

# The $\text{\TeX}$ 3 Package \*

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

2021-12-09

## **Abstract**

TODO

---

\*Version 3.0 (last revised 2021-12-09)

# Contents

<b>I</b>	<b>Manual</b>	<b>1</b>
<b>1</b>	<b>Stuff</b>	<b>2</b>
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
<b>II</b>	<b>Documentation</b>	<b>8</b>
<b>2</b>	<b>sTeX-Basics</b>	<b>9</b>
2.1	Macros and Environments	9
<b>3</b>	<b>sTeX-MathHub</b>	<b>11</b>
3.1	Macros and Environments	11
3.1.1	Files, Paths, URIs	11
3.1.2	MathHub Archives	12
<b>4</b>	<b>sTeX-References</b>	<b>14</b>
4.1	Macros and Environments	14
<b>5</b>	<b>sTeX-Modules</b>	<b>15</b>
5.1	Macros and Environments	15
5.1.1	The module-environment	17
<b>6</b>	<b>sTeX-Module Inheritance</b>	<b>20</b>
6.1	Macros and Environments	20
6.1.1	SMS Mode	20
6.1.2	Imports and Inheritance	21
<b>7</b>	<b>sTeX-Symbols</b>	<b>24</b>
7.1	Macros and Environments	24
<b>8</b>	<b>sTeX-Terms</b>	<b>27</b>
8.1	Macros and Environments	27
<b>9</b>	<b>sTeX-Structural Features</b>	<b>30</b>
9.1	Macros and Environments	30
	Structures	30
<b>10</b>	<b>sTeX-Metatheory</b>	<b>32</b>
10.1	Symbols	32
<b>III</b>	<b>Extensions</b>	<b>33</b>

<b>11 Tikzinput</b>	<b>34</b>
11.1 Macros and Environments . . . . .	34
<b>12 document-structure.sty: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>35</b>
12.1 Introduction . . . . .	35
12.2 The User Interface . . . . .	36
12.2.1 Package and Class Options . . . . .	36
12.2.2 Document Structure . . . . .	36
12.2.3 Ignoring Inputs . . . . .	37
12.2.4 Structure Sharing . . . . .	38
12.2.5 Global Variables . . . . .	38
12.2.6 Colors . . . . .	39
12.3 Limitations . . . . .	39
<b>13 Slides and Course Notes</b>	<b>40</b>
13.1 Introduction . . . . .	40
13.2 The User Interface . . . . .	40
13.2.1 Package Options . . . . .	40
13.2.2 Notes and Slides . . . . .	41
13.2.3 Header and Footer Lines of the Slides . . . . .	42
13.2.4 Frame Images . . . . .	42
13.2.5 Colors and Highlighting . . . . .	43
13.2.6 Front Matter, Titles, etc. . . . .	43
13.2.7 Excursions . . . . .	43
13.2.8 Miscellaneous . . . . .	43
13.3 Limitations . . . . .	43
<b>IV Implementation</b>	<b>44</b>
<b>14 S<sub>T</sub>E<sub>X</sub>-Basics Implementation</b>	<b>45</b>
14.1 The S <sub>T</sub> E <sub>X</sub> Document Class . . . . .	45
14.2 Preliminaries . . . . .	45
14.3 Messages and logging . . . . .	46
14.4 Persistence . . . . .	47
14.5 HTML Annotations . . . . .	48
14.6 Languages . . . . .	51
14.7 Activating/Deactivating Macros . . . . .	51
<b>15 S<sub>T</sub>E<sub>X</sub>-MathHub Implementation</b>	<b>53</b>
15.1 Generic Path Handling . . . . .	53
15.2 PWD and kpsewhich . . . . .	55
15.3 File Hooks and Tracking . . . . .	56
15.4 MathHub Repositories . . . . .	57
<b>16 S<sub>T</sub>E<sub>X</sub>-References Implementation</b>	<b>63</b>
16.1 Document URIs and URLs . . . . .	63
16.2 Setting Reference Targets . . . . .	65

<b>17</b>	<b>STeX-Modules Implementation</b>	<b>66</b>
17.1	The module environment . . . . .	69
17.2	Invoking modules . . . . .	74
<b>18</b>	<b>STeX-Module Inheritance Implementation</b>	<b>76</b>
18.1	SMS Mode . . . . .	76
18.2	Inheritance . . . . .	80
<b>19</b>	<b>STeX-Symbols Implementation</b>	<b>85</b>
19.1	Symbol Declarations . . . . .	85
19.2	Notations . . . . .	91
<b>20</b>	<b>STeX-Terms Implementation</b>	<b>99</b>
20.1	Symbol Invocations . . . . .	99
20.2	Terms . . . . .	101
20.3	Notation Components . . . . .	107
<b>21</b>	<b>STeX-Structural Features Implementation</b>	<b>110</b>
<b>22</b>	<b>STeX-Others Implementation</b>	<b>119</b>
<b>23</b>	<b>STeX-Metatheory Implementation</b>	<b>120</b>
<b>24</b>	<b>Tikzinput Implementation</b>	<b>123</b>
<b>25</b>	<b>document-structure.sty Implementation</b>	<b>125</b>
25.1	The OMDoc Class . . . . .	125
25.2	Class Options . . . . .	125
25.3	Beefing up the document environment . . . . .	126
25.4	Implementation: OMDoc Package . . . . .	126
25.5	Package Options . . . . .	126
25.6	Document Structure . . . . .	127
25.7	Front and Backmatter . . . . .	130
25.8	Ignoring Inputs . . . . .	131
25.9	Structure Sharing . . . . .	131
25.10	Global Variables . . . . .	132
25.11	Colors . . . . .	132
<b>26</b>	<b>MiKoSlides – Implementation</b>	<b>133</b>
26.1	Class and Package Options . . . . .	133
26.2	Notes and Slides . . . . .	135
26.3	Header and Footer Lines . . . . .	137
26.4	Colors and Highlighting . . . . .	139
26.5	Sectioning . . . . .	140
26.6	Excursions . . . . .	142
26.7	Miscellaneous . . . . .	143

**Part I**  
**Manual**

# Chapter 1

## Stuff

### 1.1 Modules

---

`\sTeX`  
`\stex`

---

Both print this  $\text{\TeX}$  logo.

#### 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 changed<sup>1</sup>.

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  $x \in A$

<sup>1</sup>EdNOTE: 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  $\textcolor{teal}{\$}\textcolor{teal}{\text{mult}}\textcolor{teal}{*}![\textcolor{teal}{\$}\textcolor{teal}{\text{comp}}\textcolor{teal}{\text{cdot}}\textcolor{teal}{\$}$ ) 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,  $\text{\TeX}$  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  $\text{\TeX}$ , but are treated very differently in OMDOC and by MMT. More interesting *within*  $\text{\TeX}$  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

<sup>2</sup>EDNOTE: what about e.g. `\int \_x \int \_y \int \_z f dx dy dz`?

<sup>3</sup>EDNOTE: “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 suffix<sup>1</sup>.

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.

---

<sup>1</sup>which 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 {<math>\langle log-prefix \rangle</math>} {<math>\langle message \rangle</math>}</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 <sup>A</sup> T <sub>E</sub> X2e and L <sup>A</sup> T <sub>E</sub> X3 conditionals for L <sup>A</sup> T <sub>E</sub> 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<sup>A</sup>T<sub>E</sub>XML or S<sup>C</sup>A<sup>L</sup>L<sup>A</sup>T<sub>E</sub>X) 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:N</code>	$\underline{TF}$ $\star$

---

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
<code>\c_stex_pwd_str</code>	(heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</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{&lt;repository-name&gt;}{&lt;code&gt;}</code> Change the current repository to <code>{&lt;repository-name&gt;}</code> (or not, if <code>{&lt;repository-name&gt;}</code> is empty), and passes its ID on to <code>{&lt;code&gt;}</code> as #1. Switches back to the previous repository after executing <code>{&lt;code&gt;}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{&lt;archive-ID&gt;}{&lt;filename&gt;}</code> Expands to the full path of file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\inputref:nn</code>	<code>\inputref[&lt;archive-ID&gt;]{&lt;filename&gt;}</code> <code>\inputs</code> the file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{&lt;filename&gt;}</code> Inputs <code>&lt;filename&gt;.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://Users/kohlhase/localmh/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.

---

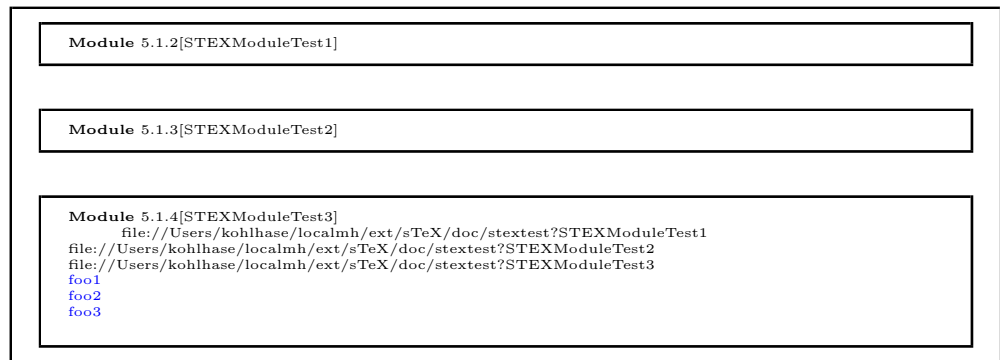
\stex\_invoke\_module:n

---

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<sub>E</sub>X 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:  $\rightarrow$  `\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?Foo?foo}<`

Meaning:  $\rightarrow$  `\protect \bar <`

**Module 6.1.2[Importtest]**

Meaning:  $\rightarrow$  `\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?Foo?foo}<`

**Module 6.1.3[Importtest2]**

Meaning:  $\rightarrow$  `\stex_invoke_symbol:n {file://Users/kohlhase/localmh/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://Users/kohlhase/localmh/ext/sTeX/doc/stextest?UseTest1?foo}<`

**Module 6.1.6[UseTest3]**  
Meaning: `\undefined<`  
Meaning: `\macro:~>\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?UseTest2?bar}<`

All modules: <http://mathhub.info/sTeX?Metatheory>, <file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?UseTest3>,  
<file://Users/kohlhase/localmh/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?sequence-index>, <http://mathhub.info/sTeX?Metatheory?seqtype>,  
<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://Users/kohlhase/localmh/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://Users/kohlhase/localmh/ext/sTeX/doc/stextest?SymdeclTest?foo}<
Result: file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://Users/kohlhase/localmh/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
```

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{&lt;symbol&gt;}{&lt;text&gt;}</code> shortcut for <code>\STEXsymbol{&lt;symbol&gt;}! [ &lt;text&gt; ]</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>&lt;URI&gt;&lt;fragment&gt;&lt;precedence&gt;&lt;body&gt;</code> Annotates <code>&lt;body&gt;</code> as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol <code>&lt;URI&gt;</code> , generated by the specific notation <code>&lt;fragment&gt;</code> with (upwards) operator precedence <code>&lt;precedence&gt;</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&lt;int&gt;&lt;prec&gt;&lt;body&gt;</code> Annotates <code>&lt;body&gt;</code> as the <code>&lt;int&gt;</code> th argument of the current OMA or OMBIND, with (downwards) argument precedence <code>&lt;prec&gt;</code> .
<hr/> <hr/> <code>\_stex_term_math_assoc_arg:nnnn</code>	<code>\stex_term_arg:nnn&lt;int&gt;&lt;prec&gt;&lt;notation&gt;&lt;body&gt;</code> Annotates <code>&lt;body&gt;</code> as the <code>&lt;int&gt;</code> th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence <code>&lt;prec&gt;</code> and associative notation <code>&lt;notation&gt;</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 $\text{\S I E X}$ 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 $\text{\S I E X}$ 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 } }
 $\bar{foo} a\{b,c,d,e,f\}g$  and  $\bar{foo} a\{b,c\}g$  and  $\bar{foo} abc$ 
\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  $\mult{a,\plus{\frac{ab}{c}}}$ 
 $\displaystyle \plus{a,\mult{b,c}}$  and  $\mult{a,\plus{\frac{ab}{c}}}$ 
\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  $a\cdot\frac{a}{b}+\frac{a}{c}$   
 $a+(b\cdot c)$  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>}`

`\@comp`

`\@defemph`

---

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

`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).

#### Structures

`structure`    `TODO`

#### Test 17

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

\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: $\mM{op}ab$

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

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



## Chapter 10

# 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  $\Pi$  in dependent type theories.

### 10.1 Symbols

**Part III**  
**Extensions**

## Chapter 11

# Tikzinput

### 11.1 Macros and Environments

## Chapter 12

# document-structure.sty: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X

The `omdoc` package is part of the  $\text{\LaTeX}$  collection, a version of  $\text{\TeX}/\text{\LaTeX}$  that allows to markup  $\text{\TeX}/\text{\LaTeX}$  documents semantically without leaving the document format, essentially turning  $\text{\TeX}/\text{\LaTeX}$  into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in  $\text{\LaTeX}$ . This includes a simple structure sharing mechanism for  $\text{\LaTeX}$  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  $\text{\LaTeX}$  sources, or after translation.

### 12.1 Introduction

$\text{\LaTeX}$  is a version of  $\text{\TeX}/\text{\LaTeX}$  that allows to markup  $\text{\TeX}/\text{\LaTeX}$  documents semantically without leaving the document format, essentially turning  $\text{\TeX}/\text{\LaTeX}$  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  $\text{\LaTeX}$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\text{\LaTeX}$  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

source and the formatter does the copying during document formatting/presentation.<sup>4</sup>

## 12.2 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`.

### 12.2.1 Package and Class Options

The `omdoc` class accept the following options:

<code>class=&lt;name&gt;</code>	load <code>&lt;name&gt;.cls</code> instead of <code>article.cls</code>
<code>topsect=&lt;sect&gt;</code>	The top-level sectioning level; the default for <code>&lt;sect&gt;</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 $\text{\TeX}$ packages

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

### 12.2.2 Document Structure

`document` The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble<sup>2</sup>. 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 L<sup>A</sup>T<sub>E</sub>XML transformation.

`omgroup` The structure of the document is given by the `omgroup` environment just like in OMDoc. In the L<sup>A</sup>T<sub>E</sub>X 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.barderv,loadmodules]{Introducing $\protect\bar$ Derivations}
```

`blindomgroup` L<sup>A</sup>T<sub>E</sub>X 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`

<sup>4</sup>EdNOTE: integrate with latexml's XMRef in the Math mode.

<sup>2</sup>We 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 frontmatter<sup>3</sup> 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.

### 12.2.3 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

<sup>3</sup>We 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  $\text{\LaTeX}$  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,  $\text{\LaTeX}$  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].

## 12.2.4 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  $\text{\LaTeX}$ 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  $\text{\LaTeX}$ . 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.<sup>5</sup>

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

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

<sup>5</sup>EdNOTE: document LMID und LMXRef here if we decide to keep them.

### 12.2.6 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.

## 12.3 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 13

## Slides and Course Notes

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

### 13.1 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`.

### 13.2 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  $\text{\LaTeX}$  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.

#### 13.2.1 Package Options

The `mikoslides` class takes a variety of class options:<sup>6</sup>

- |                           |  |
|---------------------------|--|
| <code>slides</code>       | <ul style="list-style-type: none"><li>• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 13.2.2).</li></ul>   |
| <code>notes</code>        |  |
| <code>sectocframes</code> | <ul style="list-style-type: none"><li>• 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.</li></ul> |

<code>showmeta</code>	<ul style="list-style-type: none"> <li>• <code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).</li> </ul>
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> <li>• If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 13.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.</li> </ul>
<code>topsect</code>	<ul style="list-style-type: none"> <li>• <code>topsect=&lt;sect&gt;</code> can be used to specify the top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>.</li> </ul>

### 13.2.2 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.<sup>4</sup>

⚠ 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<sup>A</sup>T<sub>E</sub>X 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.

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

<sup>4</sup>MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L<sup>A</sup>T<sub>E</sub>X 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.

### 13.2.3 Header and Footer Lines of the Slides

The default logo provided by the `mikoslides` package is the  $\text{\LaTeX}$  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.

### 13.2.4 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  $\text{\LaTeX}$  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.<sup>7</sup>

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

<sup>7</sup>EdNOTE: 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}`

## 13.2.5 Colors and Highlighting

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

## 13.2.6 Front Matter, Titles, etc.

## 13.2.7 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}
```

## 13.2.8 Miscellaneous

## 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 [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.

Part IV

# Implementation



## Chapter 14

# STEX -Basics Implementation

### 14.1 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>
```

### 14.2 Preliminaries

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

```

26 sms .bool_set:N = \c_stex_persist_mode_bool ,
27 image .bool_set:N = \c_tikzinput_image_bool
28 }
29 \ProcessKeysOptions { stex }

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

```

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

Patching expl3, if outdated:

```

37 <@@=keys>
38 \cs_if_exist:cF { \c__keys_props_root_str .str_set:N }{
39 \cs_new_protected:cpn { \c__keys_props_root_str .str_set:N } #1
40 { \__keys_variable_set:NnnN #1 { str } { } n }
41 \cs_new_protected:cpn { \c__keys_props_root_str .str_set:c } #1
42 { \__keys_variable_set:cnnN {#1} { str } { } n }
43 \cs_new_protected:cpn { \c__keys_props_root_str .str_set_x:N } #1
44 { \__keys_variable_set:NnnN #1 { str } { } x }
45 \cs_new_protected:cpn { \c__keys_props_root_str .str_set_x:c } #1
46 { \__keys_variable_set:cnnN {#1} { str } { } x }
47 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset:N } #1
48 { \__keys_variable_set:NnnN #1 { str } { g } n }
49 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset:c } #1
50 { \__keys_variable_set:cnnN {#1} { str } { g } n }
51 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset_x:N } #1
52 { \__keys_variable_set:NnnN #1 { str } { g } x }
53 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset_x:c } #1
54 { \__keys_variable_set:cnnN {#1} { str } { g } x }
55 }

```

## 14.3 Messages and logging

```

56 <@@=stex_log>
57 Warnings and error messages
58 \msg_new:nnn{stex}{error/unknownlanguage}{
59   Unknown~language:~#1
60 }
61 \msg_new:nnn{stex}{warning/nomathhub}{
62   MATHHUB~system~variable~not~found~and~no~
63   \detokenize{\mathhub}~value~set!
64 }
65 \msg_new:nnn{stex}{error/deactivated-macro}{
66   The~\detokenize{#1}~command~is~only~allowed~in~#2!
67 }

```

\stex\_debug:nn A simple macro issuing package messages with subpath.

```

67 \cs_new_protected:Nn \stex_debug:nn {

```

```

68 \clist_if_in:NnTF \c_stex_debug_clist { all } {
69   \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
70     \\Debug~#1:~#2\\
71   }
72   \msg_none:nn{stex}{debug / #1}
73 }{
74   \clist_if_in:NnT \c_stex_debug_clist { #1 } {
75     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
76       \\Debug~#1:~#2\\
77     }
78     \msg_none:nn{stex}{debug / #1}
79   }
80 }
81 }

```

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

Redirecting messages:

```

82 \clist_if_in:NnTF \c_stex_debug_clist {all} {
83   \msg_redirect_module:nnn{ stex }{ none }{ term }
84 }{
85   \clist_map_inline:Nn \c_stex_debug_clist {
86     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
87   }
88 }
89
90 \stex_debug:nn{log}{debug~mode~on}

```

## 14.4 Persistence

```

91 <@@=stex_persist>

```

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

```

92 \iow_new:N \c__stex_persist_sms_iow
93 \AddToHook{begindocument}{
94   \bool_if:NnTF \c_stex_persist_mode_bool {
95     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
96   } {
97     \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
98   }
99 }
100 \AddToHook{enddocument}{
101   \bool_if:NnNF \c_stex_persist_mode_bool {
102     \iow_close:N \c__stex_persist_sms_iow
103   }
104 }

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

105 \cs_new_protected:Nn \stex_add_to_sms:n {
106   \bool_if:NnNF \c_stex_persist_mode_bool {
107     \iow_now:Nn \c__stex_persist_sms_iow { #1 }
108   }
109 }

```

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

## 14.5 HTML Annotations

```

110 <@@=stex_annotate>
111 \RequirePackage{scalatex}

```

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

```

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

```

```

\if@latexml
\latexml_if_p:
\latexml_if:TF

```

Conditionals for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L:

```

113 \ifcsname if@latexml\endcsname\else
114   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
115 \fi
116
117 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
118   \if@latexml
119     \prg_return_true:
120   \else:
121     \prg_return_false:
122 \fi:
123 }

```

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

```

\l__stex_annotate_arg_tl
\c__stex_annotate_emptyarg_tl

```

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

```

124 \tl_new:N \l__stex_annotate_arg_tl
125 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
126   \scalatex_if:TF {
127     \scalatex_direct_HTML:n { \c_ampsand_str lrm; }
128   }{-}
129 }

```

(End definition for `\l__stex_annotate_arg_tl` and `\c__stex_annotate_emptyarg_tl`.)

```

\__stex_annotate_checkempty:n

```

```

130 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
131   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
132   \tl_if_empty:NT \l__stex_annotate_arg_tl {
133     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
134   }
135 }

```

(End definition for `\__stex_annotate_checkempty:n`.)

```

\l_stex_html_do_output_bool
\stex_if_do_html:

```

Whether to (locally) produce HTML output

```

136 \bool_new:N \l_stex_html_do_output_bool
137 \bool_set_true:N \l_stex_html_do_output_bool
138 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
139   \bool_if:nTF \l_stex_html_do_output_bool
140     \prg_return_true: \prg_return_false:
141 }

```

(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

```

142 \cs_new_protected:Nn \stex_suppress_html:n {
143   \exp_args:Nne \use:nn {
144     \bool_set_false:N \l_stex_html_do_output_bool
145     #1
146   }{
147     \stex_if_do_html:T {
148       \bool_set_true:N \l_stex_html_do_output_bool
149     }
150   }
151 }
```

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

`\stex_annotate:env`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L<sup>A</sup>T<sub>E</sub>XML, S<sub>C</sub>A<sub>L</sub>T<sub>E</sub>X, p<sub>D</sub>F<sub>L</sub>A<sub>T</sub>E<sub>X</sub>).

`\stex_annotate_invisible:n`

The p<sub>D</sub>F<sub>L</sub>A<sub>T</sub>E<sub>X</sub>-macros largely do nothing; the S<sub>C</sub>A<sub>L</sub>T<sub>E</sub>X-implementations are pretty clear in what they do, the L<sup>A</sup>T<sub>E</sub>XML-implementations resort to perl bindings.

`\stex_annotate_invisible:nnn`

```

152 \scalatex_if:TF{
153   \cs_new_protected:Nn \stex_annotate:nnn {
154     \__stex_annotate_checkempty:n { #3 }
155     \scalatex_annotate_HTML:nn {
156       property="stex:#1" ~
157       resource="#2"
158     } {
159       \tl_use:N \l__stex_annotate_arg_tl
160     }
161   }
162   \cs_new_protected:Nn \stex_annotate_invisible:n {
163     \__stex_annotate_checkempty:n { #1 }
164     \scalatex_annotate_HTML:nn {
165       stex:visible="false" ~
166       style:display="none"
167     } {
168       \tl_use:N \l__stex_annotate_arg_tl
169     }
170   }
171   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
172     \__stex_annotate_checkempty:n { #3 }
173     \scalatex_annotate_HTML:nn {
174       property="stex:#1" ~
175       resource="#2" ~
176       stex:visible="false" ~
177       style:display="none"
178     } {
179       \tl_use:N \l__stex_annotate_arg_tl
180     }
181   }
182   \NewDocumentEnvironment{stex_annotate_env} { m m } {
183     \par
184     \scalatex_annotate_HTML_begin:n {
```

```

185     property="stex:#1" ~
186     resource="#2"
187   }
188 }{
189   \scalatex_annotate_HTML_end:
190 }
191 }{
192   \latexml_if:TF {
193     \cs_new_protected:Nn \stex_annotate:nnn {
194       \__stex_annotate_checkempty:n { #3 }
195       \mode_if_math:TF {
196         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
197           \tl_use:N \l__stex_annotate_arg_tl
198         }
199       }{
200         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
201           \tl_use:N \l__stex_annotate_arg_tl
202         }
203       }
204     }
205     \cs_new_protected:Nn \stex_annotate_invisible:n {
206       \__stex_annotate_checkempty:n { #1 }
207       \mode_if_math:TF {
208         \cs:w latexml@invisible@math\cs_end:{
209           \tl_use:N \l__stex_annotate_arg_tl
210         }
211       } {
212         \cs:w latexml@invisible@text\cs_end:{
213           \tl_use:N \l__stex_annotate_arg_tl
214         }
215       }
216     }
217     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
218       \__stex_annotate_checkempty:n { #3 }
219       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
220         \tl_use:N \l__stex_annotate_arg_tl
221       }
222     }
223     \NewDocumentEnvironment{stex_annotate_env} { m m } {
224       \par\begin{latexml@annotateenv}{#1}{#2}
225     }{
226       \end{latexml@annotateenv}
227     }
228   }{
229     \cs_new_protected:Nn \stex_annotate:nnn {#3}
230     \cs_new_protected:Nn \stex_annotate_invisible:n {}
231     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
232     \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{
233   }
234 }

```

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

## 14.6 Languages

235 `<@=stex_language>`

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

236 \prop_const_from_keyval:Nn \c_stex_languages_prop {
237   en = english ,
238   de = ngerman ,
239   ar = arabic ,
240   bg = bulgarian ,
241   ru = russian ,
242   fi = finnish ,
243   ro = romanian ,
244   tr = turkish ,
245   fr = french
246 }
247
248 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
249   english   = en ,
250   ngerman   = de ,
251   arabic    = ar ,
252   bulgarian = bg ,
253   russian   = ru ,
254   finnish   = fi ,
255   romanian  = ro ,
256   turkish   = tr ,
257   french    = fr
258 }
259 % todo: chinese simplified (zhs)
260 %       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:

```

261 \clist_if_empty:NF \c_stex_languages_clist {
262   \clist_clear:N \l_tmpa_clist
263   \clist_map_inline:Nn \c_stex_languages_clist {
264     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
265       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
266     } {
267       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
268     }
269   }
270   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
271   \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
272 }

```

## 14.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

273 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
274   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
275   \def#1{

```

```

276     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
277   }
278 }

```

*(End definition for \stex\_deactivate\_macro:Nn. This function is documented on page 10.)*

**\stex\_reactivate\_macro:N**

```

279 \cs_new_protected:Nn \stex_reactivate_macro:N {
280   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
281 }

```

*(End definition for \stex\_reactivate\_macro:N. This function is documented on page 10.)*

```

282 \</package>

```



## Chapter 15

# STEX -MathHub Implementation

```
283 <*package>
284
285 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
286
287 <@@=stex_path>
288
289 Warnings and error messages
290 \msg_new:nnn{stex}{error/norepository}{
291   No~archive~#1~found~in~#2
292 }
293 \msg_new:nnn{stex}{error/notinarchive}{
294   Not~currently~in~an~archive,~but~\detokenize{#1}~
295   needs~one!
296 }
297 \msg_new:nnn{stex}{error/nofile}{
298   \detokenize{#1}~could~not~find~file~#2
299 }
```

### 15.1 Generic Path Handling

We treat paths as L<sup>A</sup>T<sub>E</sub>X3-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
298 \cs_new_protected:Nn \stex_path_from_string:Nn {
299   \str_set:Nx \l_tmpa_str { #2 }
300   \str_if_empty:NTF \l_tmpa_str {
301     \seq_clear:N #1
302   }{
303     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
304     \sys_if_platform_windows:T{
305       \seq_clear:N \l_tmpa_tl
306       \seq_map_inline:Nn #1 {
307         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
308         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
309       }
310     }
311   }
```

```

309     }
310     \seq_set_eq:NN #1 \l_tmpa_tl
311   }
312   \stex_path_canonicalize:N #1
313 }
314 }
315 \cs_generate_variant:Nn \stex_path_from_string:Nn
316 { 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
317 \cs_new_protected:Nn \stex_path_to_string:NN {
318   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
319 }
320
321 \cs_new:Nn \stex_path_to_string:N {
322   \seq_use:Nn #1 /
323 }

```

(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
324 \str_const:Nn \c__stex_path_dot_str {.}
325 \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.

```

326 \cs_new_protected:Nn \stex_path_canonicalize:N {
327   \seq_if_empty:NF #1 {
328     \seq_clear:N \l_tmpa_seq
329     \seq_get_left:NN #1 \l_tmpa_tl
330     \str_if_empty:NT \l_tmpa_tl {
331       \seq_put_right:Nn \l_tmpa_seq {}
332     }
333     \seq_map_inline:Nn #1 {
334       \str_set:Nn \l_tmpa_tl { ##1 }
335       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
336         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
337           \seq_if_empty:NTF \l_tmpa_seq {
338             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
339               \c__stex_path_up_str
340             }
341           }{
342             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
343             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
344               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
345                 \c__stex_path_up_str
346               }
347             }{
348               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
349             }

```

```

350     }
351   }{
352     \str_if_empty:NF \l_tmpa_tl {
353       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
354     }
355   }
356 }
357 }
358 \seq_gset_eq:NN #1 \l_tmpa_seq
359 }
360 }

```

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

`\stex_path_if_absolute_p:N`  
`\stex_path_if_absolute:NTF`

```

361 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
362   \seq_if_empty:NTF #1 {
363     \prg_return_false:
364   }{
365     \seq_get_left:NN #1 \l_tmpa_tl
366     \str_if_empty:NTF \l_tmpa_tl {
367       \prg_return_true:
368     }{
369       \prg_return_false:
370     }
371   }
372 }

```

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

## 15.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

373 \str_new:N\l_stex_kpsewhich_return_str
374 \cs_new_protected:Nn \stex_kpsewhich:n {
375   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
376   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
377   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
378 }

```

(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`

```

379 \sys_if_platform_windows:TF{
380   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
381 }{
382   \stex_kpsewhich:n{-var-value~PWD}
383 }
384
385 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
386 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
387 \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.)

## 15.3 File Hooks and Tracking

388 `<@@=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

389 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

391 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

392 `\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`.

393 `\seq_gclear_new:N\g_stex_currentfile_seq`

394 `\AddToHook{file/before}{`

395 `\stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}`

396 `\stex_path_if_absolute:NTF\g_stex_currentfile_seq{`

397 `\exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}`

398 `}{`

399 `\stex_path_from_string:Nn\g_stex_currentfile_seq{`

400 `\c_stex_pwd_str/\CurrentFilePath/\CurrentFile`

401 `}`

402 `}`

403 `\seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq`

404 `\exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq`

405 `}`

406 `\AddToHook{file/after}{`

407 `\seq_if_empty:NF\g__stex_files_stack{`

408 `\seq_gpop:Nn\g__stex_files_stack\l_tmpa_seq`

409 `}`

410 `\seq_if_empty:NTF\g__stex_files_stack{`

411 `\seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq`

412 `}{`

413 `\seq_get:NN\g__stex_files_stack\l_tmpa_seq`

414 `\seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq`

415 `}`

416 `}`

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

## 15.4 MathHub Repositories

```

417 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
418 \str_if_empty:NTF\mathhub{
419   \stex_kpsewhich:n{-var-value~MATHHUB}
420   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
421
422   \str_if_empty:NTF\c_stex_mathhub_str{
423     \msg_warning:nn{stex}{warning/nomathhub}
424   }{
425     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
426     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
427   }
428 }{
429   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
430   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
431     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
432       \c_stex_pwd_str/\mathhub
433     }
434   }
435   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
436   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
437 }

```

(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
438 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
439   \str_set:Nx \l_tmpa_str { #1 }
440   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
441     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
442     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
443     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
444     \__stex_mathhub_find_manifest:N \l_tmpa_seq
445     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
446       \msg_error:nnnn{stex}{error/norepository}{#1}{
447         \stex_path_to_string:N \c_stex_mathhub_str
448       }
449     } {
450       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
451     }
452   }
453 }

```

(End definition for `\__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
454 \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`:

```

455 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
456   \seq_set_eq:NN \l_tmpa_seq #1
457   \bool_set_true:N \l_tmpa_bool
458   \bool_while_do:Nn \l_tmpa_bool {
459     \seq_if_empty:NTF \l_tmpa_seq {
460       \bool_set_false:N \l_tmpa_bool
461     }{
462       \file_if_exist:nTF{
463         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
464       }{
465         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
466         \bool_set_false:N \l_tmpa_bool
467       }{
468         \file_if_exist:nTF{
469           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
470         }{
471           \seq_put_right:Nn \l_tmpa_seq{META-INF}
472           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
473           \bool_set_false:N \l_tmpa_bool
474         }{
475           \file_if_exist:nTF{
476             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
477           }{
478             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
479             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
480             \bool_set_false:N \l_tmpa_bool
481           }{
482             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
483           }
484         }
485       }
486     }
487   }
488   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
489 }

```

(End definition for `\_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

490 \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:

```

491 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
492   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
493   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
494   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
495     \str_set:Nn \l_tmpa_str {##1}
496     \exp_args:NNoo \seq_set_split:Nnn
497       \l_tmpb_seq \c_colon_str \l_tmpa_str
498     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

499 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
500   \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
501 }
502 \exp_args:No \str_case:nnTF \l_tmpa_tl {
503   {id} {
504     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
505     { id } \l_tmpb_tl
506   }
507   {narration-base} {
508     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
509     { narr } \l_tmpb_tl
510   }
511   {url-base} {
512     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
513     { docurl } \l_tmpb_tl
514   }
515   {source-base} {
516     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
517     { ns } \l_tmpb_tl
518   }
519   {ns} {
520     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
521     { ns } \l_tmpb_tl
522   }
523   {dependencies} {
524     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
525     { deps } \l_tmpb_tl
526   }
527   {}{}
528 }{}
529 }
530 \ior_close:N \c__stex_mathhub_manifest_ior
531 }

```

(End definition for `\__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

532 \cs_new_protected:Nn \stex_set_current_repository:n {
533   \stex_require_repository:n { #1 }
534   \prop_set_eq:Nc \l_stex_current_repository_prop {
535     c_stex_mathhub_#1_manifest_prop
536   }
537 }

```

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

`\stex_require_repository:n`

```

538 \cs_new_protected:Nn \stex_require_repository:n {
539   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
540     \stex_debug:nn{mathhub}{Opening~archive:~#1}
541     \__stex_mathhub_do_manifest:n { #1 }
542     \exp_args:Nx \stex_add_to_sms:n {
543       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
544         id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
545         ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

546     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
547     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
548   }
549 }
550 }
551 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

552 \prop_new:N \l_stex_current_repository_prop
553
554 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
555 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
556   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
557 } {
558   \__stex_mathhub_parse_manifest:n { main }
559   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
560   \l_tmpa_str
561   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
562   \c_stex_mathhub_main_manifest_prop
563   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
564   \stex_debug:nn{mathhub}{Current~repository:~
565     \prop_item:Nn \l_stex_current_repository_prop {id}
566   }
567 }

```

(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.

```

568 \cs_new_protected:Nn \stex_in_repository:nn {
569   \str_set:Nx \l_tmpa_str { #1 }
570   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
571   \str_if_empty:NTF \l_tmpa_str {
572     \exp_args:Ne \l_tmpa_cs{
573       \prop_item:Nn \l_stex_current_repository_prop { id }
574     }
575   }{
576     \stex_require_repository:n \l_tmpa_str
577     \str_set:Nx \l_tmpa_str { #1 }
578     \exp_args:Nne \use:nn {
579       \stex_set_current_repository:n \l_tmpa_str
580       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
581     }{
582       \stex_set_current_repository:n {
583         \prop_item:Nn \l_stex_current_repository_prop { id }
584       }
585     }
586   }
587 }

```

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



**\inputref**  
**\inputref:nn**

```

588 \newif \ifinputref \inputreffalse
589
590 \cs_new_protected:Nn \inputref:nn {
591   \stex_in_repository:nn {#1} {
592     \ifinputref
593       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
594     \else
595       \inputreftrue
596       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
597     \inputreffalse
598   \fi
599 }
600 }
601 \NewDocumentCommand \inputref { 0{} m}{
602   \inputref:nn{ #1 }{ #2 }
603 }

```

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

**\mhp**

```

604 \def \mhp #1 #2 {
605   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
606     \c_stex_mathhub_str /
607     \prop_item:Nn \l_stex_current_repository_prop { id }
608     / source / #2
609   }{
610     \c_stex_mathhub_str / #1 / source / #2
611   }
612 }

```

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

**\libinput**

```

613 \cs_new_protected:Npn \libinput #1 {
614   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
615     \msg_error:nnn{stex}{error/notinarchive}\libinput
616   }
617   \bool_set_false:N \l_tmpa_bool
618   \tl_clear:N \l_tmpa_tl
619   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
620   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
621   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
622   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
623     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
624     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
625       / meta-inf / lib / #1.tex}{
626       \bool_set_true:N \l_tmpa_bool
627       \tl_put_right:Nx \l_tmpa_tl {
628         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
629           / meta-inf / lib / #1.tex}
630       }
631     }{}
632   }

```

```

633 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
634 / \l_tmpa_str / lib / #1.tex
635 }{
636   \bool_set_true:N \l_tmpa_bool
637   \tl_put_right:Nx \l_tmpa_tl {
638     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
639       / \l_tmpa_str / lib / #1.tex}
640   }
641 }{}
642 \bool_if:NF \l_tmpa_bool {
643   \msg_error:nnnn{stex}{error/nofile}\libinput{#1.tex}
644 }
645 \l_tmpa_tl
646 }

```

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

```

647 </package>

```

## Chapter 16

# STEX -References Implementation

```
648 <*package>
649
650 %%%%%%%%%%% references.dtx %%%%%%%%%%%
651
652 %\RequirePackage{hyperref}
653 %\RequirePackage{cleveref}
654 <@@=stex_refs>
```

Warnings and error messages

```
655
```

### 16.1 Document URIs and URLs

```
656 \str_new:N \l_stex_current_docns_str
657 \cs_new_protected:Nn \stex_get_document_uri: {
658   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
659   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
660   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
661   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
662   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
663
664   \str_clear:N \l_tmpa_str
665   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
666     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
667   }
668
669   \str_if_empty:NTF \l_tmpa_str {
670     \str_set:Nx \l_stex_current_docns_str {
671       file:/\stex_path_to_string:N \l_tmpa_seq
672     }
673   }{
674     \bool_set_true:N \l_tmpa_bool
675     \bool_while_do:Nn \l_tmpa_bool {
676       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
677       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
678         {source} { \bool_set_false:N \l_tmpa_bool }
```

```

679     }{}{
680         \seq_if_empty:NT \l_tmpa_seq {
681             \bool_set_false:N \l_tmpa_bool
682         }
683     }
684 }
685
686 \seq_if_empty:NTF \l_tmpa_seq {
687     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
688 }{
689     \str_set:Nx \l_stex_current_docns_str {
690         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
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
716         \bool_while_do:Nn \l_tmpa_bool {
717             \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
718             \exp_args:No \str_case:nnTF { \l_tmpb_str } {
719                 {source} { \bool_set_false:N \l_tmpa_bool }
720             }{}{
721                 \seq_if_empty:NT \l_tmpa_seq {
722                     \bool_set_false:N \l_tmpa_bool
723                 }
724             }
725         }
726
727         \seq_if_empty:NTF \l_tmpa_seq {
728             \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
729         }{
730             \str_set:Nx \l_stex_current_docurl_str {
731                 \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
732             }

```

```

733     }
734   }
735 }

```

## 16.2 Setting Reference Targets

```

736 \str_const:Nn \c__stex_refs_url_str{URL}
737 \str_const:Nn \c__stex_refs_ref_str{REF}
738 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
739   \stex_get_document_uri:
740   \str_set:Nx \l_tmpa_str { #1 }
741   \str_if_empty:NT \l_tmpa_str {
742     \int_zero:N \l_tmpa_int
743     \bool_set_true:N \l_tmpa_bool
744     \bool_while_do:Nn \l_tmpa_bool {
745       \cs_if_exist:cTF {
746         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
747       }{
748         \int_incr:N \l_tmpa_int
749       }{
750         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
751         \bool_set_false:N \l_tmpa_bool
752       }
753     }
754   }
755   \str_set:Nx \l_tmpa_str {
756     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
757   }
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     \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
764     \str_gset:cn {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
765   }
766 }
767 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
768   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
769 }
770 </package>

```

## Chapter 17

# STEX -Modules Implementation

```
771 <*package>
772
773 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
774
775 <@@=stex_modules>
776
777 Warnings and error messages
778 \msg_new:nnn{stex}{error/unknownmodule}{
779   No~module~#1~found
780 }
781 \msg_new:nnn{stex}{error/syntax}{
782   Syntax~error:~#1
783 }
784 \msg_new:nnn{stex}{error/siglanguage}{
785   Module~#1~declares~signature~#2,~but~does~not~
786   declare~its~language
787 }
```

`\l_stex_current_module_prop` The current module:

```
786 \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

```
787 \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`

```
788 \seq_new:N \g_stex_modules_in_file_seq
789 \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
790 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
791   \prop_if_empty:NTF \l_stex_current_module_prop
792   \prg_return_false: \prg_return_true:
793 }

```

(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
794 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
795   \prop_if_exist:cTF { c_stex_module_#1_prop }
796   \prg_return_true: \prg_return_false:
797 }

```

(End definition for \stex\_if\_module\_exists:nTF. This function is documented on page 16.)

\stex\_add\_to\_current\_module:n Only allowed within modules:

```

\STEXexport
798 \cs_new_protected:Nn \stex_add_to_current_module:n {
799   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
800   \tl_put_right:Nn \l_tmpa_tl { #1 }
801   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
802 }
803 \cs_new_protected:Npn \STEXexport #1 {
804   #1
805   \stex_add_to_current_module:n { #1 }
806   \stex_smsmode_set_codes:
807 }
808 \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
809 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
810   \str_set:Nx \l_tmpa_str { #1 }
811   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
812   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
813   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
814 }

```

(End definition for \stex\_add\_constant\_to\_current\_module:n. This function is documented on page 16.)

```

\stex_add_import_to_current_module:n
815 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
816   \str_set:Nx \l_tmpa_str { #1 }
817   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
818   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
819   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
820 }

```

(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).

```

821 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
822   \str_set:Nx \l_tmpa_str { #1 }
823   \seq_set_eq:NN \l_tmpa_seq #2
824   % split off file extension
825   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
826   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
827   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
828   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
829
830   \bool_set_true:N \l_tmpa_bool
831   \bool_while_do:Nn \l_tmpa_bool {
832     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
833     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
834       {source} { \bool_set_false:N \l_tmpa_bool }
835     }{}{
836       \seq_if_empty:NT \l_tmpa_seq {
837         \bool_set_false:N \l_tmpa_bool
838       }
839     }
840   }
841
842   \seq_if_empty:NTF \l_tmpa_seq {
843     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
844   }{
845     \str_set:Nx \l_stex_modules_ns_str {
846       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
847     }
848   }
849 }

```

(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`

```

850 \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.

```

851 \cs_new_protected:Nn \stex_modules_current_namespace: {
852   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
853     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
854   }{
855     % split off file extension
856     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
857     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
858     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
859     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
860     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
861     \str_set:Nx \l_stex_modules_ns_str {
862       file:/\stex_path_to_string:N \l_tmpa_seq

```



```

863     }
864   }
865 }

```

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

## 17.1 The module environment

module arguments:

```

866 \keys_define:nn { stex / module } {
867   title      .str_set_x:N = \l_stex_module_title_str ,
868   ns         .str_set_x:N = \l_stex_module_ns_str ,
869   lang       .str_set_x:N = \l_stex_module_lang_str ,
870   sig        .str_set_x:N = \l_stex_module_sig_str ,
871   creators   .str_set_x:N = \l_stex_module_creators_str ,
872   contributors .str_set_x:N = \l_stex_module_contributors_str ,
873   meta       .str_set_x:N = \l_stex_module_meta_str
874 }
875
876 \cs_new_protected:Nn \__stex_modules_args:n {
877   \str_clear:N \l_stex_module_title_str
878   \str_clear:N \l_stex_module_ns_str
879   \str_clear:N \l_stex_module_lang_str
880   \str_clear:N \l_stex_module_sig_str
881   \str_clear:N \l_stex_module_creators_str
882   \str_clear:N \l_stex_module_contributors_str
883   \str_clear:N \l_stex_module_meta_str
884   \keys_set:nn { stex / module } { #1 }
885 }
886
887 % module parameters here? In the body?
888

```

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

```

889 \cs_new_protected:Nn \stex_module_setup:nn {
890   \str_set:Nx \l_stex_module_name_str { #2 }
891   \__stex_modules_args:n { #1 }
892
893   First, we set up the name and namespace of the module.
894   Are we in a nested module?
895
896   \stex_if_in_module:TF {
897     % Nested module
898     \prop_get:NnN \l_stex_current_module_prop
899     { ns } \l_stex_module_ns_str
900     \str_set:Nx \l_stex_module_name_str {
901       \prop_item:Nn \l_stex_current_module_prop
902       { name } / \l_stex_module_name_str
903     }
904   }{
905     % not nested:
906     \str_if_empty:NT \l_stex_module_ns_str {
907       \stex_modules_current_namespace:
908       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
909     }
910   }
911 }

```

```

905     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
906     / {\l_stex_module_ns_str}
907     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
908     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
909         \str_set:Nx \l_stex_module_ns_str {
910             \stex_path_to_string:N \l_tmpa_seq
911         }
912     }
913 }
914 }

```

Next, we determine the language of the module:

```

915 \str_if_empty:NT \l_stex_module_lang_str {
916     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
917     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
918     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
919     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
920     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
921         \stex_debug:nn{modules} {Language-\l_stex_module_lang_str~
922             inferred~from~file~name}
923         \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
924     }
925 }
926
927 \str_if_empty:NF \l_stex_module_lang_str {
928     \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
929     \l_tmpa_str {
930         \ltx@ifpackageloaded{babel}{
931             \exp_args:Nx \selectlanguage { \l_tmpa_str }
932         }{}
933     } {
934         \msg_error:nnn{stex}{error/unknownlanguage}{\l_tmpa_str}
935     }
936 }

```

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

```

937 \str_if_empty:NTF \l_stex_module_sig_str {
938     \str_clear:N \l_tmpa_str
939     \seq_clear:N \l_tmpa_seq
940     \tl_clear:N \l_tmpa_tl
941     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
942         name      = \l_stex_module_name_str ,
943         ns        = \l_stex_module_ns_str ,
944         imports   = \exp_not:o { \l_tmpa_seq } ,
945         constants = \exp_not:o { \l_tmpa_seq } ,
946         content   = \exp_not:o { \l_tmpa_tl } ,
947         file      = \exp_not:o { \g_stex_currentfile_seq } ,
948         lang      = \l_stex_module_lang_str ,
949         sig       = \l_stex_module_sig_str ,
950         meta      = \l_stex_module_meta_str
951     }
952 }{
953     \str_if_empty:NT \l_stex_module_lang_str {

```

```

954     \msg_error:nnnn{stex}{error/siglanguage}{
955       \l_stex_module_ns_str?\l_stex_module_name_str
956     }{\l_stex_module_sig_str}
957   }
958
959   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
960   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
961   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
962   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
963   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
964   \str_set:Nx \l_tmpa_str {
965     \stex_path_to_string:N \l_tmpa_seq /
966     \l_tmpa_str . \l_stex_module_sig_str .tex
967   }
968   \IfFileExists \l_tmpa_str {
969     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
970       \seq_clear:N \l_stex_all_modules_seq
971       \prop_clear:N \l_stex_current_module_prop
972       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
973       \input { \l_tmpa_str }
974     }
975   }{
976     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
977   }
978   \stex_activate_module:n {
979     \l_stex_module_ns_str ? \l_stex_module_name_str
980   }
981   \prop_set_eq:Nc \l_stex_current_module_prop {
982     c_stex_module_
983     \l_stex_module_ns_str ?
984     \l_stex_module_name_str
985     _prop
986   }
987 }

```

We load the metatheory:

```

988 \str_if_empty:NT \l_stex_module_meta_str {
989   \str_set:Nx \l_stex_module_meta_str {
990     \c_stex_metatheory_ns_str ? Metatheory
991   }
992 }
993 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
994   \exp_args:Nx \stex_add_to_current_module:n {
995     \stex_activate_module:n {\l_stex_module_meta_str}
996   }
997   \stex_activate_module:n {\l_stex_module_meta_str}
998 }
999 }

```

(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}`

```

1000 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1001   \stex_reactivate_macro:N \STEXexport
1002   \stex_reactivate_macro:N \importmodule
1003   \stex_reactivate_macro:N \symdecl
1004   \stex_reactivate_macro:N \notation
1005   \stex_reactivate_macro:N \symdef
1006   \stex_module_setup:nn{#1}{#2}
1007
1008   \stex_debug:nn{modules}{
1009     New~module:\\
1010     Namespace:~\l_stex_module_ns_str\\
1011     Name:~\l_stex_module_name_str\\
1012     Language:~\l_stex_module_lang_str\\
1013     Signature:~\l_stex_module_sig_str\\
1014     Metatheory:~\l_stex_module_meta_str\\
1015     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1016   }
1017
1018   \seq_put_right:Nx \l_stex_all_modules_seq {
1019     \l_stex_module_ns_str ? \l_stex_module_name_str
1020   }
1021
1022   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1023     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1024
1025   \stex_if_smsmode:TF {
1026     \stex_smsmode_set_codes:
1027   } {
1028     \begin{stex_annotate_env} {theory} {
1029       \l_stex_module_ns_str ? \l_stex_module_name_str
1030     }
1031
1032     \stex_annotate_invisible:nnn{header}{} {
1033       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1034       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1035       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1036         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1037       }
1038     }
1039   }
1040   % TODO: Inherit metatheory for nested modules?
1041 }
1042 \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}

```

1043 \cs_new_protected:Nn \__stex_modules_end_module: {
1044   \str_set:Nx \l_tmpa_str {
1045     c_stex_module_
1046     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1047     \prop_item:Nn \l_stex_current_module_prop { name }
1048     _prop
1049   }

```

```

1050 %^^A \prop_new:c { \l_tmpa_str }
1051 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1052 \stex_debug:nn{modules}{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
1053 }

```

(End definition for `\_stex_modules_end_module:.`)

**@module** The core environment, with no header

```

1054 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1055 \NewDocumentEnvironment { @module } { 0 } { m } {
1056   \par
1057   \_stex_modules_begin_module:nn{#1}{#2}
1058 } {
1059   \_stex_modules_end_module:
1060   \stex_if_smsmode:TF {
1061     \exp_args:Nx \stex_add_to_sms:n {
1062       \prop_gset_from_keyval:cn {
1063         c_stex_module_
1064         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1065         \prop_item:Nn \l_stex_current_module_prop { name }
1066         _prop
1067       } {
1068         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1069         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1070         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1071         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1072         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1073         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1074         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1075         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1076         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1077       }
1078     }
1079   }{
1080     \end{stex_annotate_env}
1081   }
1082 }

```

**\stex\_modules\_heading:** Code for document headers

```

1083 \cs_if_exist:NTF \thesection {
1084   \newcounter{module}[section]
1085 }{
1086   \newcounter{module}
1087 }
1088
1089 \bool_if:NT \c_stex_showmods_bool {
1090   \latexml_if:F { \RequirePackage{mdframed} }
1091 }
1092
1093 \cs_new_protected:Nn \stex_modules_heading: {
1094   \stepcounter{module}
1095   \par
1096   \bool_if:NT \c_stex_showmods_bool {
1097     \noindent{\textbf{Module}} ~

```

```

1098     \cs_if_exist:NT \thesection {\thesection.}
1099     \themodule ~ [\l_stex_module_name_str]
1100   }
1101   % TODO references
1102   % \sref@label@id{Module \thesection.\themodule [\module@name]}\%
1103   \str_if_empty:NTF \l_stex_module_title_str {
1104   }{
1105     \quad(\l_stex_module_title_str)\hfill
1106   }\par
1107 }
1108 \stex_ref_new_doc_target:n \l_stex_module_name_str
1109 }

```

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

Finally:

```

1110 \NewDocumentEnvironment { module } { 0{} m } {
1111   \bool_if:NT \c_stex_showmods_bool {
1112     \begin{mdframed}
1113   }
1114   \begin{@module}[\#1]{\#2}
1115   \stex_modules_heading:
1116 }{
1117   \end{@module}
1118   \bool_if:NT \c_stex_showmods_bool {
1119     \end{mdframed}
1120   }
1121 }

```

## 17.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1122 \NewDocumentCommand \STEXModule { m } {
1123   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1124   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1125   \tl_set:Nn \l_tmpa_tl {
1126     \msg_error:nnn{stex}{error/unknownmodule}{\#1}
1127   }
1128   \seq_map_inline:Nn \l_stex_all_modules_seq {
1129     \str_set:Nn \l_tmpb_str { ##1 }
1130     \str_if_eq:eeT { \l_tmpa_str } {
1131       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1132     } {
1133       \seq_map_break:n {
1134         \tl_set:Nn \l_tmpa_tl {
1135           \stex_invoke_module:n { ##1 }
1136         }
1137       }
1138     }
1139   }
1140   \l_tmpa_tl
1141 }
1142
1143 \cs_new_protected:Nn \stex_invoke_module:n {

```

```

1144 \stex_debug:nn{modules}{Invoking~module~#1}
1145 \peek_charcode_remove:NTF ! {
1146   \__stex_modules_invoke_uri:nN { #1 }
1147 } {
1148   \peek_charcode_remove:NTF ? {
1149     \__stex_modules_invoke_symbol:nn { #1 }
1150   } {
1151     \msg_error:nnn{stex}{error/syntax}{
1152       ?~or~!~expected~after~
1153       \c_backslash_str STEXModule{#1}
1154     }
1155   }
1156 }
1157 }
1158
1159 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1160   \str_set:Nn #2 { #1 }
1161 }
1162
1163 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1164   \stex_invoke_symbol:n{#1?#2}
1165 }

```

*(End definition for \STEXModule and \stex\_invoke\_module:n. These functions are documented on page 18.)*

**\stex\_activate\_module:n**

```

1166 \cs_new_protected:Nn \stex_activate_module:n {
1167   \stex_debug:nn{modules}{Activating~module~#1}
1168   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1169     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1170     \prop_item:cn { c_stex_module_#1_prop } { content }
1171   }
1172 }

```

*(End definition for \stex\_activate\_module:n. This function is documented on page 19.)*

```

1173 </package>

```

## Chapter 18

# STEX -Module Inheritance Implementation

```
1174 <*package>
1175
1176 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1177
```

### 18.1 SMS Mode

```
1178 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1179 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1180 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1181 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1182
1183 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1184   \makeatletter
1185   \makeatother
1186   \ExplSyntaxOn
1187   \ExplSyntaxOff
1188 }
1189
1190 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1191   \symdef
1192   \importmodule
1193   \notation
1194   \symdecl
1195   \STEXexport
1196 }
1197
1198 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1199   \tl_to_str:n {
1200     module,
1201     @module
```



```

1202 }
1203 }

```

(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
1204 \bool_new:N \g__stex_smsmode_bool
1205 \bool_set_false:N \g__stex_smsmode_bool
1206 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1207   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1208 }

```

(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
1209 \bool_new:N \g__stex_smsmode_catcode_bool
1210 \bool_set_false:N \g__stex_smsmode_catcode_bool
1211 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1212   \bool_if:NTF \g__stex_smsmode_catcode_bool
1213   \prg_return_true: \prg_return_false:
1214 }

```

(End definition for `\__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:
1215 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1216   \stex_if_smsmode:T {
1217     \__stex_smsmode_if_catcodes:F {
1218       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1219       \exp_after:wN \char_gset_active_eq:NN
1220       \c_backslash_str \__stex_smsmode_cs:
1221       \tex_global:D \char_set_catcode_active:N \
1222       \tex_global:D \char_set_catcode_other:N $
1223       \tex_global:D \char_set_catcode_other:N ^
1224       \tex_global:D \char_set_catcode_other:N _
1225       \tex_global:D \char_set_catcode_other:N &
1226       \tex_global:D \char_set_catcode_other:N ##
1227     }
1228   }
1229 } \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.
1230 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1231   \__stex_smsmode_if_catcodes:T {
1232     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1233     \exp_after:wN \tex_global:D \exp_after:wN
1234     \char_set_catcode_escape:N \c_backslash_str
1235     \tex_global:D \char_set_catcode_math_toggle:N $
1236     \tex_global:D \char_set_catcode_math_superscript:N ^
1237     \tex_global:D \char_set_catcode_math_subscript:N _
1238     \tex_global:D \char_set_catcode_alignment:N &
1239     \tex_global:D \char_set_catcode_parameter:N ##
1240   }
1241 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1242 \cs_new_protected:Nn \stex_in_smsmode:nn {
1243   \vbox_set:Nn \l_tmpa_box {
1244     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1245     \bool_gset_true:N \g__stex_smsmode_bool
1246     \stex_smsmode_set_codes:
1247     #2
1248     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1249     \stex_if_smsmode:F {
1250       \__stex_smsmode_unset_codes:
1251     }
1252   }
1253   \box_clear:N \l_tmpa_box
1254 }

```

(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:

```

1255 \cs_new_protected:Nn \_stex_smsmode_cs: {
1256   \str_clear:N \l_tmpa_str
1257   \peek_analysis_map_inline:n {
1258     % #1: token (one expansion)
1259     % #2: charcode
1260     % #3 catcode
1261     \token_if_eq_charcode:NNTF ##3 B {
1262       % token is a letter
1263       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1264     } {
1265       \str_if_empty:NTF \l_tmpa_str {
1266         % we don't allow (or need) single non-letter CSs
1267         % for now
1268         \peek_analysis_map_break:
1269       }{
1270         \str_if_eq:onTF \l_tmpa_str { begin } {
1271           \peek_analysis_map_break:n {
1272             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1273           }
1274         } {
1275           \str_if_eq:onTF \l_tmpa_str { end } {
1276             \peek_analysis_map_break:n {
1277               \exp_after:wN \_stex_smsmode_checkend:n ##1
1278             }
1279           } {
1280             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1281             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1282             \g_stex_smsmode_allowedmacros_tl
1283             { \use:c{\l_tmpa_str} } {
1284               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1285               \peek_analysis_map_break:n {
1286                 \exp_after:wN \l_tmpa_tl ##1
1287               }
1288             }
1289           }
1290         }
1291       }
1292     }
1293   }
1294 }

```

```

1288     } {
1289         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1290         \g_stex_smsmode_allowedmacros_escape_tl
1291         { \use:c{\l_tmpa_str} } {
1292             \__stex_smsmode_unset_codes:
1293             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1294             % TODO \__stex_smsmode_rescan_cs:
1295             \exp_after:wN \exp_after:wN \exp_after:wN
1296             \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1297                 \peek_analysis_map_break:n {
1298                     \__stex_smsmode_unset_codes:
1299                     \__stex_smsmode_rescan_cs:
1300                 }
1301             } {
1302                 \peek_analysis_map_break:n {
1303                     \exp_after:wN \l_tmpa_tl ##1
1304                 }
1305             }
1306         } {
1307             \peek_analysis_map_break:n { ##1 }
1308         }
1309     }
1310 }
1311 }
1312 }
1313 }
1314 }
1315 }

```

(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:

```

1316 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1317     \str_clear:N \l_tmpb_str
1318     \peek_analysis_map_inline:n {
1319         \token_if_eq_charcode:NNTF ##3 B {
1320             % token is a letter
1321             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1322         } {
1323             \peek_analysis_map_break:n {
1324                 \exp_after:wN \use:c \exp_after:wN {
1325                     \exp_after:wN \l_tmpa_str\exp_after:wN
1326                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1327             }
1328         }
1329     }
1330 }

```

(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.

```

1331 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1332     \str_set:Nn \l_tmpa_str { #1 }

```

```

1333 \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1334   \__stex_smsmode_unset_codes:
1335   \begin{#1}
1336 }
1337 }

```

(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.

```

1338 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1339   \str_set:Nn \l_tmpa_str { #1 }
1340   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1341     \end{#1}
1342   }
1343 }

```

(End definition for \\_\_stex\_smsmode\_checkend:n.)

## 18.2 Inheritance

```

1344 <@@=stex_importmodule>

```

\stex\_import\_module\_uri:nn

```

1345 \cs_new_protected:Nn \stex_import_module_uri:nn {
1346   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1347   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1348   \str_if_empty:NT \l__stex_importmodule_archive_str {
1349     \prop_if_empty:NF \l_stex_current_repository_prop {
1350       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1351     }
1352   }
1353
1354   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1355   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1356   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1357
1358   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1359     \stex_modules_current_namespace:
1360     \str_if_empty:NF \l__stex_importmodule_path_str {
1361       \str_set:Nx \l_stex_module_ns_str {
1362         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1363       }
1364     }
1365   }{
1366     \stex_require_repository:n \l__stex_importmodule_archive_str
1367     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str_manifest_prop } { ns }
1368     \l_stex_module_ns_str
1369     \str_if_empty:NF \l__stex_importmodule_path_str {
1370       \str_set:Nx \l_stex_module_ns_str {
1371         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1372       }
1373     }
1374   }
1375 }

```

(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 1376 \str_new:N \l__stex_importmodule_name_str
\l__stex_importmodule_path_str 1377 \str_new:N \l__stex_importmodule_archive_str
\l__stex_importmodule_file_str 1378 \str_new:N \l__stex_importmodule_path_str
1379 \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)}
1380 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1381   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1382
1383     % archive
1384     \str_set:Nx \l_tmpa_str { #2 }
1385     \str_if_empty:NTF \l_tmpa_str {
1386       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1387     } {
1388       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1389       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1390       \seq_put_right:Nn \l_tmpa_seq { source }
1391     }
1392
1393     % path
1394     \str_set:Nx \l_tmpb_str { #3 }
1395     \str_if_empty:NTF \l_tmpb_str {
1396       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1397
1398       \ltx@ifpackageloaded{babel} {
1399         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1400           { \languagename } \l_tmpb_str {
1401           \msg_error:nnn{stex}{error/unknownlanguage}{\languagename}
1402         }
1403       } {
1404         \str_clear:N \l_tmpb_str
1405       }
1406
1407       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1408       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1409         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1410       }{
1411         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1412         \IfFileExists{ \l_tmpa_str.tex }{
1413           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1414         }{
1415           % try english as default
1416           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1417           \IfFileExists{ \l_tmpa_str.en.tex }{
1418             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1419           }{
1420             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1421           }
1422         }

```

```

1423     }
1424
1425   } {
1426     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1427     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1428
1429     \ltx@ifpackageloaded{babel} {
1430       \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1431         { \language } \l_tmpb_str {
1432           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1433         }
1434     } {
1435       \str_clear:N \l_tmpb_str
1436     }
1437
1438     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1439
1440     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1441     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1442       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1443     }{
1444       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1445       \IfFileExists{ \l_tmpa_str/#4.tex }{
1446         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1447       }{
1448         % try english as default
1449         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1450         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1451           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1452         }{
1453           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1454           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1455             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1456           }{
1457             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1458             \IfFileExists{ \l_tmpa_str.tex }{
1459               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1460             }{
1461               % try english as default
1462               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1463               \IfFileExists{ \l_tmpa_str.en.tex }{
1464                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1465               }{
1466                 \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1467               }
1468             }
1469           }
1470         }
1471       }
1472     }
1473   }
1474
1475   \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1476   \seq_clear:N \g_stex_modules_in_file_seq

```

```

1477 % \exp_args:Nnx \use:nn {
1478 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1479 \seq_clear:N \l_stex_all_modules_seq
1480 \prop_clear:N \l_stex_current_module_prop
1481 \str_set:Nx \l_tmpb_str { #2 }
1482 \str_if_empty:NF \l_tmpb_str {
1483 \stex_set_current_repository:n { #2 }
1484 }
1485 \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1486 \input { \g__stex_importmodule_file_str }
1487 }
1488 % }{
1489
1490 % }
1491 \prop_gput:Noo \g_stex_module_files_prop
1492 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1493 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1494
1495 \stex_if_module_exists:nF { #1 ? #4 } {
1496 \msg_error:nnn{stex}{error/unknownmodule}{
1497 #1?#4~(in~file~\g__stex_importmodule_file_str)
1498 }
1499 }
1500 }
1501 \stex_activate_module:n { #1 ? #4 }
1502 }

```

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

## `\importmodule`

```

1503 \NewDocumentCommand \importmodule { 0{ } m } {
1504 \stex_import_module_uri:nn { #1 } { #2 }
1505 \stex_debug:nn{modules}{Importing~module:~
1506 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1507 }
1508 \stex_if_smsmode:F {
1509 \stex_import_require_module:nnnn
1510 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1511 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1512 \stex_annotate_invisible:nnn
1513 {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1514 }
1515 \exp_args:Nx \stex_add_to_current_module:n {
1516 \stex_import_require_module:nnnn
1517 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1518 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1519 }
1520 \exp_args:Nx \stex_add_import_to_current_module:n {
1521 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1522 }
1523 \stex_smsmode_set_codes:
1524 }
1525 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

**\usemodule**

```
1526 \NewDocumentCommand \usemodule { 0{} m } {  
1527   \stex_if_smsmode:F {  
1528     \stex_import_module_uri:nn { #1 } { #2 }  
1529     \stex_import_require_module:nnnn  
1530     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }  
1531     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }  
1532     \stex_annotate_invisible:nnn  
1533     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}  
1534   }  
1535   \stex_smsmode_set_codes:  
1536 }
```

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

```
1537 \endpackage
```



## Chapter 19

# STEX -Symbols Implementation

```
1538 <*package>
1539
1540 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1541
```

Warnings and error messages

```
1542
```

### 19.1 Symbol Declarations

```
1543 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1544 \seq_new:N \l_stex_all_symbols_seq
```

*(End definition for \l\_stex\_all\_symbols\_seq. This variable is documented on page 25.)*

`\STEXsymbol`

```
1545 \NewDocumentCommand \STEXsymbol { m } {
1546   \stex_get_symbol:n { #1 }
1547   \exp_args:No
1548   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1549 }
```

*(End definition for \STEXsymbol. This function is documented on page 27.)*

symdecl arguments:

```
1550 \keys_define:nn { stex / symdecl } {
1551   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1552   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1553   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1554   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1555   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1556   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1557   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1558   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1559 }
```

```

1560
1561 \bool_new:N \l_stex_symdecl_make_macro_bool
1562
1563 \cs_new_protected:Nn \__stex_symdecl_args:n {
1564   \str_clear:N \l_stex_symdecl_name_str
1565   \str_clear:N \l_stex_symdecl_args_str
1566   \bool_set_false:N \l_stex_symdecl_local_bool
1567   \tl_clear:N \l_stex_symdecl_type_tl
1568   \tl_clear:N \l_stex_symdecl_definiens_tl
1569
1570   \keys_set:nn { stex / symdecl } { #1 }
1571 }

```

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

```

1572
1573 \NewDocumentCommand \symdecl { s O{} m } {
1574   \__stex_symdecl_args:n { #2 }
1575   \IfBooleanTF #1 {
1576     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1577   } {
1578     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1579   }
1580   \stex_symdecl_do:n { #3 }
1581   \stex_smsmode_set_codes:
1582 }
1583 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

**\stex\_symdecl\_do:n**

```

1584 \cs_new_protected:Nn \stex_symdecl_do:n {
1585   \stex_if_in_module:F {
1586     % TODO throw error? some default namespace?
1587   }
1588
1589   \str_if_empty:NT \l_stex_symdecl_name_str {
1590     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1591   }
1592
1593   \prop_if_exist:cT { g_stex_symdecl_
1594     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1595     \prop_item:Nn \l_stex_current_module_prop {name} ?
1596     \l_stex_symdecl_name_str
1597     _prop
1598   }{
1599     % TODO throw error (beware of circular dependencies)
1600   }
1601
1602   \prop_clear:N \l_tmpa_prop
1603   \prop_put:Nnx \l_tmpa_prop { module } {
1604     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1605     \prop_item:Nn \l_stex_current_module_prop {name}
1606   }

```

```

1607 \seq_clear:N \l_tmpa_seq
1608 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1609 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1610 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1611 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1612
1613 \exp_args:No \stex_add_constant_to_current_module:n {
1614   \l_stex_symdecl_name_str
1615 }
1616
1617 % arity/args
1618 \int_zero:N \l_tmpb_int
1619
1620 \bool_set_true:N \l_tmpa_bool
1621 \str_map_inline:Nn \l_stex_symdecl_args_str {
1622   \token_case_meaning:NnF ##1 {
1623     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1624     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1625     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1626     {\tl_to_str:n a} {
1627       \bool_set_false:N \l_tmpa_bool
1628       \int_incr:N \l_tmpb_int
1629     }
1630     {\tl_to_str:n B} {
1631       \bool_set_false:N \l_tmpa_bool
1632       \int_incr:N \l_tmpb_int
1633     }
1634   }{
1635     \msg_set:nnn{stex}{error/wrongargs}{
1636       args~value~in~symbol~declaration~for~
1637       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1638       \prop_item:Nn \l_stex_current_module_prop {name} ?
1639       \l_stex_symdecl_name_str ~
1640       needs~to~be~
1641       i,~a,~b~or~B,~but~##1~given
1642     }
1643     \msg_error:nn{stex}{error/wrongargs}
1644   }
1645 }
1646 \bool_if:NTF \l_tmpa_bool {
1647   % possibly numeric
1648   \str_if_empty:NTF \l_stex_symdecl_args_str {
1649     \prop_put:Nnn \l_tmpa_prop { args } {}
1650     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1651   }{
1652     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1653     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1654     \str_clear:N \l_tmpa_str
1655     \int_step_inline:nn \l_tmpa_int {
1656       \str_put_right:Nn \l_tmpa_str i
1657     }
1658     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1659   }
1660 } {

```

```

1661 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1662 \prop_put:Nnx \l_tmpa_prop { arity }
1663 { \str_count:N \l_stex_symdecl_args_str }
1664 }
1665 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1666
1667
1668 % semantic macro
1669
1670 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1671 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1672 \prop_item:Nn \l_tmpa_prop { module } ?
1673 \prop_item:Nn \l_tmpa_prop { name }
1674 } }
1675
1676 \bool_if:NF \l_stex_symdecl_local_bool {
1677 \exp_args:Nx \stex_add_to_current_module:n {
1678 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1679 \prop_item:Nn \l_tmpa_prop { module } ?
1680 \prop_item:Nn \l_tmpa_prop { name }
1681 } }
1682 }
1683 }
1684 }
1685
1686 % add to all symbols
1687
1688 \bool_if:NF \l_stex_symdecl_local_bool {
1689 \exp_args:Nx \stex_add_to_current_module:n {
1690 \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1691 \prop_item:Nn \l_tmpa_prop { module } ?
1692 \prop_item:Nn \l_tmpa_prop { name }
1693 }
1694 }
1695 }
1696
1697 \stex_debug:nn{symbols}{New~symbol:~
1698 \prop_item:Nn \l_tmpa_prop { module } ?
1699 \prop_item:Nn \l_tmpa_prop { name } ^^J
1700 Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1701 Args:~\prop_item:Nn \l_tmpa_prop { args }
1702 }
1703
1704 % circular dependencies require this:
1705
1706 \prop_if_exist:cF {
1707 g_stex_symdecl_
1708 \prop_item:Nn \l_tmpa_prop { module } ?
1709 \prop_item:Nn \l_tmpa_prop { name }
1710 _prop
1711 } {
1712 \prop_gset_eq:cN {
1713 g_stex_symdecl_
1714 \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1715     \prop_item:Nn \l_tmpa_prop { name }
1716     _prop
1717   } \l_tmpa_prop
1718 }
1719
1720 \stex_if_smsmode:TF {
1721   \bool_if:NF \l_stex_symdecl_local_bool {
1722     \exp_args:Nx \stex_add_to_sms:n {
1723       \prop_gset_from_keyval:cn {
1724         g_stex_symdecl_
1725         \prop_item:Nn \l_tmpa_prop { module } ?
1726         \prop_item:Nn \l_tmpa_prop { name }
1727         _prop
1728       } {
1729         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1730         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1731         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1732         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1733         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1734         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1735         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1736         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1737       }
1738       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1739         \prop_item:Nn \l_tmpa_prop { module } ?
1740         \prop_item:Nn \l_tmpa_prop { name }
1741       }
1742     }
1743   }
1744 }{
1745   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1746     \prop_item:Nn \l_tmpa_prop { module } ?
1747     \prop_item:Nn \l_tmpa_prop { name }
1748   }
1749   \stex_if_do_html:T {
1750     \stex_annotate_invisible:nnn {symdecl} {
1751       \prop_item:Nn \l_tmpa_prop { module } ?
1752       \prop_item:Nn \l_tmpa_prop { name }
1753     } {
1754       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1755       \stex_annotate_invisible:nnn{args}{}{
1756         \prop_item:Nn \l_tmpa_prop { args }
1757       }
1758       \stex_annotate_invisible:nnn{macroname}{}{#1}
1759       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1760         \stex_annotate_invisible:nnn{definiens}{}
1761         {\l_stex_symdecl_definiens_tl$}
1762       }
1763     }
1764   }
1765 }
1766 }

```

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

`\stex_get_symbol:n`

```
1767 \str_new:N \l_stex_get_symbol_uri_str
1768
1769 \cs_new_protected:Nn \stex_get_symbol:n {
1770   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1771     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1772   }{
1773     % argument is a string
1774     % is it a command name?
1775     \cs_if_exist:cTF { #1 }{
1776       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1777       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1778       \str_if_empty:NTF \l_tmpa_str {
1779         \exp_args:Nx \cs_if_eq:NNTF {
1780           \tl_head:N \l_tmpa_tl
1781         } \stex_invoke_symbol:n {
1782           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1783         }{
1784           \__stex_symdecl_get_symbol_from_string:n { #1 }
1785         }
1786       } {
1787         \__stex_symdecl_get_symbol_from_string:n { #1 }
1788       }
1789     }{
1790       % argument is not a command name
1791       \__stex_symdecl_get_symbol_from_string:n { #1 }
1792       % \l_stex_all_symbols_seq
1793     }
1794   }
1795 }
1796
1797 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1798   \str_set:Nn \l_tmpa_str { #1 }
1799   \bool_set_false:N \l_tmpa_bool
1800   \stex_if_in_module:T {
1801     \prop_get:NnN \l_stex_current_module_prop
1802     { constants } \l_tmpa_seq
1803     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1804       \bool_set_true:N \l_tmpa_bool
1805       \str_set:Nx \l_stex_get_symbol_uri_str {
1806         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1807         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1808       }
1809     }
1810   }
1811   \bool_if:NF \l_tmpa_bool {
1812     \tl_set:Nn \l_tmpa_tl {
1813       \msg_set:nnn{stex}{error/unknownsymbol}{
1814         No~symbol~#1~found!
1815       }
1816     }
1817     \msg_error:nn{stex}{error/unknownsymbol}
1818   }
1819   \str_set:Nn \l_tmpa_str { #1 }
1820   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1820 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1821   \str_set:Nn \l_tmpb_str { ##1 }
1822   \str_if_eq:eeT { \l_tmpa_str } {
1823     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1824   } {
1825     \seq_map_break:n {
1826       \tl_set:Nn \l_tmpa_tl {
1827         \str_set:Nn \l_stex_get_symbol_uri_str {
1828           ##1
1829         }
1830       }
1831     }
1832   }
1833 }
1834 \l_tmpa_tl
1835 }
1836 }
1837
1838 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1839   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1840     { \tl_tail:N \l_tmpa_tl }
1841   \tl_if_single:NTF \l_tmpa_tl {
1842     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1843       \exp_after:wN \str_set:Nn \exp_after:wN
1844         \l_stex_get_symbol_uri_str \l_tmpa_tl
1845     }{
1846       % TODO
1847       % tail is not a single group
1848     }
1849   }{
1850     % TODO
1851     % tail is not a single group
1852   }
1853 }

```

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

## 19.2 Notations

```

1854 <@@=stex_notation>
1855 notation arguments:
1856 \keys_define:nn { stex / notation } {
1857   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1858   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1859   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1860   op .tl_set:N = \l__stex_notation_op_tl ,
1861   unknown .code:n = \str_set:Nx
1862     \l__stex_notation_variant_str \l_keys_key_str
1863 }
1864 \cs_new_protected:Nn \__stex_notation_args:n {
1865   \str_clear:N \l__stex_notation_lang_str
1866   \str_clear:N \l__stex_notation_variant_str

```

```

1867 \str_clear:N \l__stex_notation_prec_str
1868 \tl_clear:N \l__stex_notation_op_tl
1869
1870 \keys_set:nn { stex / notation } { #1 }
1871 }

```

## **\notation**

```

1872 \NewDocumentCommand \notation { 0{ } m } {
1873   \__stex_notation_args:n { #1 }
1874   \tl_clear:N \l_stex_symdecl_definiens_tl
1875   \stex_get_symbol:n { #2 }
1876   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1877 }
1878 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

## **\stex\_notation\_do:nn**

```

1879 \cs_new_protected:Nn \stex_notation_do:nn {
1880   \prop_set_eq:Nc \l_tmpa_prop {
1881     g_stex_symdecl_ #1 _prop
1882   }
1883
1884   \prop_clear:N \l_tmpb_prop
1885   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1886   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1887   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1888
1889   % precedences
1890   \seq_clear:N \l_tmpb_seq
1891   \exp_args:NNno
1892   \str_if_empty:NTF \l__stex_notation_prec_str {
1893     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1894     \int_compare:nNnTF \l_tmpa_str = 0 {
1895       \exp_args:NNnx
1896       \prop_put:Nno \l_tmpb_prop { opprec }
1897       { \neginfprec }
1898     }{
1899       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1900     }
1901   } {
1902     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1903       \exp_args:NNnx
1904       \prop_put:Nno \l_tmpb_prop { opprec }
1905       { \neginfprec }
1906       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1907       \int_step_inline:nn { \l_tmpa_str } {
1908         \exp_args:NNx
1909         \seq_put_right:Nn \l_tmpb_seq { \infprec }
1910       }
1911     }{
1912       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1913       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1914         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1915         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```



```

1916         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1917         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1918         \seq_map_inline:Nn \l_tmpa_seq {
1919             \seq_put_right:Nn \l_tmpb_seq { ##1 }
1920         }
1921     }
1922     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1923 }{
1924     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1925     \int_compare:nNnTF \l_tmpa_str = 0 {
1926         \exp_args:NNnx
1927         \prop_put:Nno \l_tmpb_prop { opprec }
1928         { \infprec }
1929     }{
1930         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1931     }
1932 }
1933 }
1934 }
1935
1936 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1937 \int_step_inline:nn { \l_tmpa_str } {
1938     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
1939         \exp_args:NNx
1940         \seq_put_right:Nn \l_tmpb_seq {
1941             \prop_item:Nn \l_tmpb_prop { opprec }
1942         }
1943     }
1944 }
1945
1946 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1947 \tl_clear:N \l_tmpa_tl
1948
1949 \int_compare:nNnTF \l_tmpa_str = 0 {
1950     \exp_args:NNe
1951     \cs_set:Npn \l__stex_notation_macrocode_cs {
1952         \_stex_term_math_oms:nnnn { #1 }
1953         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1954         { \prop_item:Nn \l_tmpb_prop { opprec } }
1955         { \exp_not:n { #2 } }
1956     }
1957     \__stex_notation_final:
1958 }{
1959     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1960     \str_if_in:NnTF \l_tmpb_str b {
1961         \exp_args:Nne \use:nn
1962         {
1963             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1964             \cs_set:Npn \l_tmpa_str { {
1965                 \_stex_term_math_omb:nnnn { #1 }
1966                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1967                 { \prop_item:Nn \l_tmpb_prop { opprec } }
1968                 { \exp_not:n { #2 } }
1969             }}

```

```

1970   }{
1971     \str_if_in:NnTF \l_tmpb_str B {
1972       \exp_args:Nne \use:nn
1973       {
1974         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1975         \cs_set:Npn \l_tmpa_str } { {
1976           \stex_term_math_omb:nnnn { #1 }
1977           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1978           { \prop_item:Nn \l_tmpb_prop { opprec } }
1979           { \exp_not:n { #2 } }
1980         } }
1981       }{
1982         \exp_args:Nne \use:nn
1983         {
1984           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1985           \cs_set:Npn \l_tmpa_str } { {
1986             \stex_term_math_oma:nnnn { #1 }
1987             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1988             { \prop_item:Nn \l_tmpb_prop { opprec } }
1989             { \exp_not:n { #2 } }
1990           } }
1991         }
1992       }
1993
1994     \int_zero:N \l_tmpa_int
1995     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1996     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1997     \__stex_notation_arguments:
1998   }
1999 }

```

(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

```

2000 \cs_new_protected:Nn \__stex_notation_arguments: {
2001   \int_incr:N \l_tmpa_int
2002   \str_if_empty:NnTF \l_tmpa_str {
2003     \__stex_notation_final:
2004   }{
2005     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2006     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2007     \str_if_eq:NnTF \l_tmpb_str a {
2008       \__stex_notation_argument_assoc:n
2009     }{
2010       \str_if_eq:NnTF \l_tmpb_str B {
2011         \__stex_notation_argument_assoc:n
2012       }{
2013         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2014         \tl_put_right:Nx \l_tmpa_tl {
2015           { \stex_term_math_arg:nnn
2016             { \int_use:N \l_tmpa_int }
2017             { \l_tmpb_str }
2018             { ####\int_use:N \l_tmpa_int }
2019           }

```

```

2020     }
2021     \__stex_notation_arguments:
2022   }
2023 }
2024 }
2025 }

```

(End definition for \\_\_stex\_notation\_arguments:.)

\\_\_stex\_notation\_argument\_assoc:n

```

2026 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2027   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2028   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2029   \tl_put_right:Nx \l_tmpa_tl {
2030     { \stex_term_math_assoc_arg:nnnn
2031       { \int_use:N \l_tmpa_int }
2032       { \l_tmpb_str }
2033       \exp_args:No \exp_not:n
2034       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2035       { ####\int_use:N \l_tmpa_int }
2036     }
2037   }
2038   \__stex_notation_arguments:
2039 }

```

(End definition for \\_\_stex\_notation\_argument\_assoc:n.)

\\_\_stex\_notation\_final: Called after processing all notation arguments

```

2040 \cs_new_protected:Nn \__stex_notation_final: {
2041   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2042   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2043   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2044   \exp_args:Nne \use:nn
2045   {
2046     \cs_generate_from_arg_count:cNnn {
2047       stex_notation_ \l_tmpa_str \c_hash_str
2048       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2049       _cs
2050     }
2051     \cs_gset:Npn \l_tmpb_str } { {
2052       \exp_after:wN \exp_after:wN \exp_after:wN
2053       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2054       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2055     } }
2056
2057   \tl_if_empty:NF \l__stex_notation_op_tl {
2058     \cs_gset:cpx {
2059       stex_op_notation_ \l_tmpa_str \c_hash_str
2060       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2061       _cs
2062     } {
2063       \stex_term_oms:nnn {
2064         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2065         \l__stex_notation_lang_str

```

```

2066     }{
2067         \l_tmpa_str
2068     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2069 }
2070 }
2071
2072
2073
2074 \stex_debug:nn{symbols}{
2075     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2076     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2077     Operator~precedence:~
2078     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2079     Argument~precedences:~
2080     \seq_use:Nn \l_tmpa_seq {,~}^^J
2081     Notation: \cs_meaning:c {
2082         stex_notation_ \l_tmpa_str \c_hash_str
2083         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2084         _cs
2085     }
2086 }
2087
2088 \prop_gset_eq:cN {
2089     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2090     \c_hash_str \l__stex_notation_lang_str _prop
2091 } \l_tmpb_prop
2092
2093 \exp_args:Nx
2094 \stex_add_to_current_module:n {
2095     \prop_get:cnN {
2096         g_stex_symdecl_
2097         \prop_item:Nn \l_tmpb_prop { symbol }
2098         _prop
2099     } { notations } \exp_not:N \l_tmpa_seq
2100     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2101         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2102     }
2103     \prop_put:cno {
2104         g_stex_symdecl_
2105         \prop_item:Nn \l_tmpb_prop { symbol }
2106         _prop
2107     } { notations } \exp_not:N \l_tmpa_seq
2108 }
2109
2110 \stex_if_smsmode:TF {
2111     \stex_smsmode_set_codes:
2112     \exp_args:Nx \stex_add_to_sms:n {
2113         \prop_gset_from_keyval:cn {
2114             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2115             \c_hash_str \l__stex_notation_lang_str _prop
2116         } {
2117             symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2118             language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2119             variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,

```

```

2120         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2121         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2122     }
2123 }
2124 }{
2125   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2126   \seq_put_right:Nx \l_tmpa_seq {
2127     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2128   }
2129   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2130   \prop_set_eq:cN {
2131     g_stex_symdecl_ \l_tmpa_str _prop
2132   } \l_tmpa_prop
2133
2134   % HTML annotations
2135   \stex_if_do_html:T {
2136     \stex_annotate_invisible:nnn { notation }
2137     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2138       \stex_annotate_invisible:nnn { notationfragment }
2139       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2140     }
2141     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2142     \stex_annotate_invisible:nnn { precedence }
2143     { \prop_item:Nn \l_tmpb_prop { opprec } ;
2144       \seq_use:Nn \l_tmpa_seq { x }
2145     }{}
2146
2147     \int_zero:N \l_tmpa_int
2148     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2149     \tl_clear:N \l_tmpa_tl
2150     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2151       \int_incr:N \l_tmpa_int
2152       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2153       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2154       \str_if_eq:VnTF \l_tmpb_str a {
2155         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2156           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2157           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2158         } }
2159       }{
2160         \str_if_eq:VnTF \l_tmpb_str B {
2161           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2162             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2163             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2164           } }
2165         }{
2166           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2167             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2168           } }
2169         }
2170       }
2171     }
2172     \stex_annotate_invisible:nnn { notationcomp }{}{
2173       $ \exp_args:Nno \use:nn { \use:c {
2174         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2174         \c_hash_str \l__stex_notation_variant_str
2175         \c_hash_str \l__stex_notation_lang_str _cs
2176     } } { \l_tmpa_tl } $
2177 }
2178 }
2179 }
2180 }
2181 }

```

(End definition for \\_stex\_notation\_final:.)

**\symdef**

```

2182 \keys_define:nn { stex / symdef } {
2183   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2184   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2185   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2186   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2187   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2188   op        .tl_set:N   = \l__stex_notation_op_tl ,
2189   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2190   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2191   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2192   unknown   .code:n     = \str_set:Nx
2193             \l__stex_notation_variant_str \l_keys_key_str
2194 }
2195
2196 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2197   \str_clear:N \l_stex_symdecl_name_str
2198   \str_clear:N \l_stex_symdecl_args_str
2199   \bool_set_false:N \l_stex_symdecl_local_bool
2200   \tl_clear:N \l_stex_symdecl_type_tl
2201   \tl_clear:N \l_stex_symdecl_definiens_tl
2202   \str_clear:N \l__stex_notation_lang_str
2203   \str_clear:N \l__stex_notation_variant_str
2204   \str_clear:N \l__stex_notation_prec_str
2205   \tl_clear:N \l__stex_notation_op_tl
2206
2207   \keys_set:nn { stex / symdef } { #1 }
2208 }
2209
2210 \NewDocumentCommand \symdef { 0{} m } {
2211   \_stex_notation_symdef_args:n { #1 }
2212   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2213   \stex_symdecl_do:n { #2 }
2214   \exp_args:Nx \stex_notation_do:nn {
2215     \prop_item:Nn \l_tmpa_prop { module } ?
2216     \prop_item:Nn \l_tmpa_prop { name }
2217   }
2218 }
2219 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2220 \</package>

```

## Chapter 20

# STEX -Terms Implementation

```
2221 <*package>
2222
2223 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2224
2225 <@@=stex_terms>
    Warnings and error messages
2226 \msg_new:nnn{stex}{error/nonotation}{
2227   Symbol~#1~invoked,~but~has~no~notation#2!
2228 }
2229 \msg_new:nnn{stex}{error/notationarg}{
2230   Error~in~parsing~notation~#1
2231 }
2232
```

### 20.1 Symbol Invocations

Arguments:

```
2233 \keys_define:nn { stex / terms } {
2234   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2235   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2236   unknown .code:n = \str_set:Nx
2237     \l__stex_terms_variant_str \l_keys_key_str
2238 }
2239
2240 \cs_new_protected:Nn \__stex_terms_args:n {
2241   \str_clear:N \l__stex_terms_lang_str
2242   \str_clear:N \l__stex_terms_variant_str
2243   \str_clear:N \l__stex_terms_prec_str
2244   \tl_clear:N \l__stex_terms_op_tl
2245
2246   \keys_set:nn { stex / terms } { #1 }
2247 }
```

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2248 \cs_new_protected:Nn \stex_invoke_symbol:n {
2249   \if_mode_math:
2250     \exp_after:wN \__stex_terms_invoke_math:n
2251   \else:
2252     \exp_after:wN \__stex_terms_invoke_text:n
2253   \fi: { #1 }
2254 }

```

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

`\__stex_terms_invoke_math:n`

```

2255 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2256   \peek_charcode_remove:NTF ! {
2257     \peek_charcode:NTF [ {
2258       \__stex_terms_invoke_op:nw { #1 }
2259     }{
2260       \__stex_terms_invoke_op:nw { #1 } []
2261     }
2262   }{
2263     \peek_charcode_remove:NTF * {
2264       \__stex_terms_invoke_text:n { #1 }
2265     }{
2266       \peek_charcode:NTF [ {
2267         \__stex_terms_invoke_math:nw { #1 }
2268       }{
2269         \__stex_terms_invoke_math:nw { #1 } []
2270       }
2271     }
2272   }
2273 }

```

(End definition for `\__stex_terms_invoke_math:n`.)

`\__stex_terms_invoke_op:nw`

```

2274 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2275   \__stex_terms_args:n { #2 }
2276   \cs_if_exist:cTF {
2277     stex_op_notation_ #1 \c_hash_str
2278     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2279   }{
2280     \csname stex_op_notation_ #1 \c_hash_str
2281     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2282   \endcsname
2283   }{
2284     % TODO throw error
2285   }
2286 }

```

(End definition for `\__stex_terms_invoke_op:nw`.)

`\__stex_terms_invoke_math:nw`

```

2287 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2288   \__stex_terms_args:n { #2 }
2289   \prop_set_eq:Nc \l_tmpa_prop {
2290     g_stex_symdecl_ #1 _prop

```



```

2291 }
2292 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2293 \seq_if_empty:NTF \l_tmpa_seq {
2294   \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2295 } {
2296   \seq_if_in:NxTF \l_tmpa_seq
2297   { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2298     \use:c{
2299       stex_notation_ #1 \c_hash_str
2300       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2301       _cs
2302     }
2303   }{
2304     \str_if_empty:NTF \l__stex_terms_variant_str {
2305       \str_if_empty:NTF \l__stex_terms_lang_str {
2306         \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2307         \use:c{
2308           stex_notation_ #1 \c_hash_str \l_tmpa_str
2309           _cs
2310         }
2311       }{
2312         \msg_error:nn{stex}{error/nonotation}{#1}{
2313           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2314         }
2315       }
2316     }{
2317       \msg_error:nn{stex}{error/nonotation}{#1}{
2318         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2319       }
2320     }
2321   }
2322 }
2323 }

```

(End definition for `\__stex_terms_invoke_math:nw`.)

`\__stex_terms_invoke_text:n`

```

2324 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2325   \peek_charcode_remove:NTF ! {
2326     \stex_term_custom:nn { #1 } { }
2327   }{
2328     \prop_set_eq:Nc \l_tmpa_prop {
2329       g_stex_symdecl_ #1 _prop
2330     }
2331     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2332     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2333   }
2334 }

```

(End definition for `\__stex_terms_invoke_text:n`.)

## 20.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2335 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2336 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2337 \int_new:N \l__stex_terms_downprec
2338 \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
2339 \tl_set:Nn \l__stex_terms_left_bracket_str (
2340 \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
2341 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2342 \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2343 \bool_if:NNTF \l_stex_inarray_bool { #2 }{
2344 \dobrackets { #2 }
2345 }
2346 }{ #2 }
2347 }

(End definition for \__stex_terms_maybe_brackets:nn.)

```

```

\dobrackets
2348 %\RequirePackage{scalerel}
2349 \cs_new_protected:Npn \dobrackets #1 {
2350 %\ThisStyle{\if D\m@switch
2351 % \exp_args:Nnx \use:nn
2352 % { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2353 % { \exp_not:N\right\l__stex_terms_right_bracket_str }
2354 % \else
2355 \exp_args:Nnx \use:nn
2356 { \l__stex_terms_left_bracket_str #1 }
2357 { \l__stex_terms_right_bracket_str }
2358 %\fi}
2359 }

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

```

```

\withbrackets
2360 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2361 \exp_args:Nnx \use:nn
2362 {
2363 \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2364 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2365 #3
2366 }
2367 {
2368 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2369 {\l__stex_terms_left_bracket_str}
2370 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str

```

```

2371         {\l__stex_terms_right_bracket_str}
2372     }
2373 }

```

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

#### \STEXinvisible

```

2374 \cs_new_protected:Npn \STEXinvisible #1 {
2375     \stex_annotate_invisible:n { #1 }
2376 }

```

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

OMDoc terms:

#### \\_stex\_term\_math\_oms:nnnn

```

2377 \cs_new_protected:Nn \_stex_term_oms:nnn {
2378     \stex_annotate:nnn{ OMID }{ #2 }{
2379         \stex_highlight_term:nn { #1 } { #3 }
2380     }
2381 }
2382
2383 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2384     \__stex_terms_maybe_brackets:nn { #3 }{
2385         \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2386     }
2387 }

```

(End definition for \\_stex\_term\_math\_oms:nnnn. This function is documented on page 27.)

#### \\_stex\_term\_math\_oma:nnnn

```

2388 \cs_new_protected:Nn \_stex_term_oma:nnn {
2389     \stex_annotate:nnn{ OMA }{ #2 }{
2390         \stex_highlight_term:nn { #1 } { #3 }
2391     }
2392 }
2393
2394 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2395     \__stex_terms_maybe_brackets:nn { #3 }{
2396         \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2397     }
2398 }

```

(End definition for \\_stex\_term\_math\_oma:nnnn. This function is documented on page 27.)

#### \\_stex\_term\_math\_omb:nnnn

```

2399 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2400     \stex_annotate:nnn{ OMBIND }{ #2 }{
2401         \stex_highlight_term:nn { #1 } { #3 }
2402     }
2403 }
2404
2405 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2406     \__stex_terms_maybe_brackets:nn { #3 }{
2407         \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2408     }
2409 }

```

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

`\_stex_term_math_arg:nnn`

```

2410 \cs_new_protected:Nn \_stex_term_arg:nn {
2411   \stex_unhighlight_term:n {
2412     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2413   }
2414 }
2415 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2416   \exp_args:Nnx \use:nn
2417     { \int_set:Nn \l__stex_terms_downprec { #2 }
2418       \_stex_term_arg:nn { #1 }{ #3 }
2419     }
2420   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2421 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

2422 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2423   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2424   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2425     \tl_set:Nn \l_tmpa_tl { #4 }
2426   }{
2427     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2428     \seq_reverse:N \l_tmpa_seq
2429     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2430     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2431
2432     \seq_map_inline:Nn \l_tmpa_seq {
2433       \exp_args:Nno \tl_set:No \l_tmpa_tl {
2434         \exp_args:Nno
2435           \l_tmpa_cs { ##1 } \l_tmpa_tl
2436       }
2437     }
2438
2439   }
2440   \exp_args:Nnno
2441   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2442 }

```

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

`\stex_term_custom:nn`

```

2443 \cs_new_protected:Nn \stex_term_custom:nn {
2444   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2445   \str_set:Nn \l_tmpa_str { #2 }
2446   \tl_clear:N \l_tmpa_tl
2447   \int_zero:N \l_tmpa_int
2448   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2449   \__stex_terms_custom_loop:
2450 }

```

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

\\_stex\_terms\_custom\_loop:

```

2451 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2452   \bool_set_false:N \l_tmpa_bool
2453   \bool_while_do:nn {
2454     \str_if_eq_p:ee X {
2455       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2456     }
2457   }{
2458     \int_incr:N \l_tmpa_int
2459   }
2460
2461   \peek_charcode:NTF [ {
2462     % notation/text component
2463     \_stex_terms_custom_component:w
2464   } {
2465     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2466       % all arguments read => finish
2467       \_stex_terms_custom_final:
2468     } {
2469       % arguments missing
2470       \peek_charcode_remove:NTF * {
2471         % invisible, specific argument position or both
2472         \peek_charcode:NTF [ {
2473           % visible specific argument position
2474           \_stex_terms_custom_arg:wn
2475         } {
2476           % invisible
2477           \peek_charcode_remove:NTF * {
2478             % invisible specific argument position
2479             \_stex_terms_custom_arg_inv:wn
2480           } {
2481             % invisible next argument
2482             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2483           }
2484         }
2485       } {
2486         % next normal argument
2487         \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2488       }
2489     }
2490   }
2491 }

```

(End definition for \\_stex\_terms\_custom\_loop:.)

\\_stex\_terms\_custom\_arg\_inv:wn

```

2492 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2493   \bool_set_true:N \l_tmpa_bool
2494   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2495 }

```

(End definition for \\_stex\_terms\_custom\_arg\_inv:wn.)

\\_stex\_terms\_custom\_arg:wn

```

2496 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2497   \str_set:Nx \l_tmpb_str {
2498     \str_item:Nn \l_tmpa_str { #1 }
2499   }
2500   \str_case:VnTF \l_tmpb_str {
2501     { X } {
2502       \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2503     }
2504     { i } { \__stex_terms_custom_set_X:n { #1 } }
2505     { b } { \__stex_terms_custom_set_X:n { #1 } }
2506     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2507     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2508   }{}{
2509     \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2510   }
2511
2512   \bool_if:nTF \l_tmpa_bool {
2513     \tl_put_right:Nx \l_tmpa_tl {
2514       \stex_annotate_invisible:n {
2515         \stex_term_arg:nn { \int_eval:n { #1 } }
2516         \exp_not:n { { #2 } }
2517       }
2518     }
2519   } {
2520     \tl_put_right:Nx \l_tmpa_tl {
2521       \stex_term_arg:nn { \int_eval:n { #1 } }
2522       \exp_not:n { { #2 } }
2523     }
2524   }
2525
2526   \__stex_terms_custom_loop:
2527 }

```

(End definition for \\_\_stex\_terms\_custom\_arg:wn.)

\\_\_stex\_terms\_custom\_set\_X:n

```

2528 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2529   \str_set:Nx \l_tmpa_str {
2530     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2531     X
2532     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2533   }
2534 }

```

(End definition for \\_\_stex\_terms\_custom\_set\_X:n.)

\\_\_stex\_terms\_custom\_component:

```

2535 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2536   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2537   \__stex_terms_custom_loop:
2538 }

```

(End definition for \\_\_stex\_terms\_custom\_component:.)

`\_stex_terms_custom_final:`

```

2539 \cs_new_protected:Nn \_stex_terms_custom_final: {
2540   \int_compare:nNnTF \l_tmpb_int = 0 {
2541     \exp_args:Nnno \_stex_term_oms:nnn
2542   }{
2543     \str_if_in:NnTF \l_tmpa_str {b} {
2544       \exp_args:Nnno \_stex_term_ombind:nnn
2545     } {
2546       \exp_args:Nnno \_stex_term_oma:nnn
2547     }
2548   }
2549   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2550 }

```

*(End definition for \\_stex\_terms\_custom\_final:.)*

**\symref**

**\symname**

```

2551 \NewDocumentCommand \symref { m m }{
2552   \STEXsymbol{#1}![#2]
2553 }
2554
2555 \keys_define:nn { stex / symname } {
2556   post      .str_set_x:N   = \l_stex_symname_post_str
2557 }
2558
2559 \cs_new_protected:Nn \stex_symname_args:n {
2560   \str_clear:N \l_stex_symname_post_str
2561   \keys_set:nn { stex / symname } { #1 }
2562 }
2563
2564 \NewDocumentCommand \symname { 0{} m }{
2565   \stex_symname_args:n { #1 }
2566   \stex_get_symbol:n { #2 }
2567   \str_set:Nx \l_tmpa_str {
2568     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2569   }
2570   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2571   \exp_args:NNx \use:nn
2572   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2573     \l_tmpa_str \l_stex_symname_post_str
2574   ] }
2575 }

```

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

## 20.3 Notation Components

2576 `<@@=stex_notationcomps>`

**\stex\_highlight\_term:nn**

```

2577
2578 \str_new:N \l__stex_notationcomps_highlight_uri_str
2579 \cs_new_protected:Nn \stex_highlight_term:nn {
2580   \exp_args:Nnx

```

```

2581 \use:nn {
2582   \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2583   #2
2584 } {
2585   \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2586   { \l__stex_notationcomps_highlight_uri_str }
2587 }
2588 }
2589
2590 \cs_new_protected:Nn \stex_unhighlight_term:n {
2591 % \latexml_if:TF {
2592 %   #1
2593 % } {
2594 %   \scalatex_if:TF {
2595 %     #1
2596 %   } {
2597 %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2598 %   }
2599 % }
2600 }

```

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

```

\comp
\@comp
\@defemph
2601 \cs_new_protected:Npn \comp #1 {
2602   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2603     \scalatex_if:TF {
2604       \stex_annotate:nnn { comp } { \l__stex_notationcomps_highlight_uri_str } { #1 }
2605     } {
2606       \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2607     }
2608   }
2609 }
2610
2611 \cs_new_protected:Npn \@comp #1 #2 {
2612   \textcolor{blue}{#1}
2613 }
2614
2615 \cs_new_protected:Npn \@defemph #1 #2 {
2616   \textbf{\textcolor{magenta}{#1}}
2617 }

```

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

`\ellipses`

```

2618 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
2619 \bool_new:N \l_stex_inparray_bool
2620 \bool_set_false:N \l_stex_inparray_bool
2621 \NewDocumentCommand \parray { m m } {
2622   \begingroup
2623   \bool_set_true:N \l_stex_inparray_bool

```



```

2624 \begin{array}{#1}
2625     #2
2626 \end{array}
2627 \endgroup
2628 }
2629
2630 \NewDocumentCommand \prmatrix { m } {
2631     \begingroup
2632     \bool_set_true:N \l_stex_inarray_bool
2633     \begin{matrix}
2634         #1
2635     \end{matrix}
2636 \endgroup
2637 }
2638
2639 \def \parrayline #1 #2 {
2640     #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2641 }
2642
2643 \def \parraylineh #1 #2 {
2644     #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2645 }
2646
2647 \def \parraycell #1 {
2648     #1 \bool_if:NT \l_stex_inarray_bool {&}
2649 }

```

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

```

2650 \end{package}

```

## Chapter 21

# STEX -Structural Features Implementation

```
2651 <*package>
2652
2653 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2654
2655 <@@=stex_features>
      Warnings and error messages
2656
symboldoc
2657 \NewDocumentEnvironment{symboldoc}{m}{
2658   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2659   \seq_clear:N \l_tmpb_seq
2660   \seq_map_inline:Nn \l_tmpa_seq {
2661     \stex_get_symbol:n { ##1 }
2662     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2663       \l_stex_get_symbol_uri_str
2664     }
2665   }
2666   \par
2667   \exp_args:Nnnx
2668   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2669 }{
2670   \end{stex_annotate_env}
2671 }
STEXdefinition
2672
2673 \NewDocumentCommand \definiendum { O{} m m } {
2674   \stex_get_symbol:n { #2 }
2675   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
2676   \scalatex_if:TF {
2677     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
```

```

2678 } {
2679   \exp_args:Nnx \@defemph { #3 } { \l_stex_get_symbol_uri_str }
2680 }
2681 }
2682 \stex_deactivate_macro:Nn \definiendum {definition~environments}
2683 \NewDocumentCommand \definame { 0{ } m } {
2684   % TODO: root
2685   \stex_get_symbol:n { #2 }
2686   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
2687   \str_set:Nx \l_tmpa_str {
2688     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2689   }
2690   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2691   \scalatex_if:TF {
2692     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
2693       \l_tmpa_str
2694     }
2695   } {
2696     \@defemph {
2697       \l_tmpa_str
2698     } { \l_stex_get_symbol_uri_str }
2699   }
2700 }
2701 \stex_deactivate_macro:Nn \definame {definition~environments}
2702
2703 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2704   \stex_reactivate_macro:N \definiendum
2705   \stex_reactivate_macro:N \definame
2706   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2707   \seq_clear:N \l_tmpb_seq
2708   \seq_map_inline:Nn \l_tmpa_seq {
2709     \stex_get_symbol:n { ##1 }
2710     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2711       \l_stex_get_symbol_uri_str
2712     }
2713   }
2714   \exp_args:Nnnx
2715   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2716 }
2717
2718 \cs_new_protected:Nn \__stex_features_defi_end: {
2719   \end{stex_annotate_env}
2720 }
2721
2722 \NewDocumentEnvironment{STEXdefinition}{ m }{
2723   \__stex_features_defi_begin:n { #1 }
2724 }{
2725   \__stex_features_defi_end:
2726 }

```

\setSTEXdefinition

```

2727 \cs_new_protected:Npn \setSTEXdefinition #1 {
2728   \AddToHook{env/#1/before}[stex]{\__stex_features_defi_begin:n{}}
2729   \AddToHook{env/#1/after}[stex]{\__stex_features_defi_end:}

```

2730 }

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

structural@feature

```

2731
2732 \NewDocumentEnvironment{structural@feature}{ m m m }{
2733   \stex_if_in_module:F {
2734     \msg_set:nnn{stex}{error/nomodule}{
2735       Structural~Feature~has~to~occur~in~a~module:\\
2736       Feature~#2~of~type~#1\\
2737       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2738     }
2739     \msg_error:nn{stex}{error/nomodule}
2740   }
2741
2742   \str_set:Nx \l_stex_module_name_str {
2743     \prop_item:Nn \l_stex_current_module_prop
2744       { name } / #2 - feature
2745   }
2746
2747   \str_set:Nx \l_stex_module_ns_str {
2748     \prop_item:Nn \l_stex_current_module_prop
2749       { ns }
2750   }
2751
2752
2753   \str_clear:N \l_tmpa_str
2754   \seq_clear:N \l_tmpa_seq
2755   \tl_clear:N \l_tmpa_tl
2756   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2757     origname = #2,
2758     name     = \l_stex_module_name_str ,
2759     ns       = \l_stex_module_ns_str ,
2760     imports  = \exp_not:o { \l_tmpa_seq } ,
2761     constants = \exp_not:o { \l_tmpa_seq } ,
2762     content  = \exp_not:o { \l_tmpa_tl } ,
2763     file     = \exp_not:o { \g_stex_currentfile_seq } ,
2764     lang     = \l_stex_module_lang_str ,
2765     sig      = \l_tmpa_str ,
2766     meta     = \l_tmpa_str ,
2767     feature  = #1 ,
2768   }
2769
2770   \stex_if_smsmode:TF {
2771     \stex_smsmode_set_codes:
2772   } {
2773     \begin{stex_annotate_env}{ feature:#1 }{}
2774     \stex_annotate_invisible:nnn{header}{}{ #3 }
2775   }
2776 }{
2777   \str_set:Nx \l_tmpa_str {
2778     c_stex_feature_
2779     \prop_item:Nn \l_stex_current_module_prop { ns } ?

```

```

2780     \prop_item:Nn \l_stex_current_module_prop { name }
2781     _prop
2782 }
2783 \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2784 \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2785 \stex_if_smsmode:TF {
2786     \exp_args:Nx \stex_add_to_sms:n {
2787         \prop_gset_from_keyval:cn {
2788             c_stex_feature_
2789             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2790             \prop_item:Nn \l_stex_current_module_prop { name }
2791             _prop
2792         } {
2793             origname = #2,
2794             name     = \prop_item:cn { \l_tmpa_str } { name } ,
2795             ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
2796             imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2797             constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2798             content  = \prop_item:cn { \l_tmpa_str } { content } ,
2799             file     = \prop_item:cn { \l_tmpa_str } { file } ,
2800             lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2801             sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2802             meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2803             feature  = \prop_item:cn { \l_tmpa_str } { feature }
2804         }
2805     }
2806 } {
2807     \end{stex_annotate_env}
2808 }
2809 }
2810

```

## structure

```

2811
2812 \prop_new:N \l_stex_all_structures_prop
2813
2814 \keys_define:nn { stex / features / structure } {
2815     name .str_set_x:N = \l__stex_features_structure_name_str ,
2816 }
2817
2818 \cs_new_protected:Nn \__stex_features_structure_args:n {
2819     \str_clear:N \l__stex_features_structure_name_str
2820     \keys_set:nn { stex / features / structure } { #1 }
2821 }
2822
2823 %\stex_new_feature:nnnn { structure } { 0{} m } {
2824 % \__stex_features_structure_args:n { ##1 }
2825 % \str_if_empty:NT \l__stex_features_structure_name_str {
2826 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2827 % }
2828 %} {
2829 %
2830 %}
2831

```

```

2832 \NewDocumentEnvironment{structure}{0}{m}{
2833   \__stex_features_structure_args:n { #1 }
2834   \str_if_empty:NT \l__stex_features_structure_name_str {
2835     \str_set:Nx \l__stex_features_structure_name_str { #2 }
2836   }
2837   \exp_args:Nnnx
2838   \begin{structural@feature}{ structure }
2839     { \l__stex_features_structure_name_str }{}
2840     \seq_clear:N \l_tmpa_seq
2841     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2842
2843   }{
2844     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2845     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2846     \str_set:Nx \l_tmpa_str {
2847       \prop_item:Nn \l_stex_current_module_prop { ns } ?
2848       \prop_item:Nn \l_stex_current_module_prop { name }
2849     }
2850     \seq_map_inline:Nn \l_tmpa_seq {
2851       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2852     }
2853     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2854     \exp_args:Nnx
2855     \AddToHookNext { env / structure / after }{
2856       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2857         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{}{}
2858       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2859       \STEXexport {
2860         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2861           {\prop_item:Nn \l_stex_current_module_prop { origname }}
2862           {\l_tmpa_str}
2863         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2864           {#2}{\l_tmpa_str}
2865         % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2866         %   \prop_item:Nn \l_stex_current_module_prop { origname },
2867         %   \l_tmpa_str
2868         % }
2869         % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2870         %   #2,\l_tmpa_str
2871         % }
2872         % \tl_set:cx { #2 } {
2873         %   \stex_invoke_structure:n { \l_tmpa_str }
2874         % }
2875       }
2876
2877   \end{structural@feature}
2878   % \g_stex_last_feature_prop
2879 }

```

\instantiate

```

2880 \seq_new:N \l__stex_features_structure_field_seq
2881 \str_new:N \l__stex_features_structure_field_str
2882 \str_new:N \l__stex_features_structure_def_tl
2883 \prop_new:N \l__stex_features_structure_prop

```

```

2884 \NewDocumentCommand \instantiate { m O{} m }{
2885   \stex_smsmode_set_codes:
2886   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2887   \prop_set_eq:Nc \l__stex_features_structure_prop {
2888     c_stex_feature_\l_tmpa_str _prop
2889   }
2890   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2891   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2892     \seq_set_split:Nnn \l_tmpa_seq={}{ ##1 }
2893     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2894       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2895       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2896         {!} \l_tmpa_tl
2897       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2898         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2899         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2900         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2901       }{
2902         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2903         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2904         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2905           \l_tmpa_tl
2906         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2907           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2908           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2909         }{
2910           \tl_clear:N \l_tmpb_tl
2911         }
2912       }
2913     }{
2914       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2915       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2916         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2917         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2918         \tl_clear:N \l_tmpa_tl
2919       }{
2920         % TODO throw error
2921       }
2922     }
2923     % \l_tmpa_str: name
2924     % \l_tmpa_tl: definiens
2925     % \l_tmpb_tl: notation
2926     \tl_if_empty:NT \l__stex_features_structure_field_str {
2927       % TODO throw error
2928     }
2929     \str_clear:N \l_tmpb_str
2930
2931     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2932     \seq_map_inline:Nn \l_tmpa_seq {
2933       \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2934       \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2935       \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2936         \seq_map_break:n {
2937           \str_set:Nn \l_tmpb_str { ####1 }

```

```

2938     }
2939   }
2940 }
2941 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2942   \l_tmpb_str
2943
2944 \tl_if_empty:NTF \l_tmpb_tl {
2945   \tl_if_empty:NF \l_tmpa_tl {
2946     \exp_args:Nx \use:n {
2947       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2948     }
2949   }
2950 }{
2951   \tl_if_empty:NTF \l_tmpa_tl {
2952     \exp_args:Nx \use:n {
2953       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2954     }
2955   }
2956 }{
2957   \exp_args:Nx \use:n {
2958     \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2959     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2960   }
2961 }
2962 }
2963 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2964 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2965 % #3/\l__stex_features_structure_field_str
2966 % \par
2967 % \expandafter\present\csname
2968 %   g_stex_symdecl_
2969 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2970 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2971 %   #3/\l__stex_features_structure_field_str
2972 %   _prop
2973 % \endcsname
2974 }
2975
2976 \tl_clear:N \l__stex_features_structure_def_tl
2977
2978 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2979 \seq_map_inline:Nn \l_tmpa_seq {
2980   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2981   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2982   \exp_args:Nx \use:n {
2983     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2984
2985     }
2986   }
2987 }
2988
2989 \prop_if_exist:cF {
2990   g_stex_symdecl_
2991   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2992   \prop_item:Nn \l_stex_current_module_prop {name} ?

```



```

2992     #3/\l_tmpa_str
2993     _prop
2994   }{
2995     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2996     \l_tmpb_str
2997     \exp_args:Nx \use:n {
2998       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2999     }
3000   }
3001 }
3002
3003 \symdecl*[type={\STEXsymbol{module-type}}{
3004   \_stex_term_math_oms:nnnn {
3005     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3006     \prop_item:Nn \l__stex_features_structure_prop {name}
3007     }{}{0}{}}
3008   ]]{#3}
3009
3010 % TODO: -> sms file
3011
3012 \tl_set:cx{ #3 }{
3013   \stex_invoke_structure:nnn {
3014     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3015     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3016   } {
3017     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3018     \prop_item:Nn \l__stex_features_structure_prop {name}
3019   }
3020 }
3021
3022 }

```

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

\stex\_invoke\_structure:nnn

```

3023 % #1: URI of the instance
3024 % #2: URI of the instantiated module
3025 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3026   \tl_if_empty:nTF{ #3 }{
3027     \prop_set_eq:Nc \l__stex_features_structure_prop {
3028       c_stex_feature_ #2 _prop
3029     }
3030     \tl_clear:N \l_tmpa_tl
3031     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3032     \seq_map_inline:Nn \l_tmpa_seq {
3033       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3034       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3035       \cs_if_exist:cT {
3036         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3037       }{
3038         \tl_if_empty:NF \l_tmpa_tl {
3039           \tl_put_right:Nn \l_tmpa_tl {,}
3040         }
3041         \tl_put_right:Nx \l_tmpa_tl {

```

```

3042         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3043     }
3044 }
3045 }
3046 \exp_args:No \mathstruct \l_tmpa_tl
3047 }{
3048     \stex_invoke_symbol:n{#1/#3}
3049 }
3050 }

```

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

```

3051 </package>

```

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

## Chapter 22

# STEX -Others Implementation

```
3052 <*package>
3053
3054 %%%%%%%%%% others.dtx %%%%%%%%%%
3055
3056 <@@=stex_others>
    Warnings and error messages
3057 % None

\MSC Math subject classifier

3058 \NewDocumentCommand \MSC {m} {
3059 % TODO
3060 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3061 \@ifpackageloaded{tikzinput}{
3062 \RequirePackage{stex-tikzinput}
3063 }{}
3064 </package>
```

## Chapter 23

# STEX -Metatheory Implementation

```
3065 <*package>
3066 <@@=stex_modules>
3067
3068 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3069
3070 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3071 \begingroup
3072 \stex_module_setup:nn{
3073   ns=\c_stex_metatheory_ns_str,
3074   meta=NONE
3075 }{Metatheory}
3076 \stex_reactivate_macro:N \symdecl
3077 \stex_reactivate_macro:N \notation
3078 \stex_reactivate_macro:N \symdef
3079 \ExplSyntaxOff
3080 \csname stex_suppress_html:n\endcsname{
3081   % is-a (a:A, a \in A, a is an A, etc.)
3082   \symdecl[args=ai]{isa}
3083   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3084   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3085   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3086
3087   % bind (\forall, \Pi, \lambda etc.)
3088   \symdecl[args=Bi]{bind}
3089   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3090   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3091   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; )}{#1 \comp, #2}
3092
3093   % dummy variable
3094   \symdecl{dummyvar}
3095   \notation[underscore]{dummyvar}{\comp\_}
3096   \notation[dot]{dummyvar}{\comp\cdot}
3097   \notation[dash]{dummyvar}{\comp{\rm --}}
3098
3099   %fromto (function space, Hom-set, implication etc.)
```

```

3100 \symdecl[args=ai]{fromto}
3101 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3102 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3103
3104 % mapto (lambda etc.)
3105 %\symdecl[args=Bi]{mapto}
3106 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3107 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3108 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3109
3110 % function/operator application
3111 \symdecl[args=ia]{apply}
3112 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3113 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3114
3115 % ‘‘type’’ of all collections (sets, classes, types, kinds)
3116 \symdecl{collection}
3117 \notation[U]{collection}{\comp{\mathcal{U}}}
3118 \notation[set]{collection}{\comp{\textsf{Set}}}
3119
3120 % sequences
3121 \symdecl[args=1]{seqtype}
3122 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3123
3124 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3125 \notation[ui]{sequence-index}{#1^{#2}}
3126
3127 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses},#1_{#3}}
3128 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses},#1^{#3}}
3129 % ^ superceded by \aseqfromto and \livar/\uivar
3130
3131 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3132 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
3133 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\comp,}#2\comp{\,\ellipses}
3134
3135 % letin (‘‘let’’, local definitions, variable substitution)
3136 \symdecl[args=bii]{letin}
3137 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3138 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3139 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3140
3141 % structures
3142 \symdecl*[args=1]{module-type}
3143 \notation{module-type}{\mathtt{MOD} #1}
3144 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3145 \notation[angle,prec=nobrackets]{mathstruct}{\comp\langle #1 \comp\rangle}{#1 \comp, #2}
3146
3147 }
3148 \ExplSyntaxOn
3149 \stex_add_to_current_module:n{
3150   \let\nappa\apply
3151   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3152   \def\livar{\csname sequence-index\endcsname[li]}
3153   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3154     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3155     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3156   }
3157   \__stex_modules_end_module:
3158   \endgroup
3159 </package>

```

## Chapter 24

# Tikzinput Implementation

```
3160 <*package>
3161
3162 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3163
3164 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3165 \RequirePackage{l3keys2e}
3166
3167 \keys_define:nn { tikzinput } {
3168   image .bool_set:N = \c_tikzinput_image_bool,
3169   image .default:n = false ,
3170 }
3171
3172 \ProcessKeysOptions { tikzinput }
3173
3174 \bool_if:NTF \c_tikzinput_image_bool {
3175   \RequirePackage{graphicx}
3176
3177   \providecommand\usetikzlibrary[]{}
3178   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
3179 }{
3180   \RequirePackage{tikz}
3181   \RequirePackage{standalone}
3182
3183   \newcommand \tikzinput [2] [] {
3184     \setkeys{Gin}{#1}
3185     \ifx \Gin@ewidth \Gin@exclamation
3186       \ifx \Gin@eheight \Gin@exclamation
3187         \input { #2 }
3188       \else
3189         \resizebox{!}{ \Gin@eheight }{
3190           \input { #2 }
3191         }
3192       \fi
3193     \else
3194       \ifx \Gin@eheight \Gin@exclamation
3195         \resizebox{ \Gin@ewidth }{!}{
3196           \input { #2 }
3197         }
3198       \fi
3199     \fi
3200   }
3201 }
```

```

3198     \else
3199         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3200             \input { #2 }
3201         }
3202     \fi
3203 \fi
3204 }
3205 }
3206
3207 \newcommand \ctikzinput [2] [] {
3208     \begin{center}
3209         \tikzinput [ #1 ] { #2 }
3210     \end{center}
3211 }
3212
3213 \@ifpackageloaded{stex}{
3214     \RequirePackage{stex-tikzinput}
3215 }{}
3216
3217 </package>
3218 <*stex>
3219 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3220 \RequirePackage{stex}
3221 \RequirePackage{tikzinput}
3222
3223 \newcommand\mhtikzinput[2] [] {%
3224     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3225     \stex_in_repository:nn\Gin@mhrepos{
3226         \tikzinput[ #1 ]{\mhp{##1}{#2}}
3227     }
3228 }
3229 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[ #1 ]{ #2 }\end{center}}
3230 </stex>

```



## Chapter 25

# document-structure.sty Implementation

### 25.1 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.

### 25.2 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`

```
3231 \*cls)
3232 \RequirePackage{etoolbox}
3233 \RequirePackage{kvoptions}
3234 \SetupKeyvalOptions{family=omdoc@cls,prefix=omdoc@cls@}
3235 \DeclareStringOption[article]{class}
3236 \AddToKeyvalOption*{class}{\PassOptionsToPackage{class=\omdoc@cls@class}{omdoc}}

the following options are deprecated.

3237 \DeclareVoidOption{report}{\def\omdoc@cls@class{report}}%
3238 \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}}
3239 \DeclareVoidOption{book}{\def\omdoc@cls@class{book}}%
3240 \ClassWarning{omdoc}{the option 'part' is deprecated, use 'class=book', instead}}
3241 \DeclareVoidOption{bookpart}{\def\omdoc@cls@class{book}}%
3242 \PassOptionsToPackage{topsect=chapter}{omdoc}%
3243 \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter',
3244 \DeclareBoolOption{minimal}}
```

the rest of the options are only passed on to `omdoc.sty` and the class selected by the first options. We need to load the `etoolbox` package early for `\@xappto`.

```
3245 \def\@omdoc@cls@doopt{}
3246 \DeclareDefaultOption{%
```

```

3247 \ifx\@omdoc@cls@dopt\@empty%
3248 \xdef\@omdoc@cls@dopt{\CurrentOption}%
3249 \else\xappto\@omdoc@cls@dopt{,\CurrentOption}%
3250 \fi}%
3251 \PassOptionsToPackage{\CurrentOption}{omdoc}
3252 \ProcessKeyvalOptions{omdoc@cls}

```

We load `article.cls`, and the desired packages. For the L<sup>A</sup>T<sub>E</sub>XML bindings, we make sure the right packages are loaded.

```

3253 \LoadClass[\@omdoc@cls@dopt]{\@omdoc@cls@class}

```

## 25.3 Beefing up the document environment

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

```

3254 \ifomdoc@cls@minimal\else%
3255 \RequirePackage{omdoc}

```

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<sup>A</sup>T<sub>E</sub>X level, but the document identifier is picked up by L<sup>A</sup>T<sub>E</sub>XML.<sup>8</sup>

```

3256 \srefaddidkey{document}
3257 \newcommand\documentkeys[1]{\metasetkeys{document}{#1}}
3258 \let\orig@document=\document
3259 \renewcommand{\document}[1] [] {\metasetkeys{document}{#1}\orig@document}

```

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

```

3260 \fi% \ifomdoc@cls@minimal
3261 \</cls>

```

## 25.4 Implementation: OMDoc Package

## 25.5 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).

```

3262 <*package>
3263 \RequirePackage{kvoptions}
3264 \SetupKeyvalOptions{family=omdoc@sty,prefix=omdoc@sty@}
3265 \DeclareStringOption[article]{class}
3266 \DeclareBoolOption{showignores}
3267 \DeclareStringOption[section]{topsect}
3268 \newcount\section@level
3269 \DeclareDefaultOption{\PassOptionsToPackage{\CurrentOption}{sref}}
3270 \ProcessKeyvalOptions{omdoc@sty}

```

---

<sup>8</sup>EDNOTE: faking `documentkeys` for now. @HANG, please implement

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

```

3271 \RequirePackage{stex-metakeys}
3272 %\RequirePackage{sref}
3273 \RequirePackage{xspace}
3274 \RequirePackage{comment}
3275 %\RequirePackage{pathsuris}
3276 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}
3277
3278 \def\srefaddidkey#1{\addmetakey{#1}{id}}
3279

```

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

```

3280 \ExplSyntaxOn
3281 \@ifpackageloaded{babel}{
3282   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3283   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3284     \input{omdoc-ngerman.ldf}
3285   }
3286 }{}
3287 \ExplSyntaxOff
3288 %\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.

```

3289 \section@level=2
3290 \ifdefstring{\omdoc@sty@class}{book}{\section@level=0}{}
3291 \ifdefstring{\omdoc@sty@class}{report}{\section@level=0}{}
3292 \ifdefstring{\omdoc@sty@topsect}{part}{\section@level=0}{}
3293 \ifdefstring{\omdoc@sty@topsect}{chapter}{\section@level=1}{}

```

## 25.6 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  $\text{\LaTeX}$  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.<sup>9</sup>

```

3294 \def\current@section@level{document}%
3295 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
3296 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipomgroup`

```

3297 \newcommand\skipomgroup{%
3298   \ifcase\section@level%

```

---

<sup>9</sup>EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

3299 \or\stepcounter{chapter}%
3300 \or\stepcounter{section}%
3301 \or\stepcounter{subsection}%
3302 \or\stepcounter{subsubsection}%
3303 \or\stepcounter{paragraph}%
3304 \or\stepcounter{subparagraph}%
3305 \fi}% \ifcase

```

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

**blindomgroup**

```

3306 \newcommand\at@begin@blindomgroup[1]{%
3307 \newenvironment{blindomgroup}
3308 {\advance\section@level by 1\at@begin@blindomgroup\section@level}
3309 {\advance\section@level by -1}

```

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

```

3310 \newcommand\omgroup@nonum[2]{%
3311 \ifx\hyper@anchor\@undefined\else\phantomsection\fi%
3312 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}}

```

(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.

```

3313 \newcommand\omgroup@num[2]{%
3314 \ifx\omgroup@short\@empty% no short title
3315 \@nameuse{#1}{#2}%
3316 \else% we have a short title
3317 \@ifundefined{rdfmata@sectioning}%
3318 {\@nameuse{#1}[\omgroup@short]{#2}}%
3319 {\@nameuse{rdfmata@#1@old}[\omgroup@short]{#2}}%
3320 \fi%
3321 %\sref@label@id@arg{\@nameuse{the#1}}\omgroup@id
3322 }

```

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

**omgroup**

```

3323 \def\@true{true}
3324 \def\@false{false}
3325 \srefaddidkey{omgroup}
3326 \addmetakey{omgroup}{date}
3327 \addmetakey{omgroup}{creators}
3328 \addmetakey{omgroup}{contributors}
3329 \addmetakey{omgroup}{srccite}
3330 \addmetakey{omgroup}{type}
3331 \addmetakey*{omgroup}{short}
3332 \addmetakey*{omgroup}{display}
3333 \addmetakey*{omgroup}{intro}% ignored
3334 \addmetakey[false]{omgroup}{loadmodules}[true]

```

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.

```
3335 \newif\if@mainmatter\@mainmattertrue
3336 \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.

```
3337 \addmetakey{omdoc@sect}{name}
3338 \addmetakey[false]{omdoc@sect}{clear}[true]
3339 \addmetakey{omdoc@sect}{ref}
3340 \addmetakey[false]{omdoc@sect}{num}[true]
3341 \newcommand\omdoc@sectioning[3] [] {\metasetkeys{omdoc@sect}{#1}%
3342 \ifx\omdoc@sect@clear\@true\cleardoublepage\fi%
3343 \if@mainmatter% numbering not overridden by frontmatter, etc.
3344 \ifx\omdoc@sect@num\@true\omgroup@num{#2}{#3}\else\omgroup@nonum{#2}{#3}\fi%
3345 \def\current@section@level{\omdoc@sect@name}%
3346 \else\omgroup@nonum{#2}{#3}%
3347 \fi}% if@mainmatter
```

and another one, if redefines the `\addtocontentsline` macro of L<sup>A</sup>T<sub>E</sub>X to import the respective macros. It takes as an argument a list of module names.

```
3348 \newcommand\omgroup@redefine@addtocontents[1] {%
3349 %\edef\@import{#1}%
3350 %\@for\@I:=\@import\do{%
3351 %\edef\@path{\csname module@\@I @path\endcsname}%
3352 %\@ifundefined{tf@toc}\relax%
3353 % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
3354 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
3355 %\def\addcontentsline##1##2##3{%
3356 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
3357 %\else% hyperref.sty not loaded
3358 %\def\addcontentsline##1##2##3{%
3359 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
3360 %\fi
3361 }% 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.

```
3362 \newcount\omgroup@level
3363 \newenvironment{omgroup}[2] [] % keys, title
3364 {\metasetkeys{omgroup}{#1}%\sref@target%
3365 \advance\omgroup@level by 1\relax%
```

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.

```
3366 \ifx\omgroup@loadmodules\@true%
3367 \omgroup@redefine@addtocontents{\@ifundefined{module@id}\used@modules%
3368 {\@ifundefined{module@\module@id @path}{\used@modules}\module@id}}\fi%
```

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

```
3369 \advance\section@level by 1\relax%
3370 \ifcase\section@level%
```

```

3371 \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}%
3372 \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}%
3373 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}%
3374 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}%
3375 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}%
3376 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}%
3377 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragraph}
3378 \fi% \ifcase
3379 \at@begin@omgroup[#1]\section@level{#2}
3380 \csname stex_ref_new_doc_target:n\endcsname\omgroup@id%
3381 }% for customization
3382 {\advance\section@level by -1\advance\omgroup@level by -1}

and finally, we localize the sections
3383 \newcommand\omdoc@part@kw{Part}
3384 \newcommand\omdoc@chapter@kw{Chapter}
3385 \newcommand\omdoc@section@kw{Section}
3386 \newcommand\omdoc@subsection@kw{Subsection}
3387 \newcommand\omdoc@subsubsection@kw{Subsubsection}
3388 \newcommand\omdoc@paragraph@kw{paragraph}
3389 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 25.7 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`

```

3390 \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).

```

3391 \ifcsdef{frontmatter}% to redefine if necessary
3392   {\cslet{orig@frontmatter}{\frontmatter}\cslet{frontmatter}{\relax}}
3393   {\cslet{orig@frontmatter}{\clearpage\@mainmatterfalse\pagenumbering{roman}}}
3394 \ifcsdef{backmatter}% to redefine if necessary
3395   {\cslet{orig@backmatter}{\backmatter}\cslet{backmatter}{\relax}}
3396   {\cslet{orig@backmatter}{\clearpage\@mainmatterfalse\pagenumbering{roman}}}

```

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.

```

3397 \newenvironment{frontmatter}
3398 {\orig@frontmatter}
3399 {\ifcsdef{mainmatter}{\mainmatter}{\clearpage\@mainmattertrue\pagenumbering{arabic}}}

```

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

```

3400 \newenvironment{backmatter}
3401 {\orig@backmatter}
3402 {\ifcsdef{mainmatter}{\mainmatter}{\clearpage\@mainmattertrue\pagenumbering{arabic}}}

```

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

```
3403 \@mainmattertrue\pagenumbering{arabic}
```

## 25.8 Ignoring Inputs

ignore

```
3404 \ifomdoc@sty@showignores
3405 \addmetakey{ignore}{type}
3406 \addmetakey{ignore}{comment}
3407 \newenvironment{ignore}[1] []
3408 {\metasetkeys{ignore}{#1}\textless\ignore@type\textgreater\bgroup\itshape}
3409 {\egroup\textless\ignore@type\textgreater}
3410 \renewenvironment{ignore}{}{}\else\excludecomment{ignore}\fi
```

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

```
3411 \newcommand\afterprematurestop{}
3412 \def\prematurestop@endomgroup{\ifnum\omgroup@level=0\else%
3413 \end{omgroup}\advance\omgroup@level by -1\prematurestop@endomgroup\fi}
3414 \providecommand\prematurestop{%
3415 \message{Stopping sTeX processing prematurely}
3416 \prematurestop@endomgroup\afterprematurestop
3417 \end{document}}
```

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

## 25.9 Structure Sharing

10

```
3418 \providecommand{\lxDocumentID}[1]{}%
3419 \def\LXMID#1#2{\expandafter\gdef\csname xmarg#1\endcsname{#2}\csname xmarg#1\endcsname}
3420 \def\LXMRef#1{\csname xmarg#1\endcsname}
```

`\STRlabel` The main macro, it is used to attach a label to some text expansion. Later on, using the `\STRcopy` macro, the author can use this label to get the expansion originally assigned.

```
3421 \long\def\STRlabel#1#2{\STRlabeldef{#1}{#2}{#2}}
```

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

`\STRcopy` The `\STRcopy` macro is used to call the expansion of a given label. In case the label is not defined it will issue a warning.<sup>11</sup>

```
3422 \newcommand\STRcopy[2] []{\expandafter\ifx\csname STR@#2\endcsname\relax
3423 \message{STR warning: reference #2 undefined!}
3424 \else\csname STR@#2\endcsname\fi}
```

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

<sup>10</sup>EdNOTE: The following is simply copied over from the `latexml` package, which we eliminated, we should integrate better.

<sup>11</sup>EdNOTE: MK: we need to do something about the ref!

`\STRsemantics` if we have a presentation form and a semantic form, then we can use

```

3425 \newcommand\STRsemantics[3] [] {#2\def\@test{#1}\ifx\@test\empty\STRlabeldef{#1}{#2}\fi}

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

```

`\STRlabeldef` This is the macro that does the actual labeling. Is it called inside `\STRlabel`

```

3426 \def\STRlabeldef#1{\expandafter\gdef\csname STR@#1\endcsname}

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

```

## 25.10 Global Variables

`\setSGvar` set a global variable

```

3427 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

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

```

`\useSGvar` use a global variable

```

3428 \newrobustcmd\useSGvar[1]{%
3429   \@ifundefined{sTeX@Gvar@#1}
3430   {\PackageError{omdoc}
3431     {The sTeX Global variable #1 is undefined}
3432     {set it with \protect\setSGvar}}
3433   \@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.

```

3434 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
3435   \@ifundefined{sTeX@Gvar@#1}
3436   {\PackageError{omdoc}
3437     {The sTeX Global variable #1 is undefined}
3438     {set it with \protect\setSGvar}}
3439   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

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

```

## 25.11 Colors

blue, red, green, magenta We will use the following abbreviations for colors from `color.sty`

```

3440 \def\black#1{\textcolor{black}{#1}}
3441 \def\gray#1{\textcolor{gray}{#1}}
3442 \def\blue#1{\textcolor{blue}{#1}}
3443 \def\red#1{\textcolor{red}{#1}}
3444 \def\green#1{\textcolor{green}{#1}}
3445 \def\cyan#1{\textcolor{cyan}{#1}}
3446 \def\magenta#1{\textcolor{magenta}{#1}}
3447 \def\brown#1{\textcolor{brown}{#1}}
3448 \def\yellow#1{\textcolor{yellow}{#1}}
3449 \def\orange#1{\textcolor{orange}{#1}}
3450 \end{package}

```



## Chapter 26

# MiKoSlides – Implementation

### 26.1 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.

```
3451 <*cls>
3452 %\RequirePackage{modules}
3453 \RequirePackage{kvoptions}
3454 \RequirePackage{stex-metakeys}
3455 \RequirePackage{etoolbox}
3456 \SetupKeyvalOptions{family=mks@cls,prefix=mks@cls@}
3457 \DeclareStringOption[article]{class}
3458 \AddToKeyvalOption*{class}{\PassOptionsToClass{class=\mks@cls@class}{omdoc}
3459 \ifdefstring{\mks@cls@class}{book}{\PassOptionsToPackage{defaulttopsect=part}{mikoslides}}
3460 \ifdefstring{\mks@cls@class}{report}{\PassOptionsToPackage{defaulttopsect=part}{mikoslides}}
3461 \DeclareBoolOption{notes}
3462 \DeclareComplementaryOption{slides}{notes}
3463 \DeclareDefaultOption{%
3464 \PassOptionsToClass{\CurrentOption}{omdoc}
3465 \PassOptionsToClass{\CurrentOption}{beamer}
3466 \PassOptionsToPackage{\CurrentOption}{mikoslides}}
3467 \ProcessKeyvalOptions{mks@cls}
3468 </cls>
```

now we do the same for the `mikoslides` package.

```
3469 <*package>
3470 %\RequirePackage{stex-base}
3471 \RequirePackage{kvoptions}
3472 \RequirePackage{stex-metakeys}
3473 \SetupKeyvalOptions{family=mks@sty,prefix=mks@sty@}
3474 \DeclareStringOption{topsect}
3475 \DeclareStringOption{defaulttopsect}
3476 \newif\ifnotes\notesttrue
3477 \DeclareBoolOption{notes}
3478 \AddToKeyvalOption*{notes}{\notesttrue}%\PassOptionsToPackage{notes}{statements}}
3479 \DeclareComplementaryOption{slides}{notes}
```

```

3480 \AddToKeyvalOption*{slides}{\notesfalse}%\PassOptionsToPackage{nontheorem}{statements}}
3481 \DeclareBoolOption{sectocframes}
3482 \DeclareBoolOption{frameimages}
3483 \DeclareBoolOption{fiboxed}
3484 \DeclareBoolOption{nopproblems}
3485 %\DeclareDefaultOption{
3486 %\PassOptionsToPackage{\CurrentOption}{stex}
3487 %\PassOptionsToPackage{\CurrentOption}{smglom}
3488 %\PassOptionsToPackage{\CurrentOption}{tikzinput}}
3489 \ProcessKeyvalOptions{mks@sty}

```

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

```

3490 \ifx\mks@sty@topsect\@empty\edef\@@topsect{\mks@sty@defaulttopsect}
3491 \else\edef\@@topsect{\mks@sty@topsect}\fi
3492 \</package>

```

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

```

3493 \<cls>
3494 \ifmks@cls@notes
3495 \LoadClass{omdoc}
3496 \else
3497 \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
3498 \newcounter{Item}
3499 \newcounter{paragraph}
3500 \newcounter{subparagraph}
3501 \newcounter{Hfootnote}
3502 \fi

```

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

```

3503 \RequirePackage{mikoslides}
3504 \</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 `TEX` packages. The first batch of packages we want are loaded on `mikoslides.sty`. These are the general ones, we will load the `TEX`-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.

```

3505 \<package>
3506 %\RequirePackage{stex}
3507 \RequirePackage{stex-compatibility}
3508 \ifmks@sty@notes
3509 \RequirePackage{a4wide}
3510 \RequirePackage{marginnote}
3511 \RequirePackage[dvipsnames,svgnames]{xcolor}
3512 \RequirePackage{mdframed}
3513 \RequirePackage[noxcolor,noamsthm]{beamerarticle}
3514 \fi
3515 \RequirePackage{etoolbox}
3516 \RequirePackage{amssymb}
3517 \RequirePackage{amsmath}
3518 \RequirePackage{comment}
3519 \RequirePackage{textcomp}

```

```

3520 \RequirePackage{url}
3521 \RequirePackage{graphicx}
3522 \RequirePackage{pgf}
3523 %\RequirePackage{omtext}
3524 \ifmks@sty@notes
3525 \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
3526 \fi

```

finally, we require the `metakeys` package from `TeX`, so that we can use the `\addmetakey` mechanism.

```

3527 %\RequirePackage{metakeys}

```

## 26.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.<sup>12</sup>

```

3528 \ifmks@sty@notes
3529 \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]{}
3530 \fi

```

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

```

3531 \newcounter{slide}
3532 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
3533 \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.

```

3534 \ifmks@sty@notes%
3535 \renewenvironment{note}{\ignorespaces}{}%
3536 \else%
3537 \excludecomment{note}%
3538 \fi%

```

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.

```

3539 \ifmks@sty@notes
3540 \newlength{\slideframewidth}
3541 \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

3542 \addmetakey{frame}{label}
3543 \addmetakey[yes]{frame}{allowframebreaks}
3544 \addmetakey{frame}{allowdisplaybreaks}
3545 \addmetakey[yes]{frame}{fragile}
3546 \addmetakey[yes]{frame}{shrink}
3547 \addmetakey[yes]{frame}{squeeze}
3548 \addmetakey[yes]{frame}{t}

```

<sup>12</sup>EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

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

```

3549 \renewenvironment{frame}[1][]{%
3550 \metasetkeys{frame}{#1}%
3551 \sffamily%
3552 \stepcounter{slide}%
3553 \def\@currentlabel{\theslide}%
3554 \ifx\frame@label\@empty\else\label{\frame@label}\fi%

```

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

```

3555 \def\itemize@level{outer}%
3556 \def\itemize@outer{outer}%
3557 \def\itemize@inner{inner}%
3558 \renewcommand\newpage{\addtocounter{framenumber}{1}}%
3559 \renewcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}%
3560 \renewenvironment{itemize}{%
3561 \ifx\itemize@level\itemize@outer%
3562 \def\itemize@label{$\rhd$}%
3563 \fi%
3564 \ifx\itemize@level\itemize@inner%
3565 \def\itemize@label{$\scriptstyle\rhd$}%
3566 \fi%
3567 \begin{list}%
3568 {\itemize@label}%
3569 {\setlength{\labelsep}{.3em}%
3570 \setlength{\labelwidth}{.5em}%
3571 \setlength{\leftmargin}{1.5em}%
3572 }%
3573 \edef\itemize@level{\itemize@inner}%
3574 }{%
3575 \end{list}%
3576 }%

```

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

```

3577 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
3578 }{%
3579 \medskip\miko@slidelabel\end{mdframed}%
3580 }%

```

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

`\frametitle`

```

3581 \renewcommand{\frametitle}[1]{\Large\bf\sffamily\color{blue}{#1}\medskip}%
3582 \fi %ifmks@sty@notes

```

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

`\pause` 13

```

3583 \ifmks@sty@notes\newcommand\pause{}\fi

```

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

`nomtext`

```

3584 \ifmks@sty@notes\newenvironment{nomtext}[1]{}{\begin{omtext}[#1]}{\end{omtext}}%
3585 \else\excludecomment{nomtext}\fi%

```

---

<sup>13</sup>EdNOTE: MK: fake it in notes mode for now

```

nomgroup
3586 \ifmks@sty@notes\newenvironment{nomgroup}[2][\begin{omgroup}[#1]{#2}}{\end{omgroup}}%
3587 \else\excludecomment{nomgroup}\fi%

ndefinition
3588 \ifmks@sty@notes\newenvironment{ndefinition}[1][\begin{definition}[#1]]{\end{definition}}%
3589 \else\excludecomment{ndefinition}\fi%

nassertion
3590 \ifmks@sty@notes\newenvironment{nassertion}[1][\begin{assertion}[#1]]{\end{assertion}}%
3591 \else\excludecomment{nassertion}\fi%

nsproof
3592 \ifmks@sty@notes\newenvironment{nsproof}[2][\begin{sproof}[#1]{#2}}{\end{sproof}}%
3593 \else\excludecomment{nsproof}\fi%

nexample
3594 \ifmks@sty@notes\newenvironment{nexample}[1][\begin{example}[#1]]{\end{example}}%
3595 \else\excludecomment{nexample}\fi%

\inputref@*skip We customize the hooks for in \inputref.
3596 \def\inputref@preskip{\smallskip}
3597 \def\inputref@postskip{\medskip}

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

\inputref*
3598 \let\orig@inputref\inputref
3599 \def\inputref{\@ifstar\ninputref\orig@inputref}
3600 \newcommand\ninputref[2][\ifmks@sty@notes\orig@inputref[#1]{#2}\fi}

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

```

## 26.3 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>}.
3601 \newlength{\slidelogoheight}
3602 \ifmks@sty@notes%
3603 \setlength{\slidelogoheight}{.4cm}%
3604 \else%
3605 \setlength{\slidelogoheight}{1cm}%
3606 \fi%
3607 \newsavebox{\slidelogo}%
3608 \sbox{\slidelogo}{\TeX}%
3609 \newrobustcmd{\setslidelogo}[1]{%
3610 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}%
3611 }%

(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.

```
3612 \def\source{Michael Kohlhase}% customize locally
3613 \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.

```
3614 \def\copyrightnotice{\footnotesize\copyright:\hspace{.3ex}{\source}}%
3615 \newsavebox{\cclogo}%
3616 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}%
3617 \newif\ifcchref\cchreffalse%
3618 \AtBeginDocument{%
3619   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
3620 }%
3621 \def\licensing{%
3622   \ifcchref%
3623     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}%
3624   \else%
3625     {\usebox{\cclogo}}%
3626   \fi%
3627 }%
3628 \newrobustcmd{\setlicensing}[2][{}]{%
3629   \def@url{#1}%
3630   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}%
3631   \ifx@url@empty%
3632     \def\licensing{{\usebox{\cclogo}}}%
3633   \else%
3634     \def\licensing{%
3635       \ifcchref%
3636         \href{#1}{\usebox{\cclogo}}%
3637       \else%
3638         {\usebox{\cclogo}}%
3639       \fi%
3640     }%
3641   \fi%
3642 }%
```

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

EdN:14

`\slidelabel` Now, we set up the slide label for the article mode.<sup>14</sup>

```
3643 \newrobustcmd\miko@slidelabel{%
3644   \vbox to \slidelogoheight{%
3645     \vss\hbox to \slidewidth%
3646     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}%
3647   }%
3648 }%
3649 % \subsection{Frame Images}\label{sec:impl:frameimage}
3650 %
```

---

<sup>14</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

3651 % \begin{macro}{\frameimage}
3652 % We have to make sure that the width is overwritten, for that we check the
3653 % |\Gin@ewidth| macro from the |graphicx| package. We also add the |label| key.
3654 % \begin{macrocode}
3655 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
3656 \newrobustcmd\frameimage[2] [] {%
3657   \stepcounter{slide}%
3658   \ifmks@sty@frameimages%
3659     \def\Gin@ewidth{}\setkeys{Gin}{#1}%
3660     \ifmks@sty@notes\else\vfill\fi%
3661     \begin{center}
3662       \ifmks@sty@fiboxed%
3663         \fbox{\ifx\Gin@ewidth\@empty\includegraphics[width=\slidewidth,#1]{#2}\else\mygraphics[width=\slidewidth,#1]{#2}\fi}
3664       \else
3665         \ifx\Gin@ewidth\@empty\includegraphics[width=\slidewidth,#1]{#2}\else\mygraphics[width=\slidewidth,#1]{#2}\fi
3666       \fi% \ifmks@sty@fiboxed
3667     \end{center}
3668     \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
3669     \ifmks@sty@notes\else\vfill\fi%
3670   \fi} % \ifmks@sty@frameimages

```

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

`\mhframeimage` Use the current value of `\mh@currentrepos` or the value of the `mhrepos` key if it is given in `\frameimage`.

```

3671 \def\Gin@mhrepos{}
3672 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
3673 \newcommand\mhframeimage[2] [] {%
3674   \setkeys{Gin}{#1}%
3675   \edef\mh@crepos{\mh@currentrepos}%
3676   \ifx\Gin@mhrepos\@empty%
3677     \edef\temp@path{\MathHub{\mh@currentrepos/source/#2}}%
3678   \else%
3679     \edef\temp@path{\MathHub{\Gin@mhrepos/source/#2}}%
3680   \fi%
3681   \if@iswindows@ \path@to@windows\temp@path\fi%
3682   \frameimage[#1]{\temp@path}%
3683 }%

```

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

## 26.4 Colors and Highlighting

We first specify sans serif fonts as the default.

```

3684 \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

```

3685 \AtBeginDocument{%
3686   \definecolor{green}{rgb}{0,.5,0}%
3687   \definecolor{purple}{cmyk}{.3,1,0,.17}%
3688 }%

```

We customize the `\defemph`, `\notemph`, and `\stDMemph` macros with colors for the use in the `statements` package. Furthermore we customize the `\@@lec` macro for the appearance of line end comments in `\lec`.

```

3689 % \def\STpresent#1{\textcolor{blue}{#1}}
3690 \def\defemph#1{\textcolor{magenta}{#1}}
3691 \def\termemph#1{\textcolor{cyan}{#1}}
3692 \def\notemph#1{\textcolor{magenta}{#1}}
3693 \def\stDMemph#1{\textcolor{blue}{#1}}
3694 \def\@@lec#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.

```

3695 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
3696 \def\smalltextwarning{%
3697   \pgfuseimage{miko@small@dbend}%
3698   \xspace%
3699 }%
3700 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
3701 \newrobustcmd\textwarning{%
3702   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}%
3703   \xspace%
3704 }%
3705 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}%
3706 \newrobustcmd\bigtextwarning{%
3707   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}%
3708   \xspace%
3709 }%

```

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

```

3710 \newrobustcmd\putgraphicsat[3]{%
3711   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}%
3712 }%
3713 \newrobustcmd\putat[2]{%
3714   \begin{picture}(0,0)\put(#1){#2}\end{picture}%
3715 }%

```

## 26.5 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`.

```

3716 \ifmks@sty@sectocframes%
3717 \ifdefstring\@@topsect{part}{%
3718   \newcounter{chapter}\counterwithin*{section}{chapter}}
3719 {\ifdefstring\@@topsect{chapter}{\newcounter{chapter}\counterwithin*{section}{chapter}}{}}
3720 \fi% ifsectocframes

```

Now that we have defined the counters, we can load the `TEX`-specific packages (in particular `statements` that needs these counters).

```

3721 \RequirePackage{tikzinput}

```



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

`\section@level`

```

3722 \section@level=2
3723 \def\part@prefix{}
3724 \ifdefstring{\@@topsect}{part}
3725 {\section@level=0%
3726   \def\thesection{\arabic{chapter}.\arabic{section}}}%
3727   \def\part@prefix{\arabic{chapter}.}}{}
3728 \ifdefstring{\@@topsect}{chapter}
3729 {\section@level=1%
3730   \def\thesection{\arabic{chapter}.\arabic{section}}}%
3731   \def\part@prefix{\arabic{chapter}.}}{}
3732 \ifmks@sty@notes\else% only in slides

```

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

The new counters are used in the `omgroup` environment that chooses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to `\section@level`.

`omgroup`

```

3733 \renewenvironment{omgroup}[2][]{%
3734   \metasetkeys{omgroup}{#1}%
3735   \advance\section@level by 1%
3736   \advance\omgroup@level by 1%
3737   \ifmks@sty@sectocframes%
3738   \stepcounter{slide}
3739   \begin{frame}[noframenumbering]%
3740   \vfill\Large\centering%
3741   \red{%
3742     \ifcase\section@level\or
3743     \stepcounter{part}
3744     \def\@@label{\omdoc@part@kw~\Roman{part}}
3745     \def\currentsectionlevel{\omdoc@part@kw}
3746     \or%
3747     \stepcounter{chapter}
3748     \def\@@label{\omdoc@chapter@kw~\arabic{chapter}}
3749     \def\currentsectionlevel{\omdoc@chapter@kw}
3750     \or
3751     \stepcounter{section}
3752     \def\@@label{\part@prefix\arabic{section}}
3753     \def\currentsectionlevel{\omdoc@section@kw}
3754     \or
3755     \stepcounter{subsection}
3756     \def\@@label{\part@prefix\arabic{section}.\arabic{subsection}}
3757     \def\currentsectionlevel{\omdoc@subsection@kw}
3758     \or
3759     \stepcounter{subsubsection}
3760     \def\@@label{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
3761     \def\currentsectionlevel{\omdoc@subsubsection@kw}
3762     \or
3763     \stepcounter{mparagraph}
3764     \def\@@label{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{subsubsection}.\arabic{mparagraph}}
3765     \def\currentsectionlevel{\omdoc@paragraph@kw}

```

```

3766 \fi% end ifcase
3767 \@@label\sref@label@id\@@label
3768 \quad #2%
3769 }%
3770 \vfill%
3771 \end{frame}%
3772 \fi %ifmks@sty@sectocframes
3773 \csname stex_ref_new_doc_target:n\endcsname\omgroup@id%
3774 }
3775 {\advance\section@level by -1}%
3776 \fi% ifmks@sty@notes

```

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

```

3777 \def\inserttheorembodyfont{\normalfont}
3778 \ifmks@sty@notes\else% only in slides
3779 \defbeamertemplate{theorem begin}{miko}
3780 {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
3781 \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
3782 \inserttheorempunctuation\inserttheorembodyfont\xspace}
3783 \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

3784 \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.

```

3785 \expandafter\def\csname Parent2\endcsname{}
3786 \fi% ifmks@sty@notes
3787 \ifmks@sty@notes%
3788 \renewenvironment{columns}[1][\]{%
3789 \par\noindent%
3790 \begin{minipage}%
3791 \slidewidth\centering\leavevmode%
3792 }{%
3793 \end{minipage}\par\noindent%
3794 }%
3795 \newsavebox\columnbox%
3796 \renewenvironment<>{column}[2][\]{%
3797 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
3798 }{%
3799 \end{minipage}\end{lrbox}\usebox\columnbox%
3800 }%
3801 \fi% ifmks@sty@notes
3802 \ifmks@sty@nopproblems%
3803 \newenvironment{problems}{}{}%
3804 \else%
3805 \excludcomment{problems}%
3806 \fi%

```

## 26.6 Excursions

**\excursion\*** The excursion macros are very simple, we define a new internal macro in `\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.

```

3807 \gdef\printexcursions{}
3808 \newcommand\activateexcursion[2] [] {\gappto\printexcursions{\inputref[#1]{#2}}}
3809 \newcommand\excursionref[2] {% label, text
3810 \ifnotes\begin{omtext}[title=Excursion]#2 \sref[fallback=the appendix]{#1}.\end{omtext}\fi}
3811 \newcommand\excursion[4] [] {% opt, label, path, text
3812 \ifnotes\activateexcursion[#1]{#3}\excursionref{#2}{#4}\fi}

```

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

\excursiongroup

```

3813 \srefaddidkey{excursiongroup}%
3814 \addmetakey{excursiongroup}{intro}%
3815 \newcommand\excursiongroup[1] [] {%
3816 \metasetkeys{excursiongroup}{#1}%
3817 \ifdefempty\printexcursions{}% only if there are excursions
3818 {\begin{omgroup}[#1]{Excursions}%
3819 \ifdefempty\excursiongroup@intro{\inputref{\excursiongroup@intro}}%
3820 \printexcursions%
3821 \end{omgroup}}}

```

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

\mhexcursion

```

3822 \newcommand\activatemhexcursion[2] [] {\ifstrempy{#1}%
3823 {\gappto\printexcursions{\mhinputref[\mh@currentrepos]{#2}}}%
3824 {\gappto\printexcursions{\mhinputref[#1]{#2}}}}
3825 \newcommand\mhexcursion[4] [] {% repos, label, path, text
3826 \ifnotes\activatemhexcursion[#1]{#3}\excursionref{#2}{#4}\fi}%

```

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

\mhexcursiongroup

```

3827 \srefaddidkey{mhexcursiongroup}%
3828 \addmetakey{mhexcursiongroup}{intro}%
3829 \addmetakey{mhexcursiongroup}{mhrepos}%
3830 \newcommand\mhexcursiongroup[1] [] {%
3831 \metasetkeys{mhexcursiongroup}{#1}%
3832 \ifdefempty\printexcursions{}% only if there are excursions
3833 {\begin{omgroup}[#1]{Excursions}%
3834 \ifdefempty\mhexcursiongroup@intro{%
3835 {\ifdefempty\mhexcursiongroup@mhrepos%
3836 {\mhinputref{\mhexcursiongroup@intro}}%
3837 {\mhinputref[\mhexcursiongroup@mhrepos]{\mhexcursiongroup@intro}}}%
3838 \printexcursions%
3839 \end{omgroup}}}
3840 \end{package}

```

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

## 26.7 Miscellaneous