     imports   = \exp_not:o { \l_tmpa_seq } ,
2739     constants = \exp_not:o { \l_tmpa_seq } ,
2740     content   = \exp_not:o { \l_tmpa_tl } ,
2741     file      = \exp_not:o { \g_stex_currentfile_seq } ,
2742     lang      = \l_stex_module_lang_str ,
2743     sig       = \l_tmpa_str ,
2744     meta      = \l_tmpa_str ,
2745     feature   = #1 ,
2746 }
2747
2748 \stex_if_smsmode:TF {
2749     \stex_smsmode_set_codes:
2750 } {
2751     \begin{stex_annotate_env}{ feature:#1 }{}
2752     \stex_annotate_invisible:nnn{header}{}{ #3 }
2753 }
2754 }{
2755     \str_set:Nx \l_tmpa_str {
2756         c_stex_feature_
2757         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2758         \prop_item:Nn \l_stex_current_module_prop { name }
2759         _prop
2760     }
2761     \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2762     \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2763     \stex_if_smsmode:TF {
2764         \exp_args:Nx \stex_add_to_sms:n {
2765             \prop_gset_from_keyval:cn {
2766                 c_stex_feature_
2767                 \prop_item:Nn \l_stex_current_module_prop { ns } ?
2768                 \prop_item:Nn \l_stex_current_module_prop { name }
2769                 _prop
2770             } {
2771                 origname = #2,
2772                 name      = \prop_item:cn { \l_tmpa_str } { name } ,
2773                 ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2774                 imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2775                 constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2776                 content   = \prop_item:cn { \l_tmpa_str } { content } ,
2777                 file      = \prop_item:cn { \l_tmpa_str } { file } ,
2778                 lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
2779                 sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
2780                 meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2781                 feature   = \prop_item:cn { \l_tmpa_str } { feature }
2782             }
2783         }

```

```

2784 } {
2785     \end{stex_annotate_env}
2786 }
2787 }
2788

```

24.2 Features

structure

```

2789
2790 \prop_new:N \l_stex_all_structures_prop
2791
2792 \keys_define:nn { stex / features / structure } {
2793     name .str_set_x:N = \l__stex_features_structure_name_str ,
2794 }
2795
2796 \cs_new_protected:Nn \__stex_features_structure_args:n {
2797     \str_clear:N \l__stex_features_structure_name_str
2798     \keys_set:nn { stex / features / structure } { #1 }
2799 }
2800
2801 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2802 % \__stex_features_structure_args:n { ##1 }
2803 % \str_if_empty:NT \l__stex_features_structure_name_str {
2804 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2805 % }
2806 %} {
2807 %
2808 %}
2809
2810 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2811     \__stex_features_structure_args:n { #1 }
2812     \str_if_empty:NT \l__stex_features_structure_name_str {
2813         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2814     }
2815     \exp_args:Nnnx
2816     \begin{structural@feature}{ structure }
2817         { \l__stex_features_structure_name_str }{}
2818         \seq_clear:N \l_tmpa_seq
2819         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2820
2821     }{
2822         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2823         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2824         \str_set:Nx \l_tmpa_str {
2825             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2826             \prop_item:Nn \l_stex_current_module_prop { name }
2827         }
2828         \seq_map_inline:Nn \l_tmpa_seq {
2829             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2830         }
2831         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2832         \exp_args:Nnx

```

```

2833 \AddToHookNext { env / mathstructure / after }{
2834 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2835 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2836 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2837 \STEXexport {
2838 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2839 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2840 {\l_tmpa_str}
2841 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2842 {#2}{\l_tmpa_str}
2843 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2844 % \prop_item:Nn \l_stex_current_module_prop { origname },
2845 % \l_tmpa_str
2846 % }
2847 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2848 % #2,\l_tmpa_str
2849 % }
2850 % \tl_set:cx { #2 } {
2851 % \stex_invoke_structure:n { \l_tmpa_str }
2852 % }
2853 % }
2854
2855 \end{structural@feature}
2856 % \g_stex_last_feature_prop
2857 }

\instantiate

2858 \seq_new:N \l__stex_features_structure_field_seq
2859 \str_new:N \l__stex_features_structure_field_str
2860 \str_new:N \l__stex_features_structure_def_tl
2861 \prop_new:N \l__stex_features_structure_prop
2862 \NewDocumentCommand \instantiate { m O{} m }{
2863 \stex_smsmode_set_codes:
2864 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2865 \prop_set_eq:Nc \l__stex_features_structure_prop {
2866 c_stex_feature_\l_tmpa_str _prop
2867 }
2868 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2869 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2870 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2871 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2872 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2873 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2874 {!} \l_tmpa_tl
2875 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2876 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2877 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2878 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2879 }{
2880 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2881 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2882 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2883 \l_tmpa_tl
2884 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

2885         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2886         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2887     }{
2888         \tl_clear:N \l_tmpb_tl
2889     }
2890 }
2891 }{
2892     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2893     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2894         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2895         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2896         \tl_clear:N \l_tmpa_tl
2897     }{
2898         % TODO throw error
2899     }
2900 }
2901 % \l_tmpa_str: name
2902 % \l_tmpa_tl: definiens
2903 % \l_tmpb_tl: notation
2904 \tl_if_empty:NT \l__stex_features_structure_field_str {
2905     % TODO throw error
2906 }
2907 \str_clear:N \l_tmpb_str
2908
2909 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2910 \seq_map_inline:Nn \l_tmpa_seq {
2911     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2912     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2913     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2914         \seq_map_break:n {
2915             \str_set:Nn \l_tmpb_str { ####1 }
2916         }
2917     }
2918 }
2919 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2920 \l_tmpb_str
2921
2922 \tl_if_empty:NTF \l_tmpb_tl {
2923     \tl_if_empty:NF \l_tmpa_tl {
2924         \exp_args:Nx \use:n {
2925             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2926         }
2927     }
2928 }{
2929     \tl_if_empty:NTF \l_tmpa_tl {
2930         \exp_args:Nx \use:n {
2931             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2932         }
2933     }
2934 }{
2935     \exp_args:Nx \use:n {
2936         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2937         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2938     }

```

```

2939     }
2940   }
2941   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2942   % \prop_item:Nn \l_stex_current_module_prop {name} ?
2943   % #3/\l_stex_features_structure_field_str
2944   % \par
2945   % \expandafter\present\csname
2946   %   g_stex_symdecl_
2947   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2948   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2949   %   #3/\l_stex_features_structure_field_str
2950   %   _prop
2951   % \endcsname
2952 }
2953
2954 \tl_clear:N \l__stex_features_structure_def_tl
2955
2956 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2957 \seq_map_inline:Nn \l_tmpa_seq {
2958   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2959   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2960   \exp_args:Nx \use:n {
2961     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2962
2963       }
2964     }
2965
2966   \prop_if_exist:cF {
2967     g_stex_symdecl_
2968     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2969     \prop_item:Nn \l_stex_current_module_prop {name} ?
2970     #3/\l_tmpa_str
2971     _prop
2972   }{
2973     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2974     \l_tmpb_str
2975     \exp_args:Nx \use:n {
2976       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2977     }
2978   }
2979 }
2980
2981 \symdecl*[type={\STEXsymbol{module-type}}{
2982   \_stex_term_math_oms:nnnn {
2983     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2984     \prop_item:Nn \l__stex_features_structure_prop {name}
2985     }{}{0}{}
2986   }{}{#3}
2987
2988 % TODO: -> sms file
2989
2990 \tl_set:cx{ #3 }{
2991   \stex_invoke_structure:nnn {
2992     \prop_item:Nn \l_stex_current_module_prop {ns} ?

```



```

2993     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2994   } {
2995     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2996     \prop_item:Nn \l__stex_features_structure_prop {name}
2997   }
2998 }
2999
3000 }

```

(End definition for \instantiate. This function is documented on page ??.)

\stex_invoke_structure:nnn

```

3001 % #1: URI of the instance
3002 % #2: URI of the instantiated module
3003 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3004   \tl_if_empty:nTF{ #3 }{
3005     \prop_set_eq:Nc \l__stex_features_structure_prop {
3006       c_stex_feature_ #2 _prop
3007     }
3008     \tl_clear:N \l_tmpa_tl
3009     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3010     \seq_map_inline:Nn \l_tmpa_seq {
3011       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3012       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3013       \cs_if_exist:cT {
3014         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3015       }{
3016         \tl_if_empty:NF \l_tmpa_tl {
3017           \tl_put_right:Nn \l_tmpa_tl {,}
3018         }
3019         \tl_put_right:Nx \l_tmpa_tl {
3020           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3021         }
3022       }
3023     }
3024     \exp_args:No \mathstrut \l_tmpa_tl
3025   }{
3026     \stex_invoke_symbol:n{#1/#3}
3027   }
3028 }

```

(End definition for \stex_invoke_structure:nnn. This function is documented on page ??.)

```

3029 </package>

```

Chapter 25

STEX -Statements Implementation

```
3030 <*package>
3031
3032 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3033
3034 <@@=stex_statements>
3035
3036   Warnings and error messages
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
```

symboldoc

```
\NewDocumentEnvironment{symboldoc}{m}{
  \seq_set_split:Nnn \l_tmpa_seq , { #1 }
  \seq_clear:N \l_tmpb_seq
  \seq_map_inline:Nn \l_tmpa_seq {
    \str_if_eq:nnF{ ##1 }{}{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
}
\par
\exp_args:Nnnx
\begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
}{
\end{stex_annotate_env}
}

\seq_new:N \g_stex_statements_patched_seq

\cs_new_protected:Nn \stex_statements_set_patched:n {
  \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
}

\cs_new_protected:Nn \stex_statements_patch:nn {
  \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
```

```

3061 \AddToHook{begindocument}{
3062   \cs_if_exist:cTF{end#1}{
3063     \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3064     \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3065   }{
3066     \NewDocumentEnvironment{#1}{0{}}{
3067       \use:c{__stex_statements_#2_begin:n}{}
3068     }{
3069       \use:c{__stex_statements_#2_end:}
3070     }
3071   }
3072 }
3073 }
3074 }

```

25.1 Definitions

definition

```

3075
3076 \NewDocumentCommand \definiendum { 0{ } m m } {
3077   \stex_get_symbol:n { #2 }
3078   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3079   \scalatex_if:TF {
3080     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
3081   } {
3082     \exp_args:Nnx \defemph@uri { #3 } { \l_stex_get_symbol_uri_str }
3083   }
3084 }
3085 \stex_deactivate_macro:Nn \definiendum {definition~environments}
3086 \NewDocumentCommand \definame { 0{ } m } {
3087   % TODO: root
3088   \stex_get_symbol:n { #2 }
3089   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3090   \str_set:Nx \l_tmpa_str {
3091     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3092   }
3093   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3094   \scalatex_if:TF {
3095     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3096       \l_tmpa_str
3097     }
3098   } {
3099     \defemph@uri {
3100       \l_tmpa_str
3101     } { \l_stex_get_symbol_uri_str }
3102   }
3103 }
3104 \stex_deactivate_macro:Nn \definame {definition~environments}
3105
3106 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3107   \stex_reactivate_macro:N \definiendum
3108   \stex_reactivate_macro:N \definame
3109   \seq_set_split:Nnn \l_tmpa_seq , { #1 }

```

```

3110 \seq_clear:N \l_tmpb_seq
3111 \seq_map_inline:Nn \l_tmpa_seq {
3112   \str_if_eq:nnF{ ##1 }{}{
3113     \stex_get_symbol:n { ##1 }
3114     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3115       \l_stex_get_symbol_uri_str
3116     }
3117   }
3118 }
3119 \stex_smsmode_set_codes:
3120 \exp_args:Nnnx
3121 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3122 }
3123
3124 \cs_new_protected:Nn \__stex_statements_defi_end: {
3125   \end{stex_annotate_env}
3126 }

Hook:
3127 \stex_statements_patch:nn{definition}{defi}

```

25.2 Assertions

assertion

```

3128 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3129   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3130   \seq_clear:N \l_tmpb_seq
3131   \seq_map_inline:Nn \l_tmpa_seq {
3132     \str_if_eq:nnF{ ##1 }{}{
3133       \stex_get_symbol:n { ##1 }
3134       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3135         \l_stex_get_symbol_uri_str
3136       }
3137     }
3138   }
3139   \titleemph{Assertion}~
3140   \stex_smsmode_set_codes:
3141   \exp_args:Nnnx
3142   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3143 }
3144
3145 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3146   \end{stex_annotate_env}
3147 }

Hook:
3148 \stex_statements_patch:nn{assertion}{assertion}

```

theorem

```

3149 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3150   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3151   \seq_clear:N \l_tmpb_seq

```

```

3152 \seq_map_inline:Nn \l_tmpa_seq {
3153   \str_if_eq:nnF{ ##1 }{}{
3154     \stex_get_symbol:n { ##1 }
3155     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3156       \l_stex_get_symbol_uri_str
3157     }
3158   }
3159 }
3160 \titleemph{Theorem}~
3161 \stex_smsmode_set_codes:
3162 \exp_args:Nnnx
3163 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3164 }
3165
3166 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3167   \end{stex_annotate_env}
3168 }

```

Hook:

```

3169 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3170 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3171   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3172   \seq_clear:N \l_tmpb_seq
3173   \seq_map_inline:Nn \l_tmpa_seq {
3174     \str_if_eq:nnF{ ##1 }{}{
3175       \stex_get_symbol:n { ##1 }
3176       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3177         \l_stex_get_symbol_uri_str
3178       }
3179     }
3180   }
3181   \titleemph{Lemma}~
3182   \stex_smsmode_set_codes:
3183   \exp_args:Nnnx
3184   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3185 }
3186
3187 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3188   \end{stex_annotate_env}
3189 }

```

Hook:

```

3190 \stex_statements_patch:nn{lemma}{lemma}

```

axiom

```

3191 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3192   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3193   \seq_clear:N \l_tmpb_seq
3194   \seq_map_inline:Nn \l_tmpa_seq {
3195     \str_if_eq:nnF{ ##1 }{}{
3196       \stex_get_symbol:n { ##1 }

```

```

3197     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3198       \l_stex_get_symbol_uri_str
3199     }
3200   }
3201 }
3202 \titleemph{Axiom}~
3203 \stex_smsmode_set_codes:
3204 \exp_args:Nnnx
3205 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3206 }
3207
3208 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3209   \end{stex_annotate_env}
3210 }

```

Hook:

```

3211 \stex_statements_patch:nn{axiom}{axiom}

```

25.3 Examples

example

```

3212 \cs_new_protected:Nn \__stex_statements_example_begin:n {
3213   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3214   \seq_clear:N \l_tmpb_seq
3215   \seq_map_inline:Nn \l_tmpa_seq {
3216     \str_if_eq:nnF{ ##1 }{}{
3217       \stex_get_symbol:n { ##1 }
3218       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3219         \l_stex_get_symbol_uri_str
3220       }
3221     }
3222   }
3223   \titleemph{Example}~
3224   \stex_smsmode_set_codes:
3225   \exp_args:Nnnx
3226   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3227 }
3228
3229 \cs_new_protected:Nn \__stex_statements_example_end: {
3230   \end{stex_annotate_env}
3231 }

```

Hook:

```

3232 \stex_statements_patch:nn{example}{example}
3233 \</package>

```

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 26

STEX -Others Implementation

```
3234 <*package>
3235
3236 %%%%%%%%%% others.dtx %%%%%%%%%%
3237
3238 <@@=stex_others>
    Warnings and error messages
3239 % None

\MSC Math subject classifier

3240 \NewDocumentCommand \MSC {m} {
3241 % TODO
3242 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3243 \@ifpackageloaded{tikzinput}{
3244 \RequirePackage{stex-tikzinput}
3245 }{}
3246 </package>
```

Chapter 27

STEX -Metatheory Implementation

```
3247 <*package>
3248 <@@=stex_modules>
3249
3250 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3251
3252 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3253 \begingroup
3254 \stex_module_setup:nn{
3255   ns=\c_stex_metatheory_ns_str,
3256   meta=NONE
3257 }{Metatheory}
3258 \stex_reactivate_macro:N \symdecl
3259 \stex_reactivate_macro:N \notation
3260 \stex_reactivate_macro:N \symdef
3261 \ExplSyntaxOff
3262 \csname stex_suppress_html:n\endcsname{
3263   % is-a (a:A, a \in A, a is an A, etc.)
3264   \symdecl[args=ai]{isa}
3265   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3266   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3267   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3268
3269   % bind (\forall, \Pi, \lambda etc.)
3270   \symdecl[args=Bi]{bind}
3271   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3272   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3273   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
3274
3275   % dummy variable
3276   \symdecl{dummyvar}
3277   \notation[underscore]{dummyvar}{\comp\_}
3278   \notation[dot]{dummyvar}{\comp\cdot}
3279   \notation[dash]{dummyvar}{\comp{\rm --}}
3280
3281   %fromto (function space, Hom-set, implication etc.)
```



```

3282 \symdecl[args=ai]{fromto}
3283 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3284 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3285
3286 % mapto (lambda etc.)
3287 %\symdecl[args=Bi]{mapto}
3288 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3289 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3290 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3291
3292 % function/operator application
3293 \symdecl[args=ia]{apply}
3294 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3295 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3296
3297 % ‘type’ of all collections (sets, classes, types, kinds)
3298 \symdecl{collection}
3299 \notation[U]{collection}{\comp{\mathcal{U}}}
3300 \notation[set]{collection}{\comp{\textsf{Set}}}
3301
3302 % sequences
3303 \symdecl[args=1]{seqtype}
3304 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3305
3306 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3307 \notation[ui]{sequence-index}{#1^{#2}}
3308
3309 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses,}#1_{#3}}
3310 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses,}#1^{#3}}
3311 % ^ superceded by \aseqfromto and \livar/\uivar
3312
3313 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3314 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses,}#2}{#1\comp,#2}
3315 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
3316
3317 % letin (‘let’, local definitions, variable substitution)
3318 \symdecl[args=bii]{letin}
3319 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3320 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3321 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3322
3323 % structures
3324 \symdecl*[args=1]{module-type}
3325 \notation{module-type}{\mathtt{MOD} #1}
3326 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3327 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3328
3329 }
3330 \ExplSyntaxOn
3331 \stex_add_to_current_module:n{
3332   \let\nappa\apply
3333   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3334   \def\livar{\csname sequence-index\endcsname[li]}
3335   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3336     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3337     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3338     \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3339   }
3340   \__stex_modules_end_module:
3341   \endgroup
3342 \endpackage

```

Chapter 28

Tikzinput Implementation

```
3343 <*package>
3344
3345 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3346
3347 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3348 \RequirePackage{l3keys2e}
3349
3350 \keys_define:nn { tikzinput } {
3351   image .bool_set:N = \c_tikzinput_image_bool,
3352   image .default:n = false ,
3353 }
3354
3355 \ProcessKeysOptions { tikzinput }
3356
3357 \bool_if:NTF \c_tikzinput_image_bool {
3358   \RequirePackage{graphicx}
3359
3360   \providecommand\usetikzlibrary[]{}
3361   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
3362 }{
3363   \RequirePackage{tikz}
3364   \RequirePackage{standalone}
3365
3366   \newcommand \tikzinput [2] [] {
3367     \setkeys{Gin}{#1}
3368     \ifx \Gin@ewidth \Gin@exclamation
3369       \ifx \Gin@eheight \Gin@exclamation
3370         \input { #2 }
3371       \else
3372         \resizebox{!}{ \Gin@eheight }{
3373           \input { #2 }
3374         }
3375       \fi
3376     \else
3377       \ifx \Gin@eheight \Gin@exclamation
3378         \resizebox{ \Gin@ewidth }{!}{
3379           \input { #2 }
3380         }
3381     }
```

```

3381         \else
3382         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3383             \input { #2 }
3384         }
3385     \fi
3386 \fi
3387 }
3388 }
3389
3390 \newcommand \ctikzinput [2] [] {
3391     \begin{center}
3392         \tikzinput [ #1 ] { #2 }
3393     \end{center}
3394 }
3395
3396 \@ifpackageloaded{stex}{
3397     \RequirePackage{stex-tikzinput}
3398 }{}
3399
3400 </package>
3401 <*stex>
3402 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3403 \RequirePackage{stex}
3404 \RequirePackage{tikzinput}
3405
3406 \newcommand\mhtikzinput[2] [] {%
3407     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3408     \stex_in_repository:nn\Gin@mhrepos{
3409         \tikzinput[ #1 ]{\mhp{##1}{#2}}
3410     }
3411 }
3412 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[ #1 ]{ #2 }\end{center}}
3413 </stex>

```

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhp{

Chapter 29

document-structure.sty Implementation

29.1 The OMDoc Class

The functionality is spread over the `omdoc` class and package. The class provides the `document` environment and the `omdoc` element corresponds to it, whereas the package provides the concrete functionality.

```
3414 \*cls)
3415 \<@@=document_structure>
3416 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3417 \RequirePackage{l3keys2e,expl-keystr-compat}
```

29.2 Class Options

To initialize the `omdoc` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

`\omdoc@cls@class`

```
3418 \keys_define:nn{ document-structure / pkg }{
3419   class      .str_set_x:N = \c_document_structure_class_str,
3420   minimal    .bool_set:N = \c_document_structure_minimal_bool,
3421   report     .code:n      = {
3422     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3423     \str_set:Nn \c_document_structure_class_str {report}
3424   },
3425   book       .code:n      = {
3426     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3427     \str_set:Nn \c_document_structure_class_str {book}
3428   },
3429   bookpart   .code:n      = {
3430     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3431     \str_set:Nn \c_document_structure_class_str {book}
3432     \str_set:Nn \c_document_structure_topsect_str {chapter}
3433   },
```

```

3434 docopt      .str_set_x:N = \c_document_structure_docopt_str,
3435 unknown     .code:n      = {
3436   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3437 }
3438 }
3439 \ProcessKeysOptions{ document-structure / pkg }
3440 \str_if_empty:NT \c_document_structure_class_str {
3441   \str_set:Nn \c_document_structure_class_str {article}
3442 }
3443 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3444   {\c_document_structure_class_str}
3445

```

29.3 Beefing up the document environment

Now, – unless the option `minimal` is defined – we include the `stex` package

```

3446 \RequirePackage{omdoc}
3447 \bool_if:NF \c_document_structure_minimal_bool {
3448   \RequirePackage{stex-compatibility}

```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

document For the moment we do not use them on the L^AT_EX level, but the document identifier is picked up by L^AT_EXML.⁸

```

3449 \keys_define:nn { document-structure / document }{
3450   id .str_set_x:N = \c_document_structure_document_id_str
3451 }
3452 \let\__document_structure_orig_document=\document
3453 \renewcommand{\document}[1][]{
3454   \keys_set:nn{ document-structure / document }{ #1 }
3455   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3456   \__document_structure_orig_document
3457 }

```

Finally, we end the test for the `minimal` option.

```

3458 }
3459 \</cls>

```

29.4 Implementation: OMDoc Package

```

3460 \*package>
3461 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3462 \RequirePackage{expl-keystr-compat,13keys2e}

```

29.5 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).

⁸EdNOTE: faking documentkeys for now. @HANG, please implement

```

3463
3464 \keys_define:nn{ document-structure / pkg }{
3465   class      .str_set_x:N = \c_document_structure_class_str,
3466   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3467   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3468 }
3469 \ProcessKeysOptions{ document-structure / pkg }
3470 \str_if_empty:NT \c_document_structure_class_str {
3471   \str_set:Nn \c_document_structure_class_str {article}
3472 }
3473 \str_if_empty:NT \c_document_structure_topsect_str {
3474   \str_set:Nn \c_document_structure_topsect_str {section}
3475 }

```

Then we need to set up the packages by requiring the `sref` package to be loaded.

```

3476 \RequirePackage{xspace}
3477 \RequirePackage{comment}
3478 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

We set up triggers for the other languages, currently only German.

```

3479 \@ifpackageloaded{babel}{
3480   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3481   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3482     \input{omdoc-ngerman.ldf}
3483   }
3484 }{}
3485 %\AfterBabelLanguage{ngerman}{\input{omdoc-ngerman.ldf}}

```

`\section@level`

Finally, we set the `\section@level` macro that governs sectioning. The default is two (corresponding to the `article` class), then we set the defaults for the standard classes `book` and `report` and then we take care of the levels passed in via the `topsect` option.

```

3486 \int_new:N \l_document_structure_section_level_int
3487 \str_case:VnF \c_document_structure_topsect_str {
3488   {part}{
3489     \int_set:Nn \l_document_structure_section_level_int {0}
3490   }
3491   {chapter}{
3492     \int_set:Nn \l_document_structure_section_level_int {1}
3493   }
3494 }{
3495   \str_case:VnF \c_document_structure_class_str {
3496     {book}{
3497       \int_set:Nn \l_document_structure_section_level_int {0}
3498     }
3499     {report}{
3500       \int_set:Nn \l_document_structure_section_level_int {0}
3501     }
3502   }{
3503     \int_set:Nn \l_document_structure_section_level_int {2}
3504   }
3505 }

```

29.6 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the \LaTeX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.⁹

```
3506 \def\current@section@level{document}%
3507 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3508 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

(End definition for `\currentsectionlevel`. This function is documented on page ??.)

`\skipomgroup`

```
3509 \cs_new_protected:Npn \skipomgroup {
3510   \ifcase\l_document_structure_section_level_int
3511   \or\stepcounter{part}
3512   \or\stepcounter{chapter}
3513   \or\stepcounter{section}
3514   \or\stepcounter{subsection}
3515   \or\stepcounter{subsubsection}
3516   \or\stepcounter{paragraph}
3517   \or\stepcounter{subparagraph}
3518   \fi
3519 }
```

(End definition for `\skipomgroup`. This function is documented on page ??.)

`blindomgroup`

```
3520 \newcommand\at@begin@blindomgroup[1]{%
3521 \newenvironment{blindomgroup}
3522 {
3523   \int_incr:N\l_document_structure_section_level_int
3524   \at@begin@blindomgroup\l_document_structure_section_level_int
3525 }{}}
```

`\omgroup@nonum` convenience macro: `\omgroup@nonum{<level>}{<title>}` makes an unnumbered sectioning with title `<title>` at level `<level>`.

```
3526 \newcommand\omgroup@nonum[2]{
3527   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
3528   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3529 }
```

(End definition for `\omgroup@nonum`. This function is documented on page ??.)

`\omgroup@num` convenience macro: `\omgroup@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `omgroup` environment and – if it is use it. But how to do that depends on whether the `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```
3530 \newcommand\omgroup@num[2]{
```

⁹EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.


```

3531 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
3532   \@nameuse{#1}{#2}
3533 }{
3534   \cs_if_exist:NTF\rdfmata@sectioning{
3535     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
3536   }{
3537     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
3538   }
3539 }
3540 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
3541 }

```

(End definition for \omgroup@num. This function is documented on page ??.)

omgroup

```

3542 \keys_define:nn { document-structure / omgroup }{
3543   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
3544   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
3545   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
3546   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
3547   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
3548   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
3549   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
3550   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
3551   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
3552   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
3553 }
3554 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
3555   \str_clear:N \l__document_structure_omgroup_id_str
3556   \str_clear:N \l__document_structure_omgroup_date_str
3557   \clist_clear:N \l__document_structure_omgroup_creators_clist
3558   \clist_clear:N \l__document_structure_omgroup_contributors_clist
3559   \tl_clear:N \l__document_structure_omgroup_srccite_tl
3560   \tl_clear:N \l__document_structure_omgroup_type_tl
3561   \tl_clear:N \l__document_structure_omgroup_short_tl
3562   \tl_clear:N \l__document_structure_omgroup_display_tl
3563   \tl_clear:N \l__document_structure_omgroup_intro_tl
3564   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
3565   \keys_set:nn { document-structure / omgroup } { #1 }
3566 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The \at@begin@omgroup macro allows customization. It is run at the beginning of the omgroup, i.e. after the section heading.

```

3567 \newif\if@mainmatter\@mainmattertrue
3568 \newcommand\at@begin@omgroup[3] []{}

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

3569 \keys_define:nn { document-structure / sectioning }{
3570   name .str_set_x:N = \l__document_structure_sect_name_str ,
3571   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
3572   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
3573   num .bool_set:N = \l__document_structure_sect_num_bool ,
3574 }

```

```

3575 \cs_new_protected:Nn \__document_structure_sect_args:n {
3576   \str_clear:N \l__document_structure_sect_name_str
3577   \str_clear:N \l__document_structure_sect_ref_str
3578   \bool_set_false:N \l__document_structure_sect_clear_bool
3579   \bool_set_false:N \l__document_structure_sect_num_bool
3580   \keys_set:nn { document-structure / sectioning } { #1 }
3581 }
3582 \newcommand\omdoc@sectioning[3][]{
3583   \__document_structure_sect_args:n {#1}
3584   \let\omdoc@sect@name\l__document_structure_sect_name_str
3585   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
3586   \if@mainmatter% numbering not overridden by frontmatter, etc.
3587     \bool_if:NTF \l__document_structure_sect_num_bool {
3588       \omgroup@num{#2}{#3}
3589     }{
3590       \omgroup@nonum{#2}{#3}
3591     }
3592     \def\current@section@level{\omdoc@sect@name}
3593   \else
3594     \omgroup@nonum{#2}{#3}
3595   \fi
3596 }% if@mainmatter

```

and another one, if redefines the `\addtocontentsline` macro of L^AT_EX to import the respective macros. It takes as an argument a list of module names.

```

3597 \newcommand\omgroup@redefine@addtocontents[1]{%
3598   %\edef\__document_structureimport{#1}%
3599   %\@for\@I:=\__document_structureimport\do{%
3600     %\edef\@path{\csname module@\@I @path\endcsname}%
3601     %\@ifundefined{tf@toc}\relax%
3602     %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
3603   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
3604   %\def\addcontentsline##1##2##3{%
3605     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
3606   %\else% hyperref.sty not loaded
3607   %\def\addcontentsline##1##2##3{%
3608     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
3609   %\fi
3610 }% hyperref.sty loaded?

```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of `omgroups` in the `\omgroup@level` counter.

```

3611 \int_new:N \l_document_structure_omgroup_level_int
3612 \newenvironment{omgroup}[2][]{% keys, title
3613 {
3614   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

3615 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
3616   \omgroup@redefine@addtocontents{
3617     %\@ifundefined{module@id}\used@modules%
3618     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

3619     }
3620 }

```

now we only need to construct the right sectioning depending on the value of `\section@level`.

```

3621 \int_incr:N \l_document_structure_omgroup_level_int
3622 \int_incr:N \l_document_structure_section_level_int
3623 \ifcase\l_document_structure_section_level_int
3624   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
3625   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
3626   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
3627   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
3628   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
3629   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
3630   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
3631 \fi
3632 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
3633 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
3634 }% for customization
3635 {}

```

and finally, we localize the sections

```

3636 \newcommand\omdoc@part@kw{Part}
3637 \newcommand\omdoc@chapter@kw{Chapter}
3638 \newcommand\omdoc@section@kw{Section}
3639 \newcommand\omdoc@subsection@kw{Subsection}
3640 \newcommand\omdoc@subsubsection@kw{Subsubsection}
3641 \newcommand\omdoc@paragraph@kw{paragraph}
3642 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

29.7 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `omdoc` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

3643 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}

```

(End definition for `\printindex`. This function is documented on page ??.)

some classes (e.g. `book.cls`) already have `\frontmatter`, `\mainmatter`, and `\backmatter` macros. As we want to define `frontmatter` and `backmatter` environments, we save their behavior (possibly defining it) in `orig@*matter` macros and make them undefined (so that we can define the environments).

```

3644 \cs_if_exist:NTF\frontmatter{
3645   \let\__document_structure_orig_frontmatter\frontmatter
3646   \let\frontmatter\relax
3647 }{
3648   \tl_set:Nn\__document_structure_orig_frontmatter{
3649     \clearpage
3650     \@mainmatterfalse
3651     \pagenumbering{roman}
3652   }
3653 }
3654 \cs_if_exist:NTF\backmatter{

```

```

3655 \let\__document_structure_orig_backmatter\backmatter
3656 \let\backmatter\relax
3657 }{
3658 \tl_set:Nn\__document_structure_orig_backmatter{
3659 \clearpage
3660 \@mainmatterfalse
3661 \pagenumbering{roman}
3662 }
3663 }

```

Using these, we can now define the `frontmatter` and `backmatter` environments

frontmatter we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```

3664 \newenvironment{frontmatter}{
3665 \__document_structure_orig_frontmatter
3666 }{
3667 \cs_if_exist:NTF\mainmatter{
3668 \mainmatter
3669 }{
3670 \clearpage
3671 \@mainmattertrue
3672 \pagenumbering{arabic}
3673 }
3674 }

```

backmatter As `backmatter` is at the end of the document, we do nothing for `\endbackmatter`.

```

3675 \newenvironment{backmatter}{
3676 \__document_structure_orig_backmatter
3677 }{
3678 \cs_if_exist:NTF\mainmatter{
3679 \mainmatter
3680 }{
3681 \clearpage
3682 \@mainmattertrue
3683 \pagenumbering{arabic}
3684 }
3685 }

```

finally, we make sure that page numbering is arabic and we have main matter as the default

```

3686 \@mainmattertrue\pagenumbering{arabic}

```

\prematurestop We initialize `\afterprematurestop`, and provide `\prematurestop@endomgroup` which looks up `\omgroup@level` and recursively ends enough `{omgroup}`s.

```

3687 \newcommand\afterprematurestop{}
3688 \def\prematurestop@endomgroup{
3689 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
3690 \end{omgroup}
3691 \int_decr:N \l_document_structure_omgroup_level_int
3692 \prematurestop@endomgroup
3693 }
3694 }
3695 \providecommand\prematurestop{

```

```

3696 \message{Stopping sTeX processing prematurely}
3697 \prematuarestop@endomgroup
3698 \afterprematuarestop
3699 \end{document}
3700 }

```

(End definition for \prematuarestop. This function is documented on page ??.)

29.8 Global Variables

\setSGvar set a global variable

```

3701 \RequirePackage{etoolbox}
3702 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

(End definition for \setSGvar. This function is documented on page ??.)

\useSGvar use a global variable

```

3703 \newrobustcmd\useSGvar[1]{%
3704   \@ifundefined{sTeX@Gvar@#1}
3705   {\PackageError{omdoc}
3706     {The sTeX Global variable #1 is undefined}
3707     {set it with \protect\setSGvar}}
3708   \@nameuse{sTeX@Gvar@#1}}

```

(End definition for \useSGvar. This function is documented on page ??.)

\ifSGvar execute something conditionally based on the state of the global variable.

```

3709 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
3710   \@ifundefined{sTeX@Gvar@#1}
3711   {\PackageError{omdoc}
3712     {The sTeX Global variable #1 is undefined}
3713     {set it with \protect\setSGvar}}
3714   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

(End definition for \ifSGvar. This function is documented on page ??.)

Chapter 30

MiKoSlides – Implementation

30.1 Class and Package Options

We define some Package Options and switches for the `mikoslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `mikoslides` 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.

```
3715 \*cls)
3716 \@@=mikoslides)
3717 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
3718 \RequirePackage{l3keys2e,expl-keystr-compatible}
3719
3720 \keys_define:nn{mikoslides / cls}{
3721   class .code:n = {
3722     \PassOptionsToClass{\CurrentOption}{omdoc}
3723     \str_if_eq:nnT{#1}{book}{
3724       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
3725     }
3726     \str_if_eq:nnT{#1}{report}{
3727       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
3728     }
3729   },
3730   notes .bool_set:N = \c__mikoslides_notes_bool ,
3731   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
3732   unknown .code:n = {
3733     \PassOptionsToClass{\CurrentOption}{omdoc}
3734     \PassOptionsToClass{\CurrentOption}{beamer}
3735     \PassOptionsToPackage{\CurrentOption}{mikoslides}
3736   }
3737 }
3738 \ProcessKeysOptions{ mikoslides / cls }
3739 \bool_if:NTF \c__mikoslides_notes_bool {
3740   \PassOptionsToPackage{notes=true}{mikoslides}
3741 }{
3742   \PassOptionsToPackage{notes=false}{mikoslides}
3743 }
3744 \</cls>
```

now we do the same for the mikoslides package.

```

3745 <*package>
3746 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
3747 \RequirePackage{l3keys2e,expl-keystr-compat}
3748
3749 \keys_define:nn{mikoslides / pkg}{
3750   topsect          .str_set_x:N = \c__mikoslides_topsect_str,
3751   defaulttopsect   .str_set_x:N = \c__mikoslides_defaulttopsec_str,
3752   notes            .bool_set:N = \c__mikoslides_notes_bool ,
3753   slides           .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
3754   sectocframes     .bool_set:N = \c__mikoslides_sectocframes_bool ,
3755   frameimages      .bool_set:N = \c__mikoslides_frameimages_bool ,
3756   fiboxed          .bool_set:N = \c__mikoslides_fiboxed_bool ,
3757   noproblems       .bool_set:N = \c__mikoslides_noproblems_bool,
3758   unknown          .code:n      = {
3759     \PassOptionsToClass{\CurrentOption}{stex}
3760     \PassOptionsToClass{\CurrentOption}{tikzinput}
3761   }
3762 }
3763 \ProcessKeysOptions{ mikoslides / pkg }
3764 \newif\ifnotes
3765 \bool_if:NTF \c__mikoslides_notes_bool {
3766   \notesttrue
3767 }{
3768   \notesfalse
3769 }
3770

```

we give ourselves a macro `\@@topsect` that needs only be evaluated once, so that the `\ifdefstring` conditionals work below.

```

3771 \str_if_empty:NTF \c__mikoslides_topsect_str {
3772   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
3773 }{
3774   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
3775 }
3776 </package>

```

Depending on the options, we either load the article-based omdoc or the beamer class (and set some counters).

```

3777 <*cls>
3778 \bool_if:NTF \c__mikoslides_notes_bool {
3779   \LoadClass{omdoc}
3780 }{
3781   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
3782   \newcounter{Item}
3783   \newcounter{paragraph}
3784   \newcounter{subparagraph}
3785   \newcounter{Hfootnote}
3786   \RequirePackage{omdoc}
3787 }

```

now it only remains to load the mikoslides package that does all the rest.

```

3788 \RequirePackage{mikoslides}
3789 </cls>

```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the `STEX` packages. The first batch of packages we want are loaded on `mikoslides.sty`. These are the general ones, we will load the `STEX`-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```

3790 \*package>
3791 \RequirePackage{stex-compatibility}
3792 \RequirePackage{stex-tikzinput}
3793 \bool_if:NT \c__mikoslides_notes_bool {
3794   \RequirePackage{a4wide}
3795   \RequirePackage{marginnote}
3796   \PassOptionsToPackage{dvipsnames,svgnames}{xcolor}
3797   \RequirePackage{mdframed}
3798   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
3799   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
3800 }
3801 \RequirePackage{etoolbox}
3802 \RequirePackage{amssymb}
3803 \RequirePackage{amsmath}
3804 \RequirePackage{comment}
3805 \RequirePackage{textcomp}
3806 \RequirePackage{url}
3807 \RequirePackage{graphicx}
3808 \RequirePackage{pgf}

```

30.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.¹⁰

```

3809 \bool_if:NT \c__mikoslides_notes_bool {
3810   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
3811 }

```

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

```

3812 \newcounter{slide}
3813 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
3814 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

note The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```

3815 \bool_if:NTF \c__mikoslides_notes_bool {
3816   \renewenvironment{note}{\ignorespaces}{\}
3817 }{
3818   \excludecomment{note}
3819 }

```

¹⁰EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

We first set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
3820 \bool_if:NT \c__mikoslides_notes_bool {
3821   \newlength{\slideframewidth}
3822   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
3823 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
3824   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
3825     \bool_set_true:N #1
3826   }{
3827     \bool_set_false:N #1
3828   }
3829 }
3830 \keys_define:nn{mikoslides / frame}{
3831   label .str_set_x:N = \l__mikoslides_frame_label_str,
3832   allowframebreaks .code:n = {
3833     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
3834   },
3835   allowdisplaybreaks .code:n = {
3836     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
3837   },
3838   fragile .code:n = {
3839     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
3840   },
3841   shrink .code:n = {
3842     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
3843   },
3844   squeeze .code:n = {
3845     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
3846   },
3847   t .code:n = {
3848     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
3849   },
3850 }
3851 \cs_new_protected:Nn \__mikoslides_frame_args:n {
3852   \str_clear:N \l__mikoslides_frame_label_str
3853   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
3854   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
3855   \bool_set_true:N \l__mikoslides_frame_fragile_bool
3856   \bool_set_true:N \l__mikoslides_frame_shrink_bool
3857   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
3858   \bool_set_true:N \l__mikoslides_frame_t_bool
3859   \keys_set:nn { mikoslides / frame }{ #1 }
3860 }
```

We define the environment, read them, and construct the slide number and label.

```
3861 \renewenvironment{frame}[1][]{
3862   \__mikoslides_frame_args:n{#1}
3863   \sffamily
3864   \stepcounter{slide}
3865   \def\@currentlabel{\theslide}
3866   \str_if_empty:NF \l__mikoslides_frame_label_str {
3867     \label{\l__mikoslides_frame_label_str}
```

```
3868 }
```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```
3869 \def\itemize@level{outer}
3870 \def\itemize@outer{outer}
3871 \def\itemize@inner{inner}
3872 \renewcommand\newpage{\addtocounter{framenum}{1}}
3873 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
3874 \renewenvironment{itemize}{
3875   \ifx\itemize@level\itemize@outer
3876     \def\itemize@label{\rhd$}
3877   \fi
3878   \ifx\itemize@level\itemize@inner
3879     \def\itemize@label{$\scriptstyle\rhd$}
3880   \fi
3881   \begin{list}
3882   {\itemize@label}
3883   {\setlength{\labelsep}{.3em}
3884    \setlength{\labelwidth}{.5em}
3885    \setlength{\leftmargin}{1.5em}
3886   }
3887   \edef\itemize@level{\itemize@inner}
3888 }{
3889   \end{list}
3890 }
```

We create the box with the `mdframed` environment from the `equinymous` package.

```
3891 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
3892 }{
3893   \medskip\miko@slidelabel\end{mdframed}
3894 }
```

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

`\frametitle`

```
3895 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
3896 }
```

(End definition for \frametitle. This function is documented on page ??.)

EdN:11

`\pause` 11

```
3897 \bool_if:NT \c__mikoslides_notes_bool {
3898   \newcommand\pause{}
3899 }
```

(End definition for \pause. This function is documented on page ??.)

`nomtext`

```
3900 \bool_if:NTF \c__mikoslides_notes_bool {
3901   \newenvironment{nomtext}[1][\begin{omtext}[#1]}\end{omtext}}
3902 }{
3903   \excludecomment{nomtext}
3904 }
```

¹¹EdNOTE: MK: fake it in notes mode for now

nomgroup

```
3905 \bool_if:NTF \c__mikoslides_notes_bool {
3906   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
3907 }{
3908   \excludecomment{nomgroup}
3909 }
```

ndefinition

```
3910 \bool_if:NTF \c__mikoslides_notes_bool {
3911   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
3912 }{
3913   \excludecomment{ndefinition}
3914 }
```

nassertion

```
3915 \bool_if:NTF \c__mikoslides_notes_bool {
3916   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
3917 }{
3918   \excludecomment{nassertion}
3919 }
```

nsproof

```
3920 \bool_if:NTF \c__mikoslides_notes_bool {
3921   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
3922 }{
3923   \excludecomment{nsproof}
3924 }
```

nexample

```
3925 \bool_if:NTF \c__mikoslides_notes_bool {
3926   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
3927 }{
3928   \excludecomment{nexample}
3929 }
```

\inputref@*skip We customize the hooks for in \inputref.

```
3930 \def\inputref@preskip{\smallskip}
3931 \def\inputref@postskip{\medskip}
```

(End definition for \inputref@*skip. This function is documented on page ??.)

\inputref*

```
3932 \let\orig@inputref\inputref
3933 \def\inputref{\@ifstar\ninputref\orig@inputref}
3934 \newcommand\ninputref[2] [] {
3935   \bool_if:NT \c__mikoslides_notes_bool {
3936     \orig@inputref[#1]{#2}
3937   }
3938 }
```

(End definition for \inputref*. This function is documented on page ??.)

30.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

\setslidelogo The default logo is the \TeX logo. Customization can be done by `\setslidelogo{<logo name>}`.

```

3939 \newlength{\slidelogoheight}
3940
3941 \bool_if:NTF \c__mikoslides_notes_bool {
3942   \setlength{\slidelogoheight}{.4cm}
3943 }{
3944   \setlength{\slidelogoheight}{1cm}
3945 }
3946 \newsavebox{\slidelogo}
3947 \sbox{\slidelogo}{\TeX}
3948 \newrobustcmd{\setslidelogo}[1]{
3949   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
3950 }
```

(End definition for `\setslidelogo`. This function is documented on page ??.)

\setsource `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

```

3951 \def\source{Michael Kohlhase}% customize locally
3952 \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

(End definition for `\setsource`. This function is documented on page ??.)

\setlicensing Now, we set up the copyright and licensing. By default we use the Creative Commons Attribution-ShareAlike license 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.

```

3953 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
3954 \newsavebox{\cclogo}
3955 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
3956 \newif\ifcchref\cchreffalse
3957 \AtBeginDocument{
3958   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
3959 }
3960 \def\licensing{
3961   \ifcchref
3962     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
3963   \else
3964     {\usebox{\cclogo}}
3965   \fi
3966 }
3967 \newrobustcmd{\setlicensing}[2][]{
3968   \def@url{#1}
3969   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
3970   \ifx@url@empty
3971     \def\licensing{{\usebox{\cclogo}}}
3972   \else
3973     \def\licensing{
```

```

3974     \ifcchref
3975     \href{#1}{\usebox{\cclogo}}
3976   \else
3977     {\usebox{\cclogo}}
3978   \fi
3979 }
3980 \fi
3981 }

```

(End definition for `\setlicensing`. This function is documented on page ??.)

EdN:12 `\slidelabel` Now, we set up the slide label for the article mode.¹²

```

3982 \newrobustcmd\miko@slidelabel{
3983   \vbox to \slidelogoheight{
3984     \vss\hbox to \slidewidth
3985     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
3986   }
3987 }

```

(End definition for `\slidelabel`. This function is documented on page ??.)

30.4 Frame Images

`\frameimage` We have to make sure that the width is overwritten, for that we check the `\Gin@ewidth` macro from the `graphicx` package. We also add the `label` key.

```

3988 \def\Gin@mhrepos{}
3989 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
3990 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
3991 \newrobustcmd\frameimage[2][{}{
3992   \stepcounter{slide}
3993   \bool_if:NT \c__mikoslides_frameimages_bool {
3994     \def\Gin@ewidth{} \setkeys{Gin}{#1}
3995     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
3996     \begin{center}
3997       \bool_if:NTF \c__mikoslides_fiboxed_bool {
3998         \fbox{
3999           \ifx\Gin@ewidth\@empty
4000             \ifx\Gin@mhrepos\@empty
4001               \mhgraphics[width=\slidewidth,#1]{#2}
4002             \else
4003               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4004             \fi
4005           \else% Gin@ewidth empty
4006             \ifx\Gin@mhrepos\@empty
4007               \mhgraphics[#1]{#2}
4008             \else
4009               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4010             \fi
4011           \fi% Gin@ewidth empty
4012         }
4013       }{
4014         \ifx\Gin@ewidth\@empty

```

¹²EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

4015         \ifx\Gin@mhrepos\@empty
4016             \mhgraphics[width=\slidewidth,#1]{#2}
4017         \else
4018             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4019         \fi
4020         \ifx\Gin@mhrepos\@empty
4021             \mhgraphics[#1]{#2}
4022         \else
4023             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4024         \fi
4025     \fi% Gin@ewidth empty
4026 }
4027 \end{center}
4028 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4029 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4030 }
4031 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page ??.)

30.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

4032 \sffamily

```

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to do is to adapt the green so that it is dark enough for most beamers

```

4033 \AddToHook{begindocument}{
4034     \definecolor{green}{rgb}{0,.5,0}
4035     \definecolor{purple}{cmyk}{.3,1,0,.17}
4036 }

```

We customize the `\defemph`, `\symrefemph`, `\compemph`, and `\titlemph` macros with colors. Furthermore we customize the `\@@lec` macro for the appearance of line end comments in `\lec`.

```

4037 % \def\STpresent#1{\textcolor{blue}{#1}}
4038 \def\defemph#1{\textcolor{magenta}{#1}}
4039 \def\symrefemph#1{\textcolor{cyan}{#1}}
4040 \def\compemph#1{\textcolor{blue}{#1}}
4041 \def\titleemph#1{\textcolor{blue}{#1}}
4042 \def\__mikoslideslec#1{\textcolor{green}{#1}}

```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```

4043 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4044 \def\smalltextwarning{
4045     \pgfuseimage{miko@small@dbend}
4046     \xspace
4047 }
4048 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4049 \newrobustcmd\textwarning{
4050   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4051   \xspace
4052 }
4053 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4054 \newrobustcmd\bigtextwarning{
4055   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4056   \xspace
4057 }

(End definition for \textwarning. This function is documented on page ??.)

4058 \newrobustcmd\putgraphicsat[3]{
4059   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4060 }
4061 \newrobustcmd\putat[2]{
4062   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4063 }

```

30.6 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```

4064 \bool_if:NT \c__mikoslides_sectocframes_bool {
4065   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4066     \newcounter{chapter}\counterwithin*{section}{chapter}
4067   }{
4068     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4069       \newcounter{chapter}\counterwithin*{section}{chapter}
4070     }
4071   }
4072 }

```

`\section@level` We set the `\section@level` counter that governs sectioning according to the class options. We also introduce the sectioning counters accordingly.

```

\section@level
4073 \@ifpackageloaded{omdoc}{
4074   \str_case:VnF \__mikoslidestopsect {
4075     {part}{
4076       \int_set:Nn \l_document_structure_section_level_int {0}
4077       \def\thesection{\arabic{chapter}.\arabic{section}}
4078       \def\part@prefix{\arabic{chapter}.}
4079     }
4080     {chapter}{
4081       \int_set:Nn \l_document_structure_section_level_int {1}
4082       \def\thesection{\arabic{chapter}.\arabic{section}}
4083       \def\part@prefix{\arabic{chapter}.}
4084     }
4085   }{
4086     \int_set:Nn \l_document_structure_section_level_int {2}
4087     \def\part@prefix{}
4088   }

```

```

4089 }
4090
4091 \bool_if:NF \c__mikoslides_notes_bool { % only in slides

```

(End definition for \section@level. This function is documented on page ??.)

The new counters are used in the omgroup environment that choses the L^AT_EX sectioning macros according to \section@level.

omgroup

```

4092 \renewenvironment{omgroup}[2][]{
4093   \_document_structure_omgroup_args:n { #1 }
4094   \int_incr:N \l_document_structure_omgroup_level_int
4095   \int_incr:N \l_document_structure_section_level_int
4096   \bool_if:NT \c__mikoslides_sectocframes_bool {
4097     \stepcounter{slide}
4098     \begin{frame}[noframenumbering]
4099     \vfill\Large\centering
4100     \red{
4101       \ifcase\l_document_structure_section_level_int\or
4102         \stepcounter{part}
4103         \def\_mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4104         \def\currentsectionlevel{\omdoc@part@kw}
4105       \or
4106         \stepcounter{chapter}
4107         \def\_mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4108         \def\currentsectionlevel{\omdoc@chapter@kw}
4109       \or
4110         \stepcounter{section}
4111         \def\_mikoslideslabel{\part@prefix\arabic{section}}
4112         \def\currentsectionlevel{\omdoc@section@kw}
4113       \or
4114         \stepcounter{subsection}
4115         \def\_mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4116         \def\currentsectionlevel{\omdoc@subsection@kw}
4117       \or
4118         \stepcounter{subsubsection}
4119         \def\_mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4120         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4121       \or
4122         \stepcounter{mparagraph}
4123         \def\_mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsubsection}.\arabic{subsubsubsection}}
4124         \def\currentsectionlevel{\omdoc@paragraph@kw}
4125       \fi% end ifcase
4126       \_mikoslideslabel\sref@label@id\_mikoslideslabel
4127       \quad #2%
4128     }%
4129     \vfill%
4130     \end{frame}%
4131   }
4132   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4133 }{}
4134 }

```

We set up a beamer template for theorems like ams style, but without a block environment.


```

4135 \def\inserttheorembodyfont{\normalfont}
4136 \bool_if:NF \c__mikoslides_notes_bool {
4137   \defbeamertemplate{theorem begin}{miko}
4138   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4139     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4140     \inserttheorempunctuation\inserttheorembodyfont\xspace}
4141   \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

4142 \setbeamertemplate{theorems}[miko]

```

The following fixes an error I do not understand, this has something to do with beamer compatibility, which has similar definitions but only up to 1.

```

4143 \expandafter\def\csname Parent2\endcsname{}
4144 }
4145 \bool_if:NT \c__mikoslides_notes_bool {
4146   \renewenvironment{columns}[1][]{%
4147     \par\noindent%
4148     \begin{minipage}%
4149       \slidewidth\centering\leavevmode%
4150   }{%
4151     \end{minipage}\par\noindent%
4152   }%
4153   \newsavebox\columnbox%
4154   \renewenvironment<>{column}[2][]{%
4155     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4156   }{%
4157     \end{minipage}\end{lrbox}\usebox\columnbox%
4158   }%
4159 }
4160 \bool_if:NTF \c__mikoslides_noproblems_bool {
4161   \newenvironment{problems}{}{}
4162 }{
4163   \excludecomment{problems}
4164 }

```

30.7 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.

```

4165 \gdef\printexcursions{}
4166 \newcommand\excursionref[2]{% label, text
4167   \bool_if:NT \c__mikoslides_notes_bool {
4168     \begin{omtext}[title=Excursion]
4169       #2 \sref[fallback=the appendix]{#1}.
4170     \end{omtext}
4171   }
4172 }
4173 \newcommand\activate@excursion[2][{}{
4174   \gappto\printexcursions{\inputref[#1]{#2}}
4175 }
4176 \newcommand\excursion[4][{}{ repos, label, path, text

```

```

4177 \bool_if:NT \c__mikoslides_notes_bool {
4178   \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4179 }
4180 }

```

(End definition for \excursion. This function is documented on page ??.)

\excursiongroup

```

4181 \keys_define:nn{mikoslides / excursiongroup }{
4182   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4183   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
4184   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4185 }
4186 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4187   \tl_clear:N \l__mikoslides_excursion_intro_tl
4188   \str_clear:N \l__mikoslides_excursion_id_str
4189   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4190   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4191 }
4192 \newcommand\excursiongroup[1][]{
4193   \__mikoslides_excursion_args:n{ #1 }
4194   \ifdefempty\printexcursions{}% only if there are excursions
4195   {
4196     \begin{omgroup}[#1]{Excursions}%
4197     \ifdefempty\l__mikoslides_excursion_intro_tl{
4198       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4199         \l__mikoslides_excursion_intro_tl
4200       }
4201     }
4202     \printexcursions%
4203     \end{omgroup}
4204   }
4205 }
4206 \end{package}

```

(End definition for \excursiongroup. This function is documented on page ??.)

Chapter 31

The Implementation

31.1 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.

```
4207 <*package>
4208 <@@=problems>
4209 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4210 \RequirePackage{l3keys2e,expl-keystr-compatible}
4211
4212 \keys_define:nn { problem / pkg }{
4213   notes      .default:n    = { true },
4214   notes      .bool_set:N   = \c__problems_notes_bool,
4215   gnotes     .default:n    = { true },
4216   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4217   hints      .default:n    = { true },
4218   hints      .bool_set:N   = \c__problems_hints_bool,
4219   solutions  .default:n    = { true },
4220   solutions  .bool_set:N   = \c__problems_solutions_bool,
4221   pts        .default:n    = { true },
4222   pts        .bool_set:N   = \c__problems_pts_bool,
4223   min        .default:n    = { true },
4224   min        .bool_set:N   = \c__problems_min_bool,
4225   boxed      .default:n    = { true },
4226   boxed      .bool_set:N   = \c__problems_boxed_bool
4227 }
4228 \def\solutionstrue{
4229   \bool_set_true:N \c__problems_solutions_bool
4230 }
4231 \def\solutionsfalse{
4232   \bool_set_false:N \c__problems_solutions_bool
4233 }
4234
4235 \ProcessKeysOptions{ problem / pkg }
4236
4237 Then we make sure that the necessary packages are loaded (in the right versions).
4238 \RequirePackage{stex-compatibility}
```

```
4237 \RequirePackage{comment}
```

The next package relies on the L^AT_EX3 kernel, which L^AT_EXML only partially supports. As it is purely presentational, we only load it when the boxed option is given and we run L^AT_EXML.

```
4238 \bool_if:NT \c__problems_boxed_bool { \RequirePackage{mdframed} }
```

\prob@*@kw For multilinguality, we define internal macros for keywords that can be specialized in *.ldf files.

```
4239 \def\prob@problem@kw{Problem}
4240 \def\prob@solution@kw{Solution}
4241 \def\prob@hint@kw{Hint}
4242 \def\prob@note@kw{Note}
4243 \def\prob@gnote@kw{Grading}
4244 \def\prob@pt@kw{pt}
4245 \def\prob@min@kw{min}
```

(End definition for \prob@*@kw. This function is documented on page ??.)

For the other languages, we set up triggers

```
4246 \@ifpackageloaded{babel}{
4247   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4248   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4249     \input{problem-ngerman.ldf}
4250   }
4251   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4252     \input{problem-finnish.ldf}
4253   }
4254   \clist_if_in:NnT \l_tmpa_clist {french}{
4255     \input{problem-french.ldf}
4256   }
4257   \clist_if_in:NnT \l_tmpa_clist {russian}{
4258     \input{problem-russian.ldf}
4259   }
4260 }{}
```

31.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
4261 \keys_define:nn{ problem / problem }{
4262   id      .str_set:x:N = \l__problems_prob_id_str,
4263   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4264   min     .tl_set:N    = \l__problems_prob_min_tl,
4265   title   .tl_set:N    = \l__problems_prob_title_tl,
4266   refnum  .int_set:N   = \l__problems_prob_refnum_int
4267 }
4268 \cs_new_protected:Nn \__problems_prob_args:n {
4269   \str_clear:N \l__problems_prob_id_str
4270   \tl_clear:N \l__problems_prob_pts_tl
4271   \tl_clear:N \l__problems_prob_min_tl
4272   \tl_clear:N \l__problems_prob_title_tl
4273   \int_zero_new:N \l__problems_prob_refnum_int
```

```

4274 \keys_set:nn { problem / problem }{ #1 }
4275 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4276   \let\l__problems_inclprob_refnum_int\undefined
4277 }
4278 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4279 \newcounter{problem}
4280 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}

```

(End definition for `\numberproblemsin`. This function is documented on page ??.)

`\prob@label` We provide the macro `\prob@label` to redefine later to get context involved.

```

4281 \newcommand\prob@label[1]{#1}

```

(End definition for `\prob@label`. This function is documented on page ??.)

`\prob@number` We consolidate the problem number into a reusable internal macro

```

4282 \newcommand\prob@number{
4283   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4284     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4285   }{
4286     \int_if_exist:NTF \l__problems_prob_refnum_int {
4287       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4288     }{
4289       \prob@label\theproblem
4290     }
4291   }
4292 }

```

(End definition for `\prob@number`. This function is documented on page ??.)

`\prob@title` We consolidate the problem title into a reusable internal macro as well. `\prob@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```

4293 \newcommand\prob@title[3]{%
4294   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4295     #2 \l__problems_inclprob_title_tl #3
4296   }{
4297     \tl_if_exist:NTF \l__problems_prob_title_tl {
4298       #2 \l__problems_prob_title_tl #3
4299     }{
4300       #1
4301     }
4302   }
4303 }

```

(End definition for `\prob@title`. This function is documented on page ??.)

With these the problem header is a one-liner

`\prob@heading` We consolidate the problem header line into a separate internal macro that can be reused in various settings.

```

4304 \def\prob@heading{
4305   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4306   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
4307 }

```

(End definition for `\prob@heading`. This function is documented on page ??.)

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`problem`

```

4308 \newenvironment{problem}[1][1]{
4309   \_problems_prob_args:n{#1}%\sref@target%
4310   \@in@omtexttrue% we are in a statement (for inline definitions)
4311   \stepcounter{problem}\record@problem
4312   \def\current@section@level{\prob@problem@kw}
4313   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4314 }%
4315   {\smallskip}
4316   \bool_if:NT \c__problems_boxed_bool {
4317     \surroundwithmdframed{problem}
4318   }

```

`\record@problem` This macro records information about the problems in the `*.aux` file.

```

4319 \def\record@problem{
4320   \protected@write\@auxout{}
4321   {
4322     \string\@problem{\prob@number}
4323     {
4324       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4325         \l__problems_inclprob_pts_tl
4326       }{
4327         \l__problems_prob_pts_tl
4328       }
4329     }%
4330     {
4331       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4332         \l__problems_inclprob_min_tl
4333       }{
4334         \l__problems_prob_min_tl
4335       }
4336     }
4337   }
4338 }

```

(End definition for `\record@problem`. This function is documented on page ??.)

`\@problem` This macro acts on a problem's record in the `*.aux` file. It does not have any functionality here, but can be redefined elsewhere (e.g. in the `assignment` package).

```

4339 \def\@problem#1#2#3{}

```

(End definition for \@problem. This function is documented on page ??.)

solution The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```

4340 \keys_define:nn { problem / solution }{
4341   id          .str_set_x:N = \l__problems_solution_id_str ,
4342   for         .tl_set:N    = \l__problems_solution_for_tl ,
4343   height      .dim_set:N   = \l__problems_solution_height_dim ,
4344   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4345   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4346   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4347 }
4348 \cs_new_protected:Nn \__problems_solution_args:n {
4349   \str_clear:N \l__problems_solution_id_str
4350   \tl_clear:N \l__problems_solution_for_tl
4351   \tl_clear:N \l__problems_solution_srccite_tl
4352   \clist_clear:N \l__problems_solution_creators_clist
4353   \clist_clear:N \l__problems_solution_contributors_clist
4354   \dim_zero:N \l__problems_solution_height_dim
4355   \keys_set:nn { problem / solution }{ #1 }
4356 }

```

the next step is to define a helper macro that does what is needed to start a solution.

```

4357 \newcommand\@startsolution[1][ ]{
4358   \__problems_solution_args:n { #1 }
4359   \@in@omtexttrue% we are in a statement.
4360   \bool_if:NF \c__problems_boxed_bool { \hrule }
4361   \smallskip\noindent
4362   {\textbf\prob@solution@kw : \enspace}
4363   \begin{small}
4364   \def\current@section@level{\prob@solution@kw}
4365   \ignorespacesandpars
4366 }

```

\startsolutions for the `\startsolutions` macro we use the `\specialcomment` macro from the `comment` package. Note that we use the `\@startsolution` macro in the start codes, that parses the optional argument.

```

4367 \newcommand\startsolutions{
4368   \specialcomment{solution}{\@startsolution}{
4369     \bool_if:NF \c__problems_boxed_bool {
4370       \hrule\medskip
4371     }
4372     \end{small}%
4373   }
4374   \bool_if:NT \c__problems_boxed_bool {
4375     \surroundwithmdframed{solution}
4376   }
4377 }

```

(End definition for \startsolutions. This function is documented on page ??.)

\stopsolutions

```

4378 \newcommand\stopsolutions{\excludecomment{solution}}

```

(End definition for \stopsolutions. This function is documented on page ??.)

so it only remains to start/stop solutions depending on what option was specified.

```

4379 \bool_if:NTF \c_problems_solutions_bool {
4380   \startsolutions
4381 }{
4382   \stopsolutions
4383 }

```

exnote

```

4384 \bool_if:NTF \c_problems_notes_bool {
4385   \newenvironment{exnote}[1][]{
4386     \par\smallskip\hrule\smallskip
4387     \noindent\textbf{\prob@note@kw : }\small
4388   }{
4389     \smallskip\hrule
4390   }
4391 }{
4392   \excludecomment{exnote}
4393 }

```

hint

```

4394 \bool_if:NTF \c_problems_notes_bool {
4395   \newenvironment{hint}[1][]{
4396     \par\smallskip\hrule\smallskip
4397     \noindent\textbf{\prob@hint@kw :~ }\small
4398   }{
4399     \smallskip\hrule
4400   }
4401   \newenvironment{exhint}[1][]{
4402     \par\smallskip\hrule\smallskip
4403     \noindent\textbf{\prob@hint@kw :~ }\small
4404   }{
4405     \smallskip\hrule
4406   }
4407 }{
4408   \excludecomment{hint}
4409   \excludecomment{exhint}
4410 }

```

gnote

```

4411 \bool_if:NTF \c_problems_notes_bool {
4412   \newenvironment{gnote}[1][]{
4413     \par\smallskip\hrule\smallskip
4414     \noindent\textbf{\prob@gnote@kw : }\small
4415   }{
4416     \smallskip\hrule
4417   }
4418 }{
4419   \excludecomment{gnote}
4420 }

```


31.3 Multiple Choice Blocks

EdN:13

mcb 13

```

4421 \newenvironment{mcb}{
4422   \begin{enumerate}
4423 }{
4424   \end{enumerate}
4425 }
```

we define the keys for the mcc macro

```

4426 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4427   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4428     \bool_set_true:N #1
4429   }{
4430     \bool_set_false:N #1
4431   }
4432 }
4433 \keys_define:nn { problem / mcc }{
4434   id          .str_set:x:N = \l__problems_mcc_id_str ,
4435   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
4436   T           .default:n   = { true } ,
4437   T           .bool_set:N   = \l__problems_mcc_t_bool ,
4438   F           .default:n   = { true } ,
4439   F           .bool_set:N   = \l__problems_mcc_f_bool ,
4440   Ttext       .code:n      = {
4441     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4442   } ,
4443   Ftext       .code:n      = {
4444     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4445   }
4446 }
4447 \cs_new_protected:Nn \l__problems_mcc_args:n {
4448   \str_clear:N \l__problems_mcc_id_str
4449   \tl_clear:N \l__problems_mcc_feedback_tl
4450   \bool_set_true:N \l__problems_mcc_t_bool
4451   \bool_set_true:N \l__problems_mcc_f_bool
4452   \bool_set_true:N \l__problems_mcc_Ttext_bool
4453   \bool_set_false:N \l__problems_mcc_Ftext_bool
4454   \keys_set:nn { problem / mcc }{ #1 }
4455 }
```

\mcc

```

4456 \newcommand\mcc[2][]{
4457   \l__problems_mcc_args:n{ #1 }
4458   \item #2
4459   \bool_if:NT \c__problems_solutions_bool {
4460     \
4461     \bool_if:NT \l__problems_mcc_t_bool {
4462       % TODO!
4463       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
4464     }
4465     \bool_if:NT \l__problems_mcc_f_bool {
```

¹³EdNOTE: MK: maybe import something better here from a dedicated MC package

```

4466      % TODO!
4467      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4468    }
4469    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4470      !
4471    }{
4472      \l__problems_mcc_feedback_tl
4473    }
4474  }
4475 } %solutions

```

(End definition for \mcc. This function is documented on page ??.)

31.4 Including Problems

`\includeproblem` The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

4476
4477 \keys_define:nn{ problem / inclproblem }{
4478   % id      .str_set_x:N = \l__problems_inclprob_id_str,
4479   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4480   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4481   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4482   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
4483   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
4484 }
4485 \cs_new_protected:Nn \__problems_inclprob_args:n {
4486   % \str_clear:N \l__problems_prob_id_str
4487   \tl_clear:N \l__problems_inclprob_pts_tl
4488   \tl_clear:N \l__problems_inclprob_min_tl
4489   \tl_clear:N \l__problems_inclprob_title_tl
4490   \int_zero_new:N \l__problems_inclprob_refnum_int
4491   \str_clear:N \l__problems_inclprob_mhrepos_str
4492   \keys_set:nn { problem / inclproblem }{ #1 }
4493   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
4494     \let\l__problems_inclprob_pts_tl\undefined
4495   }
4496   \tl_if_empty:NT \l__problems_inclprob_min_tl {
4497     \let\l__problems_inclprob_min_tl\undefined
4498   }
4499   \tl_if_empty:NT \l__problems_inclprob_title_tl {
4500     \let\l__problems_inclprob_title_tl\undefined
4501   }
4502   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
4503     \let\l__problems_inclprob_refnum_int\undefined
4504   }
4505 }
4506
4507 \cs_new_protected:Nn \__problems_inclprob_clear: {
4508   % \str_clear:N \l__problems_prob_id_str
4509   \let\l__problems_inclprob_pts_tl\undefined
4510   \let\l__problems_inclprob_min_tl\undefined

```

```

4511 \let\l__problems_inclprob_title_tl\undefined
4512 \let\l__problems_inclprob_refnum_int\undefined
4513 \let\l__problems_inclprob_mhrepos_str\undefined
4514 }
4515
4516 \newcommand\includeproblem[2][]{
4517   \__problems_inclprob_args:n{ #1 }
4518   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
4519     \input{#2}
4520   }{
4521     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
4522       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
4523     }
4524   }
4525   \__problems_inclprob_clear:
4526 }

```

(End definition for \includeproblem. This function is documented on page ??.)

31.5 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

4527 \AddToHook{enddocument}{
4528   \bool_if:NT \c__problems_pts_bool {
4529     \message{Total:~\arabic{pts}~points}
4530   }
4531   \bool_if:NT \c__problems_min_bool {
4532     \message{Total:~\arabic{min}~minutes}
4533   }
4534 }

```

The margin pars are reader-visible, so we need to translate

```

4535 \def\pts#1{
4536   \bool_if:NT \c__problems_pts_bool {
4537     \marginpar{#1~\prob@pt@kw}
4538   }
4539 }
4540 \def\min#1{
4541   \bool_if:NT \c__problems_min_bool {
4542     \marginpar{#1~\prob@min@kw}
4543   }
4544 }

```

\show@pts The **\show@pts** shows the points: if no points are given from the outside and also no points are given locally do nothing, else show and add. If there are outside points then we show them in the margin.

```

4545 \newcounter{pts}
4546 \def\show@pts{
4547   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4548     \bool_if:NT \c__problems_pts_bool {
4549       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
4550       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

4551     }
4552   }{
4553     \tl_if_exist:NT \l__problems_prob_pts_tl {
4554       \bool_if:NT \c__problems_pts_bool {
4555         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
4556         \addtocounter{pts}{\l__problems_prob_pts_tl}
4557       }
4558     }
4559   }
4560 }

```

(End definition for \show@pts. This function is documented on page ??.)
and now the same for the minutes

\show@min

```

4561 \newcounter{min}
4562 \def\show@min{
4563   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4564     \bool_if:NT \c__problems_min_bool {
4565       \marginpar{\l__problems_inclprob_pts_tl;min}
4566       \addtocounter{min}{\l__problems_inclprob_min_tl}
4567     }
4568   }{
4569     \tl_if_exist:NT \l__problems_prob_min_tl {
4570       \bool_if:NT \c__problems_min_bool {
4571         \marginpar{\l__problems_prob_min_tl;min}
4572         \addtocounter{min}{\l__problems_prob_min_tl}
4573       }
4574     }
4575   }
4576 }
4577 \</package>

```

(End definition for \show@min. This function is documented on page ??.)

Chapter 32

Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

32.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
4578 \@@=hwexam>
4579 \*cls>
4580 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
4581 \RequirePackage{l3keys2e,expl-keystr-compatible}
4582 \DeclareOption*{
4583   \PassOptionsToClass{\CurrentOption}{omdoc}
4584   \PassOptionsToPackage{\CurrentOption}{stex}
4585   \PassOptionsToPackage{\CurrentOption}{hwexam}
4586   \PassOptionsToPackage{\CurrentOption}{tikzinput}
4587 }
4588 \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the L^AT_EXML bindings, we make sure the right packages are loaded.

```
4589 \LoadClass{omdoc}
4590 \RequirePackage{stex}
4591 \RequirePackage{hwexam}
4592 \RequirePackage{tikzinput}
4593 \RequirePackage{graphicx}
4594 \RequirePackage{a4wide}
4595 \RequirePackage{amssymb}
4596 \RequirePackage{amstext}
4597 \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```
4598 \newcommand\assig@default@type{\hwexam@assignment@kw}  
4599 \addmetakey[\assig@default@type]{document}{hwexamtype}  
4600 \def\document@hwexamtype{\assig@default@type}  
4601 \</cls>
```

Chapter 33

Implementation: The hwexam Package

33.1 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.

```
4602 \*package>
4603 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
4604 \RequirePackage{l3keys2e,expl-keystr-compat}
4605
4606 \newif\iftest\testfalse
4607 \DeclareOption{test}{\testtrue}
4608 \newif\ifmultiple\multiplefalse
4609 \DeclareOption{multiple}{\multipletrue}
4610 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
4611 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
4612 \RequirePackage{keyval}[1997/11/10]
4613 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
4614 \newcommand\hwexam@assignment@kw{Assignment}
4615 \newcommand\hwexam@given@kw{Given}
4616 \newcommand\hwexam@due@kw{Due}
4617 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
4618 space}%
4619 \newcommand\correction@probs@kw{prob.}%
4620 \newcommand\correction@pts@kw{total}%
4621 \newcommand\correction@reached@kw{reached}%
4622 \newcommand\correction@sum@kw{Sum}%
4623 \newcommand\correction@grade@kw{grade}%
4624 \newcommand\correction@forgrading@kw{To be used for grading, do not write here}
```

(End definition for \hwexam@*kw. This function is documented on page ??.)

For the other languages, we set up triggers

```

4625 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}
4626
4627 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4628 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4629   \input{hwexam-ngerman.ldf}
4630 }
4631 \clist_if_in:NnT \l_tmpa_clist {finnish}{
4632   \input{hwexam-finnish.ldf}
4633 }
4634 \clist_if_in:NnT \l_tmpa_clist {french}{
4635   \input{hwexam-french.ldf}
4636 }
4637 \clist_if_in:NnT \l_tmpa_clist {russian}{
4638   \input{hwexam-russian.ldf}
4639 }

```

33.2 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.

```

4640 \newcounter{assignment}
4641 \numberproblemsin{assignment}
4642 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

We will prepare the keyval support for the `assignment` environment.

```

4643 \keys_define:nn { hwexam / assignment } {
4644   id .str_set:N = \l__hwexam_assign_id_str,
4645   number .int_set:N = \l__hwexam_assign_number_int,
4646   title .tl_set:N = \l__hwexam_assign_title_tl,
4647   type .tl_set:N = \l__hwexam_assign_type_tl,
4648   given .tl_set:N = \l__hwexam_assign_given_tl,
4649   due .tl_set:N = \l__hwexam_assign_due_tl,
4650   loadmodules .code:n = {
4651     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
4652   }
4653 }
4654 \cs_new_protected:Nn \__hwexam_assignment_args:n {
4655   \str_clear:N \l__hwexam_assign_id_str
4656   \int_set:Nn \l__hwexam_assign_number_int {-1}
4657   \tl_clear:N \l__hwexam_assign_title_tl
4658   \tl_clear:N \l__hwexam_assign_type_tl
4659   \tl_clear:N \l__hwexam_assign_given_tl
4660   \tl_clear:N \l__hwexam_assign_due_tl
4661   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
4662   \keys_set:nn { hwexam / assignment }{ #1 }
4663 }

```

The next three macros are intermediate functions that handle the case gracefully, where the respective token registers are undefined.

The `\given@due` macro prints information about the given and due status of the assignment. Its arguments specify the brackets.

```

4664 \newcommand\given@due[2]{
4665 \bool_lazy_all:nF {
4666 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
4667 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
4668 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
4669 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
4670 }{ #1 }
4671
4672 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
4673 \tl_if_empty:NF \l__hwexam_assign_given_tl {
4674 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
4675 }
4676 }{
4677 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
4678 }
4679
4680 \bool_lazy_or:nnF {
4681 \bool_lazy_and_p:nn {
4682 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
4683 }{
4684 \tl_if_empty_p:V \l__hwexam_assign_due_tl
4685 }
4686 }{
4687 \bool_lazy_and_p:nn {
4688 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
4689 }{
4690 \tl_if_empty_p:V \l__hwexam_assign_due_tl
4691 }
4692 }{ ,~ }
4693
4694 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
4695 \tl_if_empty:NF \l__hwexam_assign_due_tl {
4696 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
4697 }
4698 }{
4699 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
4700 }
4701
4702 \bool_lazy_all:nF {
4703 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
4704 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
4705 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
4706 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
4707 }{ #2 }
4708 }

```

`\assignment@title` This macro prints the title of an assignment, the local title is overwritten, if there is one from the `\inputassignment`. `\assignment@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```

4709 \newcommand\assignment@title[3]{

```

```

4710 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
4711 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
4712 #1
4713 }{
4714 #2\l__hwexam_assign_title_tl#3
4715 }
4716 }{
4717 #2\l__hwexam_inclassassign_title_tl#3
4718 }
4719 }

```

(End definition for \assignment@title. This function is documented on page ??.)

\assignment@number Like \assignment@title only for the number, and no around part.

```

4720 \newcommand\assignment@number{
4721 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
4722 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
4723 \int_use:N \l__hwexam_assign_number_int
4724 }
4725 }{
4726 \int_use:N \l__hwexam_inclassassign_number_int
4727 }
4728 }

```

(End definition for \assignment@number. This function is documented on page ??.)

With them, we can define the central **assignment** environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

assignment For the assignment environment we delegate the work to the @assignment environment that depends on whether multiple option is given.

```

4729 \newenvironment{assignment}[1][ ]{
4730 \__hwexam_assignment_args:n { #1 }
4731 %\sref@target
4732 \let\__hwexamnum\l__hwexam_assign_number_int
4733 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
4734 \stepcounter{assignment}
4735 }{
4736 \setcounter{assignment}{\int_use:N\__hwexamnum}
4737 }
4738 \setcounter{problem}{0}
4739 \def\current@section@level{\document@hwexamtype}
4740 %\sref@label@id{\document@hwexamtype \thesection}
4741 \begin{@assignment}
4742 }{
4743 \end{@assignment}
4744 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

4745 \def\__hwexasstitle{
4746 \protect\document@hwexamtype~\arabic{assignment}
4747 \assignment@title{}\;{} \; -- \given@due{}\}
4748 }

```

```

4749 \ifmultiple
4750 \newenvironment{@assignment}{
4751 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
4752 \begin{omgroup}[loadmodules]{\__hwexasstitle}
4753 }{
4754 \begin{omgroup}{\__hwexasstitle}
4755 }
4756 }{
4757 \end{omgroup}
4758 }

```

for the single-page case we make a title block from the same components.

```

4759 \else
4760 \newenvironment{@assignment}{
4761 \begin{center}\bf
4762 \Large\@title\strut\
4763 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:};{\}\}
4764 \large\given@due{--\;}{\;}{--}
4765 \end{center}
4766 }{}
4767 \fi% multiple

```

33.3 Including Assignments

\in*assignment This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

4768 \keys_define:nn { hwexam / inclassignment } {
4769 %id .str_set_x:N = \l__hwexam_assign_id_str,
4770 number .int_set:N = \l__hwexam_inclassign_number_int,
4771 title .tl_set:N = \l__hwexam_inclassign_title_tl,
4772 type .tl_set:N = \l__hwexam_inclassign_type_tl,
4773 given .tl_set:N = \l__hwexam_inclassign_given_tl,
4774 due .tl_set:N = \l__hwexam_inclassign_due_tl,
4775 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
4776 }
4777 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
4778 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
4779 \tl_clear:N \l__hwexam_inclassign_title_tl
4780 \tl_clear:N \l__hwexam_inclassign_type_tl
4781 \tl_clear:N \l__hwexam_inclassign_given_tl
4782 \tl_clear:N \l__hwexam_inclassign_due_tl
4783 \str_clear:N \l__hwexam_inclassign_mhrepos_str
4784 \keys_set:nn { hwexam / inclassignment }{ #1 }
4785 }
4786 \__hwexam_inclassignment_args:n {}
4787
4788 \newcommand\inputassignment[2][ ]{
4789 \__hwexam_inclassignment_args:n { #1 }
4790 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
4791 \input{#2}
4792 }{
4793 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

4794 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
4795 }
4796 }
4797 \__hwexam_inclasssign_assignment_args:n {}
4798 }
4799 \newcommand\includeassignment[2][ ]{
4800 \newpage
4801 \inputassignment[#1]{#2}
4802 }

```

(End definition for \in*assignment. This function is documented on page ??.)

33.4 Typesetting Exams

\quizheading

```

4803 \newcommand\quizheading[1]{
4804 \def\@tas{#1}
4805 \large\noindent NAME:~\hspace{8cm} MAILBOX:~\hspace{2cm}
4806 \ifx\@tas\empty\else
4807 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\hspace{2cm}
4808 \fi
4809 }

```

(End definition for \quizheading. This function is documented on page ??.)

\testheading

```

4810 \keys_define:nn { hwexam / testheading } {
4811 min .tl_set:N = \l__hwexam_testheading_min_tl,
4812 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
4813 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
4814 }
4815 \cs_new_protected:Nn \__hwexam_testheading_args:n {
4816 \tl_clear:N \l__hwexam_testheading_min_tl
4817 \tl_clear:N \l__hwexam_testheading_duration_tl
4818 \tl_clear:N \l__hwexam_testheading_reqpts_tl
4819 \keys_set:nn { hwexam / testheading }{ #1 }
4820 }
4821 \newenvironment{testheading}[1][ ]{
4822 \__hwexam_testheading_args:n{ #1 }
4823 \noindent\large{Name:~\hspace{2cm}}
4824 Matriculation Number:\hspace*{2cm}\strut\\[1ex]
4825 \begin{center}
4826 \Large\textbf{\@title}\\[1ex]
4827 \large\@date\\[3ex]
4828 \end{center}
4829 \textbf{You~have~}
4830 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
4831 \l__hwexam_testheading_min_tl~minutes
4832 }{
4833 \l__hwexam_testheading_duration_tl
4834 }~
4835 (sharp)~for~the~test
4836 };\\

```

```

4837 Write~the~solutions~to~the~sheet.
4838 \par\noindent
4839 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
4840 \advance\check@time by -\theassignment@totalmin
4841 The~estimated~time~for~solving~this~exam~is~
4842 {\theassignment@totalmin}~minutes,~
4843 leaving~you~{\the\check@time}~minutes~for~revising~
4844 your~exam.
4845
4846 \par\noindent
4847 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
4848 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
4849 You~can~reach~{\theassignment@totalpts}~points~if~you~
4850 solve~all~problems.~You~will~only~need~
4851 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
4852 i.e.~\ {\the\bonus@pts}~points~are~bonus~points.
4853 \vfill
4854 \begin{center}
4855 {
4856 \Large\em You~have~ample~time,~so~take~it~slow~
4857 and~avoid~rushing~to~mistakes!\}[2ex]
4858 Different~problems~test~different~skills~and~
4859 knowledge,~so~do~not~get~stuck~on~one~problem.
4860 }
4861 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
4862 \end{center}
4863 }{
4864 \newpage
4865 }

```

(End definition for \testheading. This function is documented on page ??.)

\testspace

```

4866 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

(End definition for \testspace. This function is documented on page ??.)

\testnewpage

```

4867 \newcommand\testnewpage{\iftest\newpage\fi}

```

(End definition for \testnewpage. This function is documented on page ??.)

\testemptypage

```

4868 \newcommand\testemptypage[1][\iftest\begin{center}\hwexam@testemptypage@kw\end{center}\vfi

```

(End definition for \testemptypage. This function is documented on page ??.)

\@problem This macro acts on a problem's record in the *.aux file. Here we redefine it (it was defined to do nothing in problem.sty) to generate the correction table.

```

4869 <@=problems>
4870 \renewcommand\@problem[3]{
4871 \stepcounter{assignment@probs}
4872 \def\__problemspts{#2}
4873 \ifx\__problemspts\emptyelse
4874 \addtocounter{assignment@totalpts}{#2}

```

```

4875 \fi
4876 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
4877 \xdef\correction@probs{\correction@probs & #1}%
4878 \xdef\correction@pts{\correction@pts & #2}
4879 \xdef\correction@reached{\correction@reached &}
4880 }
4881 \@@=hwexam)

```

(End definition for \@problem. This function is documented on page ??.)

\correction@table This macro generates the correction table

```

4882 \newcounter{assignment@probs}
4883 \newcounter{assignment@totalpts}
4884 \newcounter{assignment@totalmin}
4885 \def\correction@probs{\correction@probs@kw}%
4886 \def\correction@pts{\correction@pts@kw}%
4887 \def\correction@reached{\correction@reached@kw}%
4888 \def\after@correction@table{}%
4889 \stepcounter{assignment@probs}
4890 \newcommand\correction@table{
4891 \resizebox{\textwidth}{!}{%
4892 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
4893 &\multicolumn{\theassignment@probs}{c|}|%|
4894 {\footnotesize\correction@forgrading@kw} &\\ \hline
4895 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
4896 \correction@pts & \theassignment@totalpts & \\ \hline
4897 \correction@reached & & \[.7cm]\hline
4898 \end{tabular}}
4899 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
4900 \end{package}

```

(End definition for \correction@table. This function is documented on page ??.)

33.5 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}

```