

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

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

2021-12-11

## **Abstract**

TODO

---

\*Version 3.0 (last revised 2021-12-11)

# 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 L<sup>A</sup>T<sub>E</sub>X-Basics Implementation</b>	<b>45</b>
14.1 The L <sup>A</sup> T <sub>E</sub> XDocument 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 L<sup>A</sup>T<sub>E</sub>X-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 L<sup>A</sup>T<sub>E</sub>X-References Implementation</b>	<b>63</b>
16.1 Document URIs and URLs	63
16.2 Setting Reference Targets	65
16.3 Using References	66

<b>17</b>	<b>sTeX-Modules Implementation</b>	<b>67</b>
17.1	The module environment . . . . .	70
17.2	Invoking modules . . . . .	75
<b>18</b>	<b>sTeX-Module Inheritance Implementation</b>	<b>77</b>
18.1	SMS Mode . . . . .	77
18.2	Inheritance . . . . .	81
<b>19</b>	<b>sTeX-Symbols Implementation</b>	<b>86</b>
19.1	Symbol Declarations . . . . .	86
19.2	Notations . . . . .	92
<b>20</b>	<b>sTeX-Terms Implementation</b>	<b>100</b>
20.1	Symbol Invocations . . . . .	100
20.2	Terms . . . . .	102
20.3	Notation Components . . . . .	109
<b>21</b>	<b>sTeX-Structural Features Implementation</b>	<b>111</b>
<b>22</b>	<b>sTeX-Others Implementation</b>	<b>120</b>
<b>23</b>	<b>sTeX-Metatheory Implementation</b>	<b>121</b>
<b>24</b>	<b>Tikzinput Implementation</b>	<b>124</b>
<b>25</b>	<b>document-structure.sty Implementation</b>	<b>126</b>
25.1	The OMDoc Class . . . . .	126
25.2	Class Options . . . . .	126
25.3	Beefing up the document environment . . . . .	127
25.4	Implementation: OMDoc Package . . . . .	127
25.5	Package Options . . . . .	127
25.6	Document Structure . . . . .	128
25.7	Front and Backmatter . . . . .	131
25.8	Ignoring Inputs . . . . .	132
25.9	Structure Sharing . . . . .	132
25.10	Global Variables . . . . .	133
25.11	Colors . . . . .	133
<b>26</b>	<b>MiKoSlides – Implementation</b>	<b>134</b>
26.1	Class and Package Options . . . . .	134
26.2	Notes and Slides . . . . .	136
26.3	Header and Footer Lines . . . . .	138
26.4	Colors and Highlighting . . . . .	140
26.5	Sectioning . . . . .	141
26.6	Excursions . . . . .	143
26.7	Miscellaneous . . . . .	144

**Part I**  
**Manual**

# Chapter 1

## Stuff

### 1.1 Modules

---

`\sTeX` Both print this  $\text{\TeX}$  logo.  
`\stex`

---

#### 1.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

##### Example 1

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

$ab$

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

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

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

### Example 2

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

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

.

Not using an explicit option with a semantic macro yields the first declared notation, unless 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  $\textcolor{teal}{\$}\textcolor{teal}{add}\textcolor{teal}{\$}$  adds two elements, as in  $\textcolor{teal}{\$}\textcolor{teal}{add}\textcolor{teal}{ab}\textcolor{teal}{\$}$ .
```

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

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

### Example 7

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

$\textcolor{teal}{Multiplication}$  (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.  $\int \int \int f \, dx \, dy \, dz$ ?

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

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

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

**mathhub** ( $\langle\text{directory}\rangle$ ) MathHub folder to search for repositories.

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

**image** ( $\langle\text{boolean}\rangle$ ) 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\text{-}prefix\rangle</math>} {<math>\langle\text{message}\rangle</math>}</code>
-----------------------------	---

---

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

---

<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> X <sub>2</sub> e and L <sup>A</sup> T <sub>E</sub> X <sub>3</sub> conditionals for L <sup>A</sup> T <sub>E</sub> X <sub>ML</sub> .
<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>X<sub>ML</sub> 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://home/jazzpirate/work/Software/ext/sTeX/doc/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

.

### 5.1.1 The module-environment

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

---

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

---

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

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

### Test 4

```

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

```

```

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

```

.

## Test 5

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

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

---

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

---

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

---

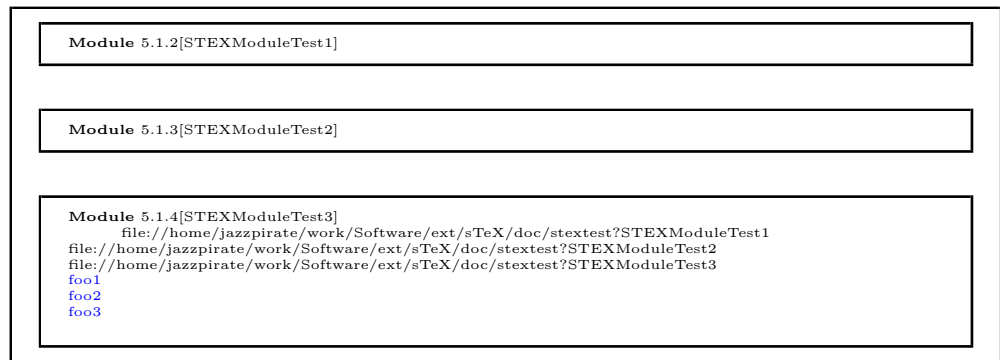
`\stex_invoke_module:n`

---

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

## Test 6

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




---

`\stex_activate_module:n`

---

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

## Chapter 6

# STEX-Module Inheritance

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

### 6.1 Macros and Environments

#### 6.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T<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://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

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

**Module 6.1.2[Importtest]**

Meaning:  $\rightarrow$  `\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

**Module 6.1.3[Importtest2]**

Meaning:  $\rightarrow$  `\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

---

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

---

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

### Test 9

```

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

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

```

**Module 6.1.4[UseTest1]**

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

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

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

### Test 10

```

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

```

Circular dependencies:

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

---

---

`\stex_import_module_uri:nn`

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

---

`\stex_import_require_module:nnnn`

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

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

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

# Chapter 7

## STEX-Symbols

Code related to symbol declarations and notations

### 7.1 Macros and Environments

---

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

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

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

---

---

`\stex_symdecl_do:n`

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

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

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

### Test 11

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

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

---

---

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

---

---

`\stex_get_symbol:n`

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

---

---

`\notation`

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

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

---

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

---

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

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

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

### Test 12

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

Module 7.1.2[NotationTest]

---

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

---

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

### Test 13

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

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

# Chapter 8

## STEX-Terms

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

### 8.1 Macros and Environments

<hr/> <hr/> <code>\STEXsymbol</code>	Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>
<hr/> <hr/> <code>\symref</code>	<code>\symref{&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 } }
 $\bar{foobar} a\{b,c,d,e,f\}g$  and  $\bar{foobar}[foo] a\{b,c\}g$  and  $\bar{foobar} abc$ 

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

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

---

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

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

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

---

`\STEXinvisible`

---

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

---

`\ellipses`

---

TODO

# Chapter 9

## STEX-Structural Features

Code related to structural features

### 9.1 Macros and Environments

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



## Chapter 10

# sTeX-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

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

## 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 $\text{\LaTeX}$ 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

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.

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.

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  $\text{\TeX}$ and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `mikoslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

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

#### 13.2.1 Package Options

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

- |                           |   |
|---------------------------|---|
| <code>slides</code>       | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see   |
| <code>notes</code>        | Section 13.2.2).  |
| <code>sectocframes</code> | • If the option <code>sectocframes</code> is given, then for the <code>omgroups</code> , special frames with the <code>omgroup</code> title (and number) are generated. |

<code>showmeta</code>	<ul style="list-style-type: none"> <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 \RequirePackage{morewrites}
22
23 Package options:
24 \keys_define:nn { stex } {
25   debug      .clist_set:N = \c_stex_debug_clist ,
26   showmods   .bool_set:N  = \c_stex_showmods_bool ,
27   lang       .clist_set:N = \c_stex_languages_clist ,
```

```

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

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

```

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

Patching `expl3`, if outdated:

```

38 <@@=keys>
39 \cs_if_exist:cF { \c__keys_props_root_str .str_set:N }{
40 \cs_new_protected:cpn { \c__keys_props_root_str .str_set:N } #1
41 { \__keys_variable_set:NnnN #1 { str } { } n }
42 \cs_new_protected:cpn { \c__keys_props_root_str .str_set:c } #1
43 { \__keys_variable_set:cnnN {#1} { str } { } n }
44 \cs_new_protected:cpn { \c__keys_props_root_str .str_set_x:N } #1
45 { \__keys_variable_set:NnnN #1 { str } { } x }
46 \cs_new_protected:cpn { \c__keys_props_root_str .str_set_x:c } #1
47 { \__keys_variable_set:cnnN {#1} { str } { } x }
48 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset:N } #1
49 { \__keys_variable_set:NnnN #1 { str } { g } n }
50 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset:c } #1
51 { \__keys_variable_set:cnnN {#1} { str } { g } n }
52 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset_x:N } #1
53 { \__keys_variable_set:NnnN #1 { str } { g } x }
54 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset_x:c } #1
55 { \__keys_variable_set:cnnN {#1} { str } { g } x }
56 }

```

## 14.3 Messages and logging

```

57 <@@=stex_log>
    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.

```

68 \cs_new_protected:Nn \stex_debug:nn {
69   \clist_if_in:NnTF \c_stex_debug_clist { all } {
70     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
71       \\Debug~#1:~#2\\
72     }
73     \msg_none:nn{stex}{debug / #1}
74   }{
75     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
76       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
77         \\Debug~#1:~#2\\
78       }
79       \msg_none:nn{stex}{debug / #1}
80     }
81   }
82 }

```

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

Redirecting messages:

```

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

```

## 14.4 Persistence

```

92 <@=stex_persist>

```

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

```

93 \iow_new:N \c__stex_persist_sms_iow
94 \AddToHook{begindocument}{
95   \bool_if:NTF \c_stex_persist_mode_bool {
96     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
97   } {
98     \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
99   }
100 }
101 \AddToHook{enddocument}{
102   \bool_if:NF \c_stex_persist_mode_bool {
103     \iow_close:N \c__stex_persist_sms_iow
104   }
105 }

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

```

109 }
110 }

```

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

## 14.5 HTML Annotations

```

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

```

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

```

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

```

```

\if@latexml Conditional for LATEXML:
\latexml_if_p: 114 \ifcsname if@latexml\endcsname\else
\latexml_if:TF 115 \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
116 \fi
117
118 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
119   \if@latexml
120     \prg_return_true:
121   \else:
122     \prg_return_false:
123   \fi:
124 }

```

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

```

\l__stex_annotate_arg_tl Used by annotation macros to ensure that the HTML output to annotate is not empty.
\c__stex_annotate_emptyarg_tl
125 \tl_new:N \l__stex_annotate_arg_tl
126 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
127   \scalatex_if:TF {
128     \scalatex_direct_HTML:n { \c_ampsand_str lrm; }
129   }{-}
130 }

```

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

```

\__stex_annotate_checkempty:n
131 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
132   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
133   \tl_if_empty:NT \l__stex_annotate_arg_tl {
134     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
135   }
136 }

```

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

```

\l_stex_html_do_output_bool Whether to (locally) produce HTML output
\stex_if_do_html:
137 \bool_new:N \l_stex_html_do_output_bool
138 \bool_set_true:N \l_stex_html_do_output_bool
139 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
140   \bool_if:nTF \l_stex_html_do_output_bool
141     \prg_return_true: \prg_return_false:
142 }

```

(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

```

143 \cs_new_protected:Nn \stex_suppress_html:n {
144   \exp_args:Nne \use:nn {
145     \bool_set_false:N \l_stex_html_do_output_bool
146     #1
147   }{
148     \stex_if_do_html:T {
149       \bool_set_true:N \l_stex_html_do_output_bool
150     }
151   }
152 }
```

(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`

```

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

```

186     property="stex:#1" ~
187     resource="#2"
188   }
189 }{
190   \scalatex_annotate_HTML_end:
191 }
192 }{
193   \latexml_if:TF {
194     \cs_new_protected:Nn \stex_annotate:nnn {
195       \__stex_annotate_checkempty:n { #3 }
196       \mode_if_math:TF {
197         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
198           \tl_use:N \l__stex_annotate_arg_tl
199         }
200       }{
201         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
202           \tl_use:N \l__stex_annotate_arg_tl
203         }
204       }
205     }
206     \cs_new_protected:Nn \stex_annotate_invisible:n {
207       \__stex_annotate_checkempty:n { #1 }
208       \mode_if_math:TF {
209         \cs:w latexml@invisible@math\cs_end:{
210           \tl_use:N \l__stex_annotate_arg_tl
211         }
212       } {
213         \cs:w latexml@invisible@text\cs_end:{
214           \tl_use:N \l__stex_annotate_arg_tl
215         }
216       }
217     }
218     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
219       \__stex_annotate_checkempty:n { #3 }
220       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
221         \tl_use:N \l__stex_annotate_arg_tl
222       }
223     }
224     \NewDocumentEnvironment{stex_annotate_env} { m m } {
225       \par\begin{latexml@annotateenv}{#1}{#2}
226     }{
227       \end{latexml@annotateenv}
228     }
229   }{
230     \cs_new_protected:Nn \stex_annotate:nnn {#3}
231     \cs_new_protected:Nn \stex_annotate_invisible:n {}
232     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
233     \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{\par}
234   }
235 }

```

(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

236 `<@@=stex_language>`

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

237 \prop_const_from_keyval:Nn \c_stex_languages_prop {
238   en = english ,
239   de = ngerman ,
240   ar = arabic ,
241   bg = bulgarian ,
242   ru = russian ,
243   fi = finnish ,
244   ro = romanian ,
245   tr = turkish ,
246   fr = french
247 }
248
249 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
250   english   = en ,
251   ngerman   = de ,
252   arabic    = ar ,
253   bulgarian = bg ,
254   russian   = ru ,
255   finnish   = fi ,
256   romanian  = ro ,
257   turkish   = tr ,
258   french    = fr
259 }
260 % todo: chinese simplified (zhs)
261 %       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:

```

262 \clist_if_empty:NF \c_stex_languages_clist {
263   \clist_clear:N \l_tmpa_clist
264   \clist_map_inline:Nn \c_stex_languages_clist {
265     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
266       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
267     } {
268       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
269     }
270   }
271   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
272   \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
273 }

```

## 14.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

```

277     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
278   }
279 }

```

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

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

```

280 \cs_new_protected:Nn \stex_reactivate_macro:N {
281   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
282 }

```

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

```

283 \</package>

```



## Chapter 15

# STEX -MathHub Implementation

```
284 <*package>
285
286 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
287
288 <@@=stex_path>
289
290 Warnings and error messages
291 \msg_new:nnn{stex}{error/norepository}{
292   No~archive~#1~found~in~#2
293 }
294 \msg_new:nnn{stex}{error/notinarchive}{
295   Not~currently~in~an~archive,~but~\detokenize{#1}~
296   needs~one!
297 }
298 \msg_new:nnn{stex}{error/nofile}{
299   \detokenize{#1}~could~not~find~file~#2
300 }
```

### 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
299 \cs_new_protected:Nn \stex_path_from_string:Nn {
300   \str_set:Nx \l_tmpa_str { #2 }
301   \str_if_empty:NTF \l_tmpa_str {
302     \seq_clear:N #1
303   }{
304     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
305     \sys_if_platform_windows:T{
306       \seq_clear:N \l_tmpa_tl
307       \seq_map_inline:Nn #1 {
308         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
309         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl

```

```

310     }
311     \seq_set_eq:NN #1 \l_tmpa_tl
312   }
313   \stex_path_canonicalize:N #1
314 }
315 }
316 \cs_generate_variant:Nn \stex_path_from_string:Nn
317 { 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
318 \cs_new_protected:Nn \stex_path_to_string:NN {
319   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
320 }
321
322 \cs_new:Nn \stex_path_to_string:N {
323   \seq_use:Nn #1 /
324 }

```

(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
325 \str_const:Nn \c__stex_path_dot_str {.}
326 \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.

```

327 \cs_new_protected:Nn \stex_path_canonicalize:N {
328   \seq_if_empty:NF #1 {
329     \seq_clear:N \l_tmpa_seq
330     \seq_get_left:NN #1 \l_tmpa_tl
331     \str_if_empty:NT \l_tmpa_tl {
332       \seq_put_right:Nn \l_tmpa_seq {}
333     }
334     \seq_map_inline:Nn #1 {
335       \str_set:Nn \l_tmpa_tl { ##1 }
336       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
337         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
338           \seq_if_empty:NTF \l_tmpa_seq {
339             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
340               \c__stex_path_up_str
341             }
342           }{
343             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
344             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
345               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
346                 \c__stex_path_up_str
347             }
348           }{
349             \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
350           }

```

```

351     }
352   }{
353     \str_if_empty:NF \l_tmpa_tl {
354       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
355     }
356   }
357 }
358 }
359 \seq_gset_eq:NN #1 \l_tmpa_seq
360 }
361 }

```

(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`

```

362 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
363   \seq_if_empty:NTF #1 {
364     \prg_return_false:
365   }{
366     \seq_get_left:NN #1 \l_tmpa_tl
367     \str_if_empty:NTF \l_tmpa_tl {
368       \prg_return_true:
369     }{
370       \prg_return_false:
371     }
372   }
373 }

```

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

## 15.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

374 \str_new:N\l_stex_kpsewhich_return_str
375 \cs_new_protected:Nn \stex_kpsewhich:n {
376   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
377   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
378   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
379 }

```

(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`

```

380 \sys_if_platform_windows:TF{
381   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
382 }{
383   \stex_kpsewhich:n{-var-value~PWD}
384 }
385
386 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
387 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
388 \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

389 `<@@=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  $\text{\TeX}$ -purposes.

`\g__stex_files_stack` keeps track of file changes

390 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

392 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

393 `\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`.

```

394 \seq_gclear_new:N\g_stex_currentfile_seq
395 \AddToHook{file/before}{
396   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
397   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
398     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
399   }{
400     \stex_path_from_string:Nn\g_stex_currentfile_seq{
401       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
402     }
403   }
404   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
405   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
406 }
407 \AddToHook{file/after}{
408   \seq_if_empty:NF\g__stex_files_stack{
409     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
410   }
411   \seq_if_empty:NTF\g__stex_files_stack{
412     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
413   }{
414     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
415     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
416   }
417 }

```

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

## 15.4 MathHub Repositories

```

418 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
419 \str_if_empty:NTF\mathhub{
420   \stex_kpsewhich:n{-var-value~MATHHUB}
421   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
422
423   \str_if_empty:NTF\c_stex_mathhub_str{
424     \msg_warning:nn{stex}{warning/nomathhub}
425   }{
426     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
427     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
428   }
429 }{
430   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
431   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
432     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
433       \c_stex_pwd_str/\mathhub
434     }
435   }
436   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
437   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
438 }

```

(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
439 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
440   \str_set:Nx \l_tmpa_str { #1 }
441   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
442     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
443     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
444     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
445     \__stex_mathhub_find_manifest:N \l_tmpa_seq
446     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
447       \msg_error:nnnn{stex}{error/norepository}{#1}{
448         \stex_path_to_string:N \c_stex_mathhub_str
449       }
450     } {
451       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
452     }
453   }
454 }

```

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

```

\l__stex_mathhub_manifest_file_seq
455 \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`:

```

456 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
457   \seq_set_eq:NN \l_tmpa_seq #1
458   \bool_set_true:N \l_tmpa_bool
459   \bool_while_do:Nn \l_tmpa_bool {
460     \seq_if_empty:NTF \l_tmpa_seq {
461       \bool_set_false:N \l_tmpa_bool
462     }{
463       \file_if_exist:nTF{
464         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
465       }{
466         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
467         \bool_set_false:N \l_tmpa_bool
468       }{
469         \file_if_exist:nTF{
470           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
471         }{
472           \seq_put_right:Nn \l_tmpa_seq{META-INF}
473           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
474           \bool_set_false:N \l_tmpa_bool
475         }{
476           \file_if_exist:nTF{
477             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
478           }{
479             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
480             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
481             \bool_set_false:N \l_tmpa_bool
482           }{
483             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
484           }
485         }
486       }
487     }
488   }
489   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
490 }

```

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

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

491 \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:

```

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

```

```

500 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
501 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
502 }
503 \exp_args:No \str_case:nnTF \l_tmpa_tl {
504 {id} {
505 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
506 { id } \l_tmpb_tl
507 }
508 {narration-base} {
509 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
510 { narr } \l_tmpb_tl
511 }
512 {url-base} {
513 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
514 { docurl } \l_tmpb_tl
515 }
516 {source-base} {
517 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
518 { ns } \l_tmpb_tl
519 }
520 {ns} {
521 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
522 { ns } \l_tmpb_tl
523 }
524 {dependencies} {
525 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
526 { deps } \l_tmpb_tl
527 }
528 }{}{}
529 }{}
530 }
531 \ior_close:N \c__stex_mathhub_manifest_ior
532 }

```

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

`\stex_set_current_repository:n`

```

533 \cs_new_protected:Nn \stex_set_current_repository:n {
534 \stex_require_repository:n { #1 }
535 \prop_set_eq:Nc \l_stex_current_repository_prop {
536 c_stex_mathhub_#1_manifest_prop
537 }
538 }

```

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

`\stex_require_repository:n`

```

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

```

```

547     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
548     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
549   }
550 }
551 }
552 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

553 \prop_new:N \l_stex_current_repository_prop
554
555 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
556 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
557   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
558 } {
559   \__stex_mathhub_parse_manifest:n { main }
560   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
561   \l_tmpa_str
562   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
563   \c_stex_mathhub_main_manifest_prop
564   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
565   \stex_debug:nn{mathhub}{Current~repository:~
566     \prop_item:Nn \l_stex_current_repository_prop {id}
567   }
568 }

```

(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.

```

569 \cs_new_protected:Nn \stex_in_repository:nn {
570   \str_set:Nx \l_tmpa_str { #1 }
571   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
572   \str_if_empty:NTF \l_tmpa_str {
573     \exp_args:Ne \l_tmpa_cs{
574       \prop_item:Nn \l_stex_current_repository_prop { id }
575     }
576   }{
577     \stex_require_repository:n \l_tmpa_str
578     \str_set:Nx \l_tmpa_str { #1 }
579     \exp_args:Nne \use:nn {
580       \stex_set_current_repository:n \l_tmpa_str
581       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
582     }{
583       \stex_set_current_repository:n {
584         \prop_item:Nn \l_stex_current_repository_prop { id }
585       }
586     }
587   }
588 }

```

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



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

```

589 \newif \ifinputref \inputreffalse
590
591 \cs_new_protected:Nn \inputref:nn {
592   \stex_in_repository:nn {#1} {
593     \ifinputref
594       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
595     \else
596       \inputreftrue
597       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
598     \inputreffalse
599   \fi
600 }
601 }
602 \NewDocumentCommand \inputref { 0{} m}{
603   \inputref:nn{ #1 }{ #2 }
604 }

```

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

**\mhp**

```

605 \def \mhp #1 #2 {
606   \exp_args:Ne \str_if_eq:nnTF{#1}{#2}{
607     \c_stex_mathhub_str /
608     \prop_item:Nn \l_stex_current_repository_prop { id }
609     / source / #2
610   }{
611     \c_stex_mathhub_str / #1 / source / #2
612   }
613 }

```

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

**\libinput**

```

614 \cs_new_protected:Npn \libinput #1 {
615   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
616     \msg_error:nnn{stex}{error/notinarchive}\libinput
617   }
618   \bool_set_false:N \l_tmpa_bool
619   \tl_clear:N \l_tmpa_tl
620   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
621   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
622   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
623   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
624     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
625     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
626       / meta-inf / lib / #1.tex}{
627       \bool_set_true:N \l_tmpa_bool
628       \tl_put_right:Nx \l_tmpa_tl {
629         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
630           / meta-inf / lib / #1.tex}
631       }
632     }{}
633 }

```

```

634 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
635 / \l_tmpa_str / lib / #1.tex
636 }{
637   \bool_set_true:N \l_tmpa_bool
638   \tl_put_right:Nx \l_tmpa_tl {
639     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
640       / \l_tmpa_str / lib / #1.tex}
641   }
642 }{}
643 \bool_if:NF \l_tmpa_bool {
644   \msg_error:nnnn{stex}{error/nofile}\libinput{#1.tex}
645 }
646 \l_tmpa_tl
647 }

```

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

```

648 </package>

```

## Chapter 16

# STEX -References Implementation

```
649 <*package>
650
651 %%%%%%%%%% references.dtx %%%%%%%%%%
652
653 %\RequirePackage{hyperref}
654 %\RequirePackage{cleveref}
655 <@@=stex_refs>
656
657 Warnings and error messages
658
659 \iow_new:N \c__stex_refs_refs_iow
660 \AddToHook{begindocument}{
661   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
662 }
663 \AddToHook{enddocument}{
664   \iow_close:N \c__stex_refs_refs_iow
665 }
666
667 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
668
669 \NewDocumentCommand \STEXreftitle { m } {
670   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
671 }
```

### 16.1 Document URIs and URLs

```
670 \seq_new:N \g__stex_refs_all_refs_seq
671
672 \str_new:N \l_stex_current_docns_str
673
674 \cs_new_protected:Nn \stex_get_document_uri: {
675   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
676   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
677   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
678   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
```

```

679 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
680
681 \str_clear:N \l_tmpa_str
682 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
683   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
684 }
685
686 \str_if_empty:NTF \l_tmpa_str {
687   \str_set:Nx \l_stex_current_docns_str {
688     file:/\stex_path_to_string:N \l_tmpa_seq
689   }
690 }{
691   \bool_set_true:N \l_tmpa_bool
692   \bool_while_do:Nn \l_tmpa_bool {
693     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
694     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
695       {source} { \bool_set_false:N \l_tmpa_bool }
696     }{}{
697       \seq_if_empty:NT \l_tmpa_seq {
698         \bool_set_false:N \l_tmpa_bool
699       }
700     }
701   }
702
703   \seq_if_empty:NTF \l_tmpa_seq {
704     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
705   }{
706     \str_set:Nx \l_stex_current_docns_str {
707       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
708     }
709   }
710 }
711 }
712
713 \str_new:N \l_stex_current_docurl_str
714 \cs_new_protected:Nn \stex_get_document_url: {
715   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
716   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
717   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
718   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
719   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
720
721   \str_clear:N \l_tmpa_str
722   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
723     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
724       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
725     }
726   }
727
728   \str_if_empty:NTF \l_tmpa_str {
729     \str_set:Nx \l_stex_current_docurl_str {
730       file:/\stex_path_to_string:N \l_tmpa_seq
731     }
732   }{
733     \bool_set_true:N \l_tmpa_bool

```

```

733 \bool_while_do:Nn \l_tmpa_bool {
734   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
735   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
736     {source} { \bool_set_false:N \l_tmpa_bool }
737   }{}{
738     \seq_if_empty:NT \l_tmpa_seq {
739       \bool_set_false:N \l_tmpa_bool
740     }
741   }
742 }
743
744 \seq_if_empty:NTF \l_tmpa_seq {
745   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
746 }{
747   \str_set:Nx \l_stex_current_docurl_str {
748     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
749   }
750 }
751 }
752 }

```

## 16.2 Setting Reference Targets

```

753 \str_const:Nn \c__stex_refs_url_str{URL}
754 \str_const:Nn \c__stex_refs_ref_str{REF}
755 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
756   \stex_get_document_uri:
757   \str_set:Nx \l_tmpa_str { #1 }
758   \str_if_empty:NT \l_tmpa_str {
759     \int_zero:N \l_tmpa_int
760     \bool_set_true:N \l_tmpa_bool
761     \bool_while_do:Nn \l_tmpa_bool {
762       \cs_if_exist:cTF {
763         sref_\l_stex_current_docns_str\c_hash_str REF\int_use:N \l_tmpa_int _type
764       }{
765         \int_incr:N \l_tmpa_int
766       }{}
767       \str_set:Nx \l_tmpa_str { REF\int_use:N \l_tmpa_int }
768       \bool_set_false:N \l_tmpa_bool
769     }
770   }
771 }
772 \str_set:Nx \l_tmpa_str {
773   \l_stex_current_docns_str\c_hash_str\l_tmpa_str
774 }
775 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
776 \stex_if_smsmode:TF {
777   \stex_get_document_url:
778   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
779   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
780 }{
781   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~::~\expandafter{\@currentlabel~in~\exp_a
782   \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
783   \str_gset:cn {sref_\l_tmpa_str _type}\c__stex_refs_ref_str

```

```

784 }
785 }
786 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
787   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
788 }

```

## 16.3 Using References

```

789 \keys_define:nn { stex / sref } {
790   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
791   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
792   pre           .tl_set:N = \l__stex_refs_pre_tl ,
793   post          .tl_set:N = \l__stex_refs_post_tl ,
794   indoc         .str_set_x:N = \l__stex_refs_repo_str ,
795 }
796
797 \cs_new_protected:Nn \__stex_refs_args:n {
798   \tl_clear:N \l__stex_refs_linktext_tl
799   \tl_clear:N \l__stex_refs_fallback_tl
800   \tl_clear:N \l__stex_refs_pre_tl
801   \tl_clear:N \l__stex_refs_post_tl
802   \str_clear:N \l__stex_refs_repo_str
803   \keys_set:nn { stex / sref } { #1 }
804 }
805
806 \</package>

```

## Chapter 17

# STEX -Modules Implementation

```
807 <*package>
808
809 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
810
811 <@@=stex_modules>
      Warnings and error messages
812 \msg_new:nnn{stex}{error/unknownmodule}{
813   No~module~#1~found
814 }
815 \msg_new:nnn{stex}{error/syntax}{
816   Syntax~error:~#1
817 }
818 \msg_new:nnn{stex}{error/siglanguage}{
819   Module~#1~declares~signature~#2,~but~does~not~
820   declare~its~language
821 }
```

`\l_stex_current_module_prop` The current module:

```
822 \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

```
823 \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`

```
824 \seq_new:N \g_stex_modules_in_file_seq
825 \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
826 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
827   \prop_if_empty:NTF \l_stex_current_module_prop
828   \prg_return_false: \prg_return_true:
829 }

```

(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
830 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
831   \prop_if_exist:cTF { c_stex_module_#1_prop }
832   \prg_return_true: \prg_return_false:
833 }

```

(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
834 \cs_new_protected:Nn \stex_add_to_current_module:n {
835   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
836   \tl_put_right:Nn \l_tmpa_tl { #1 }
837   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
838 }
839 \cs_new_protected:Npn \STEXexport #1 {
840   #1
841   \stex_add_to_current_module:n { #1 }
842   \stex_smsmode_set_codes:
843 }
844 \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
845 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
846   \str_set:Nx \l_tmpa_str { #1 }
847   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
848   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
849   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
850 }

```

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

```

\stex_add_import_to_current_module:n
851 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
852   \str_set:Nx \l_tmpa_str { #1 }
853   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
854   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
855   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
856 }

```

(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).

```

857 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
858   \str_set:Nx \l_tmpa_str { #1 }
859   \seq_set_eq:NN \l_tmpa_seq #2
860   % split off file extension
861   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
862   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
863   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
864   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
865
866   \bool_set_true:N \l_tmpa_bool
867   \bool_while_do:Nn \l_tmpa_bool {
868     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
869     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
870       {source} { \bool_set_false:N \l_tmpa_bool }
871     }{}{
872       \seq_if_empty:NT \l_tmpa_seq {
873         \bool_set_false:N \l_tmpa_bool
874       }
875     }
876   }
877
878   \seq_if_empty:NTF \l_tmpa_seq {
879     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
880   }{
881     \str_set:Nx \l_stex_modules_ns_str {
882       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
883     }
884   }
885 }

```

(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`

```

886 \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.

```

887 \cs_new_protected:Nn \stex_modules_current_namespace: {
888   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
889     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
890   }{
891     % split off file extension
892     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
893     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
894     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
895     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
896     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
897     \str_set:Nx \l_stex_modules_ns_str {
898       file:/\stex_path_to_string:N \l_tmpa_seq

```

```

899     }
900   }
901 }

```

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

## 17.1 The module environment

module arguments:

```

902 \keys_define:nn { stex / module } {
903   title      .str_set_x:N = \l_stex_module_title_str ,
904   ns         .str_set_x:N = \l_stex_module_ns_str ,
905   lang       .str_set_x:N = \l_stex_module_lang_str ,
906   sig        .str_set_x:N = \l_stex_module_sig_str ,
907   creators   .str_set_x:N = \l_stex_module_creators_str ,
908   contributors .str_set_x:N = \l_stex_module_contributors_str ,
909   meta       .str_set_x:N = \l_stex_module_meta_str
910 }
911
912 \cs_new_protected:Nn \__stex_modules_args:n {
913   \str_clear:N \l_stex_module_title_str
914   \str_clear:N \l_stex_module_ns_str
915   \str_clear:N \l_stex_module_lang_str
916   \str_clear:N \l_stex_module_sig_str
917   \str_clear:N \l_stex_module_creators_str
918   \str_clear:N \l_stex_module_contributors_str
919   \str_clear:N \l_stex_module_meta_str
920   \keys_set:nn { stex / module } { #1 }
921 }
922
923 % module parameters here? In the body?
924

```

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

```

925 \cs_new_protected:Nn \stex_module_setup:nn {
926   \str_set:Nx \l_stex_module_name_str { #2 }
927   \__stex_modules_args:n { #1 }
928
929   First, we set up the name and namespace of the module.
930   Are we in a nested module?
931
932   \stex_if_in_module:TF {
933     % Nested module
934     \prop_get:NnN \l_stex_current_module_prop
935       { ns } \l_stex_module_ns_str
936     \str_set:Nx \l_stex_module_name_str {
937       \prop_item:Nn \l_stex_current_module_prop
938       { name } / \l_stex_module_name_str
939     }
940   }{
941     % not nested:
942     \str_if_empty:NT \l_stex_module_ns_str {
943       \stex_modules_current_namespace:
944       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
945     }
946   }
947 }

```

```

941 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
942 / {\l_stex_module_ns_str}
943 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
944 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
945   \str_set:Nx \l_stex_module_ns_str {
946     \stex_path_to_string:N \l_tmpa_seq
947   }
948 }
949 }
950 }

```

Next, we determine the language of the module:

```

951 \str_if_empty:NT \l_stex_module_lang_str {
952   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
953   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
954   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
955   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
956   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
957     \stex_debug:nn{modules} {Language-\l_stex_module_lang_str~
958       inferred~from~file~name}
959     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
960   }
961 }
962
963 \str_if_empty:NF \l_stex_module_lang_str {
964   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
965   \l_tmpa_str {
966     \ltx@ifpackageloaded{babel}{
967       \exp_args:Nx \selectlanguage { \l_tmpa_str }
968     }{}
969   } {
970     \msg_error:nnn{stex}{error/unknownlanguage}{\l_tmpa_str}
971   }
972 }

```

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

```

973 \str_if_empty:NTF \l_stex_module_sig_str {
974   \str_clear:N \l_tmpa_str
975   \seq_clear:N \l_tmpa_seq
976   \tl_clear:N \l_tmpa_tl
977   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
978     name      = \l_stex_module_name_str ,
979     ns        = \l_stex_module_ns_str ,
980     imports   = \exp_not:o { \l_tmpa_seq } ,
981     constants = \exp_not:o { \l_tmpa_seq } ,
982     content   = \exp_not:o { \l_tmpa_tl } ,
983     file      = \exp_not:o { \g_stex_currentfile_seq } ,
984     lang      = \l_stex_module_lang_str ,
985     sig       = \l_stex_module_sig_str ,
986     meta      = \l_stex_module_meta_str
987   }
988 }{
989   \str_if_empty:NT \l_stex_module_lang_str {

```

```

990     \msg_error:nnnn{stex}{error/siglanguage}{
991       \l_stex_module_ns_str?\l_stex_module_name_str
992     }{\l_stex_module_sig_str}
993   }
994
995   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
996   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
997   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
998   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
999   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1000   \str_set:Nx \l_tmpa_str {
1001     \stex_path_to_string:N \l_tmpa_seq /
1002     \l_tmpa_str . \l_stex_module_sig_str .tex
1003   }
1004   \IfFileExists \l_tmpa_str {
1005     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1006       \seq_clear:N \l_stex_all_modules_seq
1007       \prop_clear:N \l_stex_current_module_prop
1008       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1009       \input { \l_tmpa_str }
1010     }
1011   }{
1012     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1013   }
1014   \stex_activate_module:n {
1015     \l_stex_module_ns_str ? \l_stex_module_name_str
1016   }
1017   \prop_set_eq:Nc \l_stex_current_module_prop {
1018     c_stex_module_
1019     \l_stex_module_ns_str ?
1020     \l_stex_module_name_str
1021     _prop
1022   }
1023 }

```

We load the metatheory:

```

1024 \str_if_empty:NT \l_stex_module_meta_str {
1025   \str_set:Nx \l_stex_module_meta_str {
1026     \c_stex_metatheory_ns_str ? Metatheory
1027   }
1028 }
1029 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1030   \exp_args:Nx \stex_add_to_current_module:n {
1031     \stex_activate_module:n {\l_stex_module_meta_str}
1032   }
1033   \stex_activate_module:n {\l_stex_module_meta_str}
1034 }
1035 }

```

*(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}`

```

1036 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1037   \stex_reactivate_macro:N \STEXexport
1038   \stex_reactivate_macro:N \importmodule
1039   \stex_reactivate_macro:N \symdecl
1040   \stex_reactivate_macro:N \notation
1041   \stex_reactivate_macro:N \symdef
1042   \stex_module_setup:nn{#1}{#2}
1043
1044   \stex_debug:nn{modules}{
1045     New~module:\\
1046     Namespace:~\l_stex_module_ns_str\\
1047     Name:~\l_stex_module_name_str\\
1048     Language:~\l_stex_module_lang_str\\
1049     Signature:~\l_stex_module_sig_str\\
1050     Metatheory:~\l_stex_module_meta_str\\
1051     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1052   }
1053
1054   \seq_put_right:Nx \l_stex_all_modules_seq {
1055     \l_stex_module_ns_str ? \l_stex_module_name_str
1056   }
1057
1058   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1059     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1060
1061   \stex_if_smsmode:TF {
1062     \stex_smsmode_set_codes:
1063   } {
1064     \begin{stex_annotate_env} {theory} {
1065       \l_stex_module_ns_str ? \l_stex_module_name_str
1066     }
1067
1068     \stex_annotate_invisible:nnn{header}{} {
1069       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1070       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1071       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1072         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1073       }
1074     }
1075   }
1076   % TODO: Inherit metatheory for nested modules?
1077 }
1078 \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}

```

1079 \cs_new_protected:Nn \__stex_modules_end_module: {
1080   \str_set:Nx \l_tmpa_str {
1081     c_stex_module_
1082     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1083     \prop_item:Nn \l_stex_current_module_prop { name }
1084     _prop
1085   }

```

```

1086 %^^A \prop_new:c { \l_tmpa_str }
1087 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1088 \stex_debug:nn{modules}{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
1089 }

```

(End definition for `\_stex_modules_end_module:.`)

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

```

1090 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1091 \NewDocumentEnvironment { @module } { 0{} m } {
1092   \par
1093   \_stex_modules_begin_module:nn{#1}{#2}
1094 } {
1095   \_stex_modules_end_module:
1096   \stex_if_smsmode:TF {
1097     \exp_args:Nx \stex_add_to_sms:n {
1098       \prop_gset_from_keyval:cn {
1099         c_stex_module_
1100         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1101         \prop_item:Nn \l_stex_current_module_prop { name }
1102         _prop
1103       } {
1104         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1105         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1106         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1107         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1108         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1109         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1110         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1111         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1112         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1113       }
1114     }
1115   }{
1116     \end{stex_annotate_env}
1117   }
1118 }

```

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

```

1119 \cs_if_exist:NTF \thesection {
1120   \newcounter{module}[section]
1121 }{
1122   \newcounter{module}
1123 }
1124
1125 \bool_if:NT \c_stex_showmods_bool {
1126   \latexml_if:F { \RequirePackage{mdframed} }
1127 }
1128
1129 \cs_new_protected:Nn \stex_modules_heading: {
1130   \stepcounter{module}
1131   \par
1132   \bool_if:NT \c_stex_showmods_bool {
1133     \noindent{\textbf{Module}} ~

```

```

1134     \cs_if_exist:NT \thesection {\thesection.}
1135     \themodule ~ [\l_stex_module_name_str]
1136   }
1137   \str_if_empty:NTF \l_stex_module_title_str {
1138   }{
1139     \quad(\l_stex_module_title_str)\hfill
1140   }\par
1141 }
1142 \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1143 % TODO
1144 \stex_ref_new_doc_target:n \l_stex_module_name_str
1145 }

```

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

Finally:

```

1146 \NewDocumentEnvironment { module } { 0{} m } {
1147   \bool_if:NT \c_stex_showmods_bool {
1148     \begin{mdframed}
1149   }
1150   \begin{@module} [#1] {#2}
1151   \stex_modules_heading:
1152 }{
1153   \end{@module}
1154   \bool_if:NT \c_stex_showmods_bool {
1155     \end{mdframed}
1156   }
1157 }

```

## 17.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1158 \NewDocumentCommand \STEXModule { m } {
1159   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1160   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1161   \tl_set:Nn \l_tmpa_tl {
1162     \msg_error:nnn{stex}{error/unknownmodule}{#1}
1163   }
1164   \seq_map_inline:Nn \l_stex_all_modules_seq {
1165     \str_set:Nn \l_tmpb_str { ##1 }
1166     \str_if_eq:eeT { \l_tmpa_str } {
1167       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1168     } {
1169       \seq_map_break:n {
1170         \tl_set:Nn \l_tmpa_tl {
1171           \stex_invoke_module:n { ##1 }
1172         }
1173       }
1174     }
1175   }
1176   \l_tmpa_tl
1177 }
1178
1179 \cs_new_protected:Nn \stex_invoke_module:n {

```

```

1180 \stex_debug:nn{modules}{Invoking~module~#1}
1181 \peek_charcode_remove:NTF ! {
1182   \__stex_modules_invoke_uri:nN { #1 }
1183 } {
1184   \peek_charcode_remove:NTF ? {
1185     \__stex_modules_invoke_symbol:nn { #1 }
1186   } {
1187     \msg_error:nnn{stex}{error/syntax}{
1188       ?~or~!~expected~after~
1189       \c_backslash_str STEXModule{#1}
1190     }
1191   }
1192 }
1193 }
1194
1195 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1196   \str_set:Nn #2 { #1 }
1197 }
1198
1199 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1200   \stex_invoke_symbol:n{#1?#2}
1201 }

```

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

`\stex_activate_module:n`

```

1202 \cs_new_protected:Nn \stex_activate_module:n {
1203   \stex_debug:nn{modules}{Activating~module~#1}
1204   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1205     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1206     \prop_item:cn { c_stex_module_#1_prop } { content }
1207   }
1208 }

```

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

```

1209 </package>

```



## Chapter 18

# STEX -Module Inheritance Implementation

```
1210 ⟨*package⟩
1211
1212 %%%%%%%%% inheritance.dtx %%%%%%%%%
1213
```

### 18.1 SMS Mode

```
1214 ⟨@@=stex_smsmode⟩

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1215 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1216 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1217 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1218
1219 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1220   \makeatletter
1221   \makeatother
1222   \ExplSyntaxOn
1223   \ExplSyntaxOff
1224 }
1225
1226 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1227   \symdef
1228   \importmodule
1229   \notation
1230   \symdecl
1231   \STEXexport
1232 }
1233
1234 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1235   \tl_to_str:n {
1236     module,
1237     @module
```

```

1238 }
1239 }

```

(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

```

```

1240 \bool_new:N \g__stex_smsmode_bool
1241 \bool_set_false:N \g__stex_smsmode_bool
1242 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1243   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1244 }

```

(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

```

```

1245 \bool_new:N \g__stex_smsmode_catcode_bool
1246 \bool_set_false:N \g__stex_smsmode_catcode_bool
1247 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1248   \bool_if:NTF \g__stex_smsmode_catcode_bool
1249   \prg_return_true: \prg_return_false:
1250 }

```

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

```

\stex_smsmode_set_codes:

```

```

1251 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1252   \stex_if_smsmode:T {
1253     \__stex_smsmode_if_catcodes:F {
1254       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1255       \exp_after:wN \char_gset_active_eq:NN
1256       \c_backslash_str \__stex_smsmode_cs:
1257       \tex_global:D \char_set_catcode_active:N \
1258       \tex_global:D \char_set_catcode_other:N $
1259       \tex_global:D \char_set_catcode_other:N ^
1260       \tex_global:D \char_set_catcode_other:N _
1261       \tex_global:D \char_set_catcode_other:N &
1262       \tex_global:D \char_set_catcode_other:N ##
1263     }
1264   }
1265 } \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.

```

1266 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1267   \__stex_smsmode_if_catcodes:T {
1268     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1269     \exp_after:wN \tex_global:D \exp_after:wN
1270     \char_set_catcode_escape:N \c_backslash_str
1271     \tex_global:D \char_set_catcode_math_toggle:N $
1272     \tex_global:D \char_set_catcode_math_superscript:N ^
1273     \tex_global:D \char_set_catcode_math_subscript:N _
1274     \tex_global:D \char_set_catcode_alignment:N &
1275     \tex_global:D \char_set_catcode_parameter:N ##
1276   }
1277 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1278 \cs_new_protected:Nn \stex_in_smsmode:nn {
1279   \vbox_set:Nn \l_tmpa_box {
1280     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1281     \bool_gset_true:N \g__stex_smsmode_bool
1282     \stex_smsmode_set_codes:
1283     #2
1284     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1285     \stex_if_smsmode:F {
1286       \__stex_smsmode_unset_codes:
1287     }
1288   }
1289   \box_clear:N \l_tmpa_box
1290 }

```

(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:

```

1291 \cs_new_protected:Nn \_stex_smsmode_cs: {
1292   \str_clear:N \l_tmpa_str
1293   \peek_analysis_map_inline:n {
1294     % #1: token (one expansion)
1295     % #2: charcode
1296     % #3 catcode
1297     \token_if_eq_charcode:NNTF ##3 B {
1298       % token is a letter
1299       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1300     } {
1301       \str_if_empty:NTF \l_tmpa_str {
1302         % we don't allow (or need) single non-letter CSs
1303         % for now
1304         \peek_analysis_map_break:
1305       }{
1306         \str_if_eq:onTF \l_tmpa_str { begin } {
1307           \peek_analysis_map_break:n {
1308             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1309           }
1310         } {
1311           \str_if_eq:onTF \l_tmpa_str { end } {
1312             \peek_analysis_map_break:n {
1313               \exp_after:wN \_stex_smsmode_checkend:n ##1
1314             }
1315           } {
1316             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1317             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1318               \g_stex_smsmode_allowedmacros_tl
1319               { \use:c{\l_tmpa_str} } {
1320               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1321               \peek_analysis_map_break:n {
1322                 \exp_after:wN \l_tmpa_tl ##1
1323               }

```

```

1324     } {
1325         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1326         \g_stex_smsmode_allowedmacros_escape_tl
1327         { \use:c{\l_tmpa_str} } {
1328             \__stex_smsmode_unset_codes:
1329             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1330             % TODO \__stex_smsmode_rescan_cs:
1331             % \exp_after:wN \exp_after:wN \exp_after:wN
1332             % \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1333             % \peek_analysis_map_break:n {
1334             % \__stex_smsmode_unset_codes:
1335             % \__stex_smsmode_rescan_cs:
1336             % }
1337             % } {
1338             % \peek_analysis_map_break:n {
1339             % \exp_after:wN \l_tmpa_tl ##1
1340             % }
1341             % }
1342             % } {
1343             % \peek_analysis_map_break:n { ##1 }
1344             % }
1345             % }
1346             % }
1347             % }
1348             % }
1349             % }
1350             % }
1351             % }

```

(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:

```

1352 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1353     \str_clear:N \l_tmpb_str
1354     \peek_analysis_map_inline:n {
1355         \token_if_eq_charcode:NNTF ##3 B {
1356             % token is a letter
1357             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1358         } {
1359             \peek_analysis_map_break:n {
1360                 \exp_after:wN \use:c \exp_after:wN {
1361                     \exp_after:wN \l_tmpa_str\exp_after:wN
1362                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1363             }
1364         }
1365     }
1366 }

```

(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.

```

1367 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1368     \str_set:Nn \l_tmpa_str { #1 }

```

```

1369 \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1370   \__stex_smsmode_unset_codes:
1371   \begin{#1}
1372 }
1373 }

```

(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.

```

1374 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1375   \str_set:Nn \l_tmpa_str { #1 }
1376   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1377     \end{#1}
1378   }
1379 }

```

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

## 18.2 Inheritance

```

1380 <@@=stex_importmodule>

```

\stex\_import\_module\_uri:nn

```

1381 \cs_new_protected:Nn \stex_import_module_uri:nn {
1382   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1383   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1384   \str_if_empty:NT \l__stex_importmodule_archive_str {
1385     \prop_if_empty:NF \l_stex_current_repository_prop {
1386       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1387     }
1388   }
1389
1390   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1391   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1392   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1393
1394   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1395     \stex_modules_current_namespace:
1396     \str_if_empty:NF \l__stex_importmodule_path_str {
1397       \str_set:Nx \l_stex_module_ns_str {
1398         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1399       }
1400     }
1401   }{
1402     \stex_require_repository:n \l__stex_importmodule_archive_str
1403     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str_manifest_prop } { ns }
1404     \l_stex_module_ns_str
1405     \str_if_empty:NF \l__stex_importmodule_path_str {
1406       \str_set:Nx \l_stex_module_ns_str {
1407         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1408       }
1409     }
1410   }
1411 }

```

(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 1412 \str_new:N \l__stex_importmodule_name_str
\l__stex_importmodule_path_str 1413 \str_new:N \l__stex_importmodule_archive_str
\l__stex_importmodule_file_str 1414 \str_new:N \l__stex_importmodule_path_str
1415 \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)}
1416 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1417 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1418
1419 % archive
1420 \str_set:Nx \l_tmpa_str { #2 }
1421 \str_if_empty:NTF \l_tmpa_str {
1422 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1423 } {
1424 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1425 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1426 \seq_put_right:Nn \l_tmpa_seq { source }
1427 }
1428
1429 % path
1430 \str_set:Nx \l_tmpb_str { #3 }
1431 \str_if_empty:NTF \l_tmpb_str {
1432 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1433
1434 \ltx@ifpackageloaded{babel} {
1435 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1436 { \languagename } \l_tmpb_str {
1437 \msg_error:nnn{stex}{error/unknownlanguage}{\languagename}
1438 }
1439 } {
1440 \str_clear:N \l_tmpb_str
1441 }
1442
1443 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1444 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1445 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1446 }{
1447 \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1448 \IfFileExists{ \l_tmpa_str.tex }{
1449 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1450 }{
1451 % try english as default
1452 \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1453 \IfFileExists{ \l_tmpa_str.en.tex }{
1454 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1455 }{
1456 \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1457 }
1458 }

```

```

1459     }
1460
1461   } {
1462     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1463     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1464
1465     \ltx@ifpackageloaded{babel} {
1466       \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1467         { \language } \l_tmpb_str {
1468         \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1469       }
1470     } {
1471       \str_clear:N \l_tmpb_str
1472     }
1473
1474     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1475
1476     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1477     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1478       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1479     }{
1480       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1481       \IfFileExists{ \l_tmpa_str/#4.tex }{
1482         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1483       }{
1484         % try english as default
1485         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1486         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1487           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1488         }{
1489           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1490           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1491             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1492           }{
1493             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1494             \IfFileExists{ \l_tmpa_str.tex }{
1495               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1496             }{
1497               % try english as default
1498               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1499               \IfFileExists{ \l_tmpa_str.en.tex }{
1500                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1501               }{
1502                 \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1503               }
1504             }
1505           }
1506         }
1507       }
1508     }
1509   }
1510
1511   \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1512   \seq_clear:N \g_stex_modules_in_file_seq

```

```

1513 % \exp_args:Nnx \use:nn {
1514 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1515 \seq_clear:N \l_stex_all_modules_seq
1516 \prop_clear:N \l_stex_current_module_prop
1517 \str_set:Nx \l_tmpb_str { #2 }
1518 \str_if_empty:NF \l_tmpb_str {
1519 \stex_set_current_repository:n { #2 }
1520 }
1521 \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1522 \input { \g__stex_importmodule_file_str }
1523 }
1524 % }{
1525
1526 % }
1527 \prop_gput:Noo \g_stex_module_files_prop
1528 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1529 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1530
1531 \stex_if_module_exists:nF { #1 ? #4 } {
1532 \msg_error:nnn{stex}{error/unknownmodule}{
1533 #1?#4~(in~file~\g__stex_importmodule_file_str)
1534 }
1535 }
1536 }
1537 \stex_activate_module:n { #1 ? #4 }
1538 }

```

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

## `\importmodule`

```

1539 \NewDocumentCommand \importmodule { 0{ } m } {
1540 \stex_import_module_uri:nn { #1 } { #2 }
1541 \stex_debug:nn{modules}{Importing~module:~
1542 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1543 }
1544 \stex_if_smsmode:F {
1545 \stex_import_require_module:nnnn
1546 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1547 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1548 \stex_annotate_invisible:nnn
1549 {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1550 }
1551 \exp_args:Nx \stex_add_to_current_module:n {
1552 \stex_import_require_module:nnnn
1553 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1554 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1555 }
1556 \exp_args:Nx \stex_add_import_to_current_module:n {
1557 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1558 }
1559 \stex_smsmode_set_codes:
1560 }
1561 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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



**\usemodule**

```
1562 \NewDocumentCommand \usemodule { 0{} m } {  
1563   \stex_if_smsmode:F {  
1564     \stex_import_module_uri:nn { #1 } { #2 }  
1565     \stex_import_require_module:nnnn  
1566     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }  
1567     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }  
1568     \stex_annotate_invisible:nnn  
1569     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}  
1570   }  
1571   \stex_smsmode_set_codes:  
1572 }
```

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

```
1573 \endpackage
```

## Chapter 19

# STEX -Symbols Implementation

```
1574 <*package>
1575
1576 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1577
```

Warnings and error messages

```
1578
```

### 19.1 Symbol Declarations

```
1579 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1580 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1581 \NewDocumentCommand \STEXsymbol { m } {
1582   \stex_get_symbol:n { #1 }
1583   \exp_args:No
1584   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1585 }
```

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

symdecl arguments:

```
1586 \keys_define:nn { stex / symdecl } {
1587   name      .str_set:N = \l_stex_symdecl_name_str ,
1588   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1589   args      .str_set:N = \l_stex_symdecl_args_str ,
1590   type      .tl_set:N  = \l_stex_symdecl_type_tl ,
1591   align     .str_set:N  = \l_stex_symdecl_align_str , % TODO(?)
1592   gfc       .str_set:N  = \l_stex_symdecl_gfc_str , % TODO(?)
1593   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1594   def       .tl_set:N  = \l_stex_symdecl_definiens_tl
1595 }
```

```

1596
1597 \bool_new:N \l_stex_symdecl_make_macro_bool
1598
1599 \cs_new_protected:Nn \__stex_symdecl_args:n {
1600   \str_clear:N \l_stex_symdecl_name_str
1601   \str_clear:N \l_stex_symdecl_args_str
1602   \bool_set_false:N \l_stex_symdecl_local_bool
1603   \tl_clear:N \l_stex_symdecl_type_tl
1604   \tl_clear:N \l_stex_symdecl_definiens_tl
1605
1606   \keys_set:nn { stex / symdecl } { #1 }
1607 }

```

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

```

1608
1609 \NewDocumentCommand \symdecl { s O{} m } {
1610   \__stex_symdecl_args:n { #2 }
1611   \IfBooleanTF #1 {
1612     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1613   } {
1614     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1615   }
1616   \stex_symdecl_do:n { #3 }
1617   \stex_smsmode_set_codes:
1618 }
1619 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1620 \cs_new_protected:Nn \stex_symdecl_do:n {
1621   \stex_if_in_module:F {
1622     % TODO throw error? some default namespace?
1623   }
1624
1625   \str_if_empty:NT \l_stex_symdecl_name_str {
1626     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1627   }
1628
1629   \prop_if_exist:cT { g_stex_symdecl_
1630     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1631     \prop_item:Nn \l_stex_current_module_prop {name} ?
1632     \l_stex_symdecl_name_str
1633     _prop
1634   }{
1635     % TODO throw error (beware of circular dependencies)
1636   }
1637
1638   \prop_clear:N \l_tmpa_prop
1639   \prop_put:Nnx \l_tmpa_prop { module } {
1640     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1641     \prop_item:Nn \l_stex_current_module_prop {name}
1642   }

```

```

1643 \seq_clear:N \l_tmpa_seq
1644 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1645 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1646 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1647 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1648
1649 \exp_args:No \stex_add_constant_to_current_module:n {
1650   \l_stex_symdecl_name_str
1651 }
1652
1653 % arity/args
1654 \int_zero:N \l_tmpb_int
1655
1656 \bool_set_true:N \l_tmpa_bool
1657 \str_map_inline:Nn \l_stex_symdecl_args_str {
1658   \token_case_meaning:NnF ##1 {
1659     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1660     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1661     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1662     {\tl_to_str:n a} {
1663       \bool_set_false:N \l_tmpa_bool
1664       \int_incr:N \l_tmpb_int
1665     }
1666     {\tl_to_str:n B} {
1667       \bool_set_false:N \l_tmpa_bool
1668       \int_incr:N \l_tmpb_int
1669     }
1670   }{
1671     \msg_set:nnn{stex}{error/wrongargs}{
1672       args~value~in~symbol~declaration~for~
1673       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1674       \prop_item:Nn \l_stex_current_module_prop {name} ?
1675       \l_stex_symdecl_name_str ~
1676       needs~to~be~
1677       i,~a,~b~or~B,~but~##1~given
1678     }
1679     \msg_error:nn{stex}{error/wrongargs}
1680   }
1681 }
1682 \bool_if:NTF \l_tmpa_bool {
1683   % possibly numeric
1684   \str_if_empty:NTF \l_stex_symdecl_args_str {
1685     \prop_put:Nnn \l_tmpa_prop { args } {}
1686     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1687   }{
1688     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1689     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1690     \str_clear:N \l_tmpa_str
1691     \int_step_inline:nn \l_tmpa_int {
1692       \str_put_right:Nn \l_tmpa_str i
1693     }
1694     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1695   }
1696 } {

```

```

1697 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1698 \prop_put:Nnx \l_tmpa_prop { arity }
1699 { \str_count:N \l_stex_symdecl_args_str }
1700 }
1701 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1702
1703
1704 % semantic macro
1705
1706 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1707   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1708     \prop_item:Nn \l_tmpa_prop { module } ?
1709     \prop_item:Nn \l_tmpa_prop { name }
1710   } }
1711
1712   \bool_if:NF \l_stex_symdecl_local_bool {
1713     \exp_args:Nx \stex_add_to_current_module:n {
1714       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1715         \prop_item:Nn \l_tmpa_prop { module } ?
1716         \prop_item:Nn \l_tmpa_prop { name }
1717       } }
1718     }
1719   }
1720 }
1721
1722 % add to all symbols
1723
1724 \bool_if:NF \l_stex_symdecl_local_bool {
1725   \exp_args:Nx \stex_add_to_current_module:n {
1726     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1727       \prop_item:Nn \l_tmpa_prop { module } ?
1728       \prop_item:Nn \l_tmpa_prop { name }
1729     }
1730   }
1731 }
1732
1733 \stex_debug:nn{symbols}{New~symbol:~
1734   \prop_item:Nn \l_tmpa_prop { module } ?
1735   \prop_item:Nn \l_tmpa_prop { name } ^^J
1736   Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1737   Args:~\prop_item:Nn \l_tmpa_prop { args }
1738 }
1739
1740 % circular dependencies require this:
1741
1742 \prop_if_exist:cF {
1743   g_stex_symdecl_
1744   \prop_item:Nn \l_tmpa_prop { module } ?
1745   \prop_item:Nn \l_tmpa_prop { name }
1746   _prop
1747 } {
1748   \prop_gset_eq:cN {
1749     g_stex_symdecl_
1750     \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1751     \prop_item:Nn \l_tmpa_prop { name }
1752     _prop
1753   } \l_tmpa_prop
1754 }
1755
1756 \stex_if_smsmode:TF {
1757   \bool_if:NF \l_stex_symdecl_local_bool {
1758     \exp_args:Nx \stex_add_to_sms:n {
1759       \prop_gset_from_keyval:cn {
1760         g_stex_symdecl_
1761         \prop_item:Nn \l_tmpa_prop { module } ?
1762         \prop_item:Nn \l_tmpa_prop { name }
1763         _prop
1764       } {
1765         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1766         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1767         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1768         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1769         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1770         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1771         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1772         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1773       }
1774       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1775         \prop_item:Nn \l_tmpa_prop { module } ?
1776         \prop_item:Nn \l_tmpa_prop { name }
1777       }
1778     }
1779   }
1780 }{
1781   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1782     \prop_item:Nn \l_tmpa_prop { module } ?
1783     \prop_item:Nn \l_tmpa_prop { name }
1784   }
1785   \stex_if_do_html:T {
1786     \stex_annotate_invisible:nnn {symdecl} {
1787       \prop_item:Nn \l_tmpa_prop { module } ?
1788       \prop_item:Nn \l_tmpa_prop { name }
1789     } {
1790       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1791       \stex_annotate_invisible:nnn{args}{}{
1792         \prop_item:Nn \l_tmpa_prop { args }
1793       }
1794       \stex_annotate_invisible:nnn{macroname}{}{#1}
1795       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1796         \stex_annotate_invisible:nnn{definiens}{}
1797         {\l_stex_symdecl_definiens_tl$}
1798       }
1799     }
1800   }
1801 }
1802 }

```

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

`\stex_get_symbol:n`

```
1803 \str_new:N \l_stex_get_symbol_uri_str
1804
1805 \cs_new_protected:Nn \stex_get_symbol:n {
1806   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1807     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1808   }{
1809     % argument is a string
1810     % is it a command name?
1811     \cs_if_exist:cTF { #1 }{
1812       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1813       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1814       \str_if_empty:NTF \l_tmpa_str {
1815         \exp_args:Nx \cs_if_eq:NNTF {
1816           \tl_head:N \l_tmpa_tl
1817         } \stex_invoke_symbol:n {
1818           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1819         }{
1820           \__stex_symdecl_get_symbol_from_string:n { #1 }
1821         }
1822       } {
1823         \__stex_symdecl_get_symbol_from_string:n { #1 }
1824       }
1825     }{
1826       % argument is not a command name
1827       \__stex_symdecl_get_symbol_from_string:n { #1 }
1828       % \l_stex_all_symbols_seq
1829     }
1830   }
1831 }
1832
1833 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1834   \str_set:Nn \l_tmpa_str { #1 }
1835   \bool_set_false:N \l_tmpa_bool
1836   \stex_if_in_module:T {
1837     \prop_get:NnN \l_stex_current_module_prop
1838     { constants } \l_tmpa_seq
1839     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1840       \bool_set_true:N \l_tmpa_bool
1841       \str_set:Nx \l_stex_get_symbol_uri_str {
1842         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1843         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1844       }
1845     }
1846   }
1847   \bool_if:NF \l_tmpa_bool {
1848     \tl_set:Nn \l_tmpa_tl {
1849       \msg_set:nnn{stex}{error/unknownsymbol}{
1850         No~symbol~#1~found!
1851       }
1852     }
1853     \msg_error:nn{stex}{error/unknownsymbol}
1854   }
1855   \str_set:Nn \l_tmpa_str { #1 }
1856   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1856 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1857   \str_set:Nn \l_tmpb_str { ##1 }
1858   \str_if_eq:eeT { \l_tmpa_str } {
1859     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1860   } {
1861     \seq_map_break:n {
1862       \tl_set:Nn \l_tmpa_tl {
1863         \str_set:Nn \l_stex_get_symbol_uri_str {
1864           ##1
1865         }
1866       }
1867     }
1868   }
1869 }
1870 \l_tmpa_tl
1871 }
1872 }
1873
1874 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1875   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1876   { \tl_tail:N \l_tmpa_tl }
1877   \tl_if_single:NTF \l_tmpa_tl {
1878     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1879       \exp_after:wN \str_set:Nn \exp_after:wN
1880       \l_stex_get_symbol_uri_str \l_tmpa_tl
1881     }{
1882       % TODO
1883       % tail is not a single group
1884     }
1885   }{
1886     % TODO
1887     % tail is not a single group
1888   }
1889 }

```

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

## 19.2 Notations

```

1890 <@@=stex_notation>
1891 notation arguments:
1892 \keys_define:nn { stex / notation } {
1893   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1894   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1895   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1896   op .tl_set:N = \l__stex_notation_op_tl ,
1897   unknown .code:n = \str_set:Nx
1898     \l__stex_notation_variant_str \l_keys_key_str
1899 }
1900 \cs_new_protected:Nn \__stex_notation_args:n {
1901   \str_clear:N \l__stex_notation_lang_str
1902   \str_clear:N \l__stex_notation_variant_str

```



```

1903 \str_clear:N \l__stex_notation_prec_str
1904 \tl_clear:N \l__stex_notation_op_tl
1905
1906 \keys_set:nn { stex / notation } { #1 }
1907 }

```

## **\notation**

```

1908 \NewDocumentCommand \notation { 0{ } m } {
1909   \__stex_notation_args:n { #1 }
1910   \tl_clear:N \l_stex_symdecl_definiens_tl
1911   \stex_get_symbol:n { #2 }
1912   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1913 }
1914 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

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

```

1915 \cs_new_protected:Nn \stex_notation_do:nn {
1916   \prop_set_eq:Nc \l_tmpa_prop {
1917     g_stex_symdecl_ #1 _prop
1918   }
1919
1920   \prop_clear:N \l_tmpb_prop
1921   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1922   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1923   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1924
1925   % precedences
1926   \seq_clear:N \l_tmpb_seq
1927   \exp_args:NNno
1928   \str_if_empty:NTF \l__stex_notation_prec_str {
1929     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1930     \int_compare:nNnTF \l_tmpa_str = 0 {
1931       \exp_args:NNnx
1932       \prop_put:Nno \l_tmpb_prop { opprec }
1933       { \neginfprec }
1934     }{
1935       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1936     }
1937   } {
1938     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1939       \exp_args:NNnx
1940       \prop_put:Nno \l_tmpb_prop { opprec }
1941       { \neginfprec }
1942       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1943       \int_step_inline:nn { \l_tmpa_str } {
1944         \exp_args:NNx
1945         \seq_put_right:Nn \l_tmpb_seq { \infprec }
1946       }
1947     }{
1948       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1949       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1950         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1951         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

1952         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1953         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
1954         \seq_map_inline:Nn \l_tmpa_seq {
1955             \seq_put_right:Nn \l_tmpb_seq { ##1 }
1956         }
1957     }
1958     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1959 }{
1960     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1961     \int_compare:nNnTF \l_tmpa_str = 0 {
1962         \exp_args:NNnx
1963         \prop_put:Nno \l_tmpb_prop { opprec }
1964         { \infprec }
1965     }{
1966         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1967     }
1968 }
1969 }
1970 }
1971
1972 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1973 \int_step_inline:nn { \l_tmpa_str } {
1974     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
1975         \exp_args:NNx
1976         \seq_put_right:Nn \l_tmpb_seq {
1977             \prop_item:Nn \l_tmpb_prop { opprec }
1978         }
1979     }
1980 }
1981
1982 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1983 \tl_clear:N \l_tmpa_tl
1984
1985 \int_compare:nNnTF \l_tmpa_str = 0 {
1986     \exp_args:NNe
1987     \cs_set:Npn \l__stex_notation_macrocode_cs {
1988         \_stex_term_math_oms:nnnn { #1 }
1989         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1990         { \prop_item:Nn \l_tmpb_prop { opprec } }
1991         { \exp_not:n { #2 } }
1992     }
1993     \__stex_notation_final:
1994 }{
1995     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1996     \str_if_in:NnTF \l_tmpb_str b {
1997         \exp_args:Nne \use:nn
1998         {
1999             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2000             \cs_set:Npn \l_tmpa_str { {
2001                 \_stex_term_math_omb:nnnn { #1 }
2002                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2003                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2004                 { \exp_not:n { #2 } }
2005             }
2006         }
2007     }

```

```

2006   }{
2007     \str_if_in:NnTF \l_tmpb_str B {
2008       \exp_args:Nne \use:nn
2009       {
2010         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2011         \cs_set:Npn \l_tmpa_str } { {
2012           \_stex_term_math_omb:nnnn { #1 }
2013           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2014           { \prop_item:Nn \l_tmpb_prop { opprec } }
2015           { \exp_not:n { #2 } }
2016         } }
2017       }{
2018         \exp_args:Nne \use:nn
2019         {
2020         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2021         \cs_set:Npn \l_tmpa_str } { {
2022           \_stex_term_math_oma:nnnn { #1 }
2023           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2024           { \prop_item:Nn \l_tmpb_prop { opprec } }
2025           { \exp_not:n { #2 } }
2026         } }
2027       }
2028     }
2029
2030     \int_zero:N \l_tmpa_int
2031     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2032     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2033     \__stex_notation_arguments:
2034   }
2035 }

```

(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

```

2036 \cs_new_protected:Nn \__stex_notation_arguments: {
2037   \int_incr:N \l_tmpa_int
2038   \str_if_empty:NnTF \l_tmpa_str {
2039     \__stex_notation_final:
2040   }{
2041     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2042     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2043     \str_if_eq:VnTF \l_tmpb_str a {
2044       \__stex_notation_argument_assoc:n
2045     }{
2046       \str_if_eq:VnTF \l_tmpb_str B {
2047         \__stex_notation_argument_assoc:n
2048       }{
2049         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2050         \tl_put_right:Nx \l_tmpa_tl {
2051           { \_stex_term_math_arg:nnn
2052             { \int_use:N \l_tmpa_int }
2053             { \l_tmpb_str }
2054             { ####\int_use:N \l_tmpa_int }
2055           }

```

```

2056     }
2057     \__stex_notation_arguments:
2058   }
2059 }
2060 }
2061 }

```

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

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

```

2062 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2063   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2064   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2065   \tl_put_right:Nx \l_tmpa_tl {
2066     { \stex_term_math_assoc_arg:nnnn
2067       { \int_use:N \l_tmpa_int }
2068       { \l_tmpb_str }
2069       \exp_args:No \exp_not:n
2070       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2071       { ####\int_use:N \l_tmpa_int }
2072     }
2073   }
2074   \__stex_notation_arguments:
2075 }

```

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

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

```

2076 \cs_new_protected:Nn \__stex_notation_final: {
2077   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2078   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2079   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2080   \exp_args:Nne \use:nn
2081   {
2082     \cs_generate_from_arg_count:cNnn {
2083       stex_notation_ \l_tmpa_str \c_hash_str
2084       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2085       _cs
2086     }
2087     \cs_gset:Npn \l_tmpb_str } { {
2088       \exp_after:wN \exp_after:wN \exp_after:wN
2089       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2090       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2091     } }
2092
2093   \tl_if_empty:NF \l__stex_notation_op_tl {
2094     \cs_gset:cpx {
2095       stex_op_notation_ \l_tmpa_str \c_hash_str
2096       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2097       _cs
2098     } {
2099       \stex_term_oms:nnn {
2100         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2101         \l__stex_notation_lang_str

```

```

2102     }{
2103         \l_tmpa_str
2104     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2105 }
2106 }
2107
2108
2109
2110 \stex_debug:nn{symbols}{
2111     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2112     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2113     Operator~precedence:~
2114     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2115     Argument~precedences:~
2116     \seq_use:Nn \l_tmpa_seq {,~}^^J
2117     Notation: \cs_meaning:c {
2118         stex_notation_ \l_tmpa_str \c_hash_str
2119         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2120         _cs
2121     }
2122 }
2123
2124 \prop_gset_eq:cN {
2125     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2126     \c_hash_str \l__stex_notation_lang_str _prop
2127 } \l_tmpb_prop
2128
2129 \exp_args:Nx
2130 \stex_add_to_current_module:n {
2131     \prop_get:cnN {
2132         g_stex_symdecl_
2133         \prop_item:Nn \l_tmpb_prop { symbol }
2134         _prop
2135     } { notations } \exp_not:N \l_tmpa_seq
2136     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2137         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2138     }
2139     \prop_put:cno {
2140         g_stex_symdecl_
2141         \prop_item:Nn \l_tmpb_prop { symbol }
2142         _prop
2143     } { notations } \exp_not:N \l_tmpa_seq
2144 }
2145
2146 \stex_if_smsmode:TF {
2147     \stex_smsmode_set_codes:
2148     \exp_args:Nx \stex_add_to_sms:n {
2149         \prop_gset_from_keyval:cn {
2150             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2151             \c_hash_str \l__stex_notation_lang_str _prop
2152         } {
2153             symbol = \prop_item:Nn \l_tmpb_prop { symbol } ,
2154             language = \prop_item:Nn \l_tmpb_prop { language } ,
2155             variant = \prop_item:Nn \l_tmpb_prop { variant } ,

```

```

2156         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2157         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2158     }
2159 }
2160 }{
2161     \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2162     \seq_put_right:Nx \l_tmpa_seq {
2163         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2164     }
2165     \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2166     \prop_set_eq:cN {
2167         g_stex_symdecl_ \l_tmpa_str _prop
2168     } \l_tmpa_prop
2169
2170     % HTML annotations
2171     \stex_if_do_html:T {
2172         \stex_annotate_invisible:nnn { notation }
2173         { \prop_item:Nn \l_tmpb_prop { symbol } } {
2174             \stex_annotate_invisible:nnn { notationfragment }
2175             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2176             \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2177             \stex_annotate_invisible:nnn { precedence }
2178             { \prop_item:Nn \l_tmpb_prop { opprec } ;
2179               \seq_use:Nn \l_tmpa_seq { x }
2180             }{}
2181
2182             \int_zero:N \l_tmpa_int
2183             \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2184             \tl_clear:N \l_tmpa_tl
2185             \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
2186                 \int_incr:N \l_tmpa_int
2187                 \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2188                 \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2189                 \str_if_eq:VnTF \l_tmpb_str a {
2190                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2191                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2192                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2193                     } }
2194                 }{
2195                     \str_if_eq:VnTF \l_tmpb_str B {
2196                         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2197                             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2198                             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2199                         } }
2200                     }{
2201                         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2202                             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2203                         } }
2204                     }
2205                 }
2206             }
2207             \stex_annotate_invisible:nnn { notationcomp }{}{
2208                 $ \exp_args:Nno \use:nn { \use:c {
2209                     stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2210         \c_hash_str \l__stex_notation_variant_str
2211         \c_hash_str \l__stex_notation_lang_str _cs
2212     } } { \l_tmpa_tl } $
2213   }
2214 }
2215 }
2216 }
2217 }

```

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

**\symdef**

```

2218 \keys_define:nn { stex / symdef } {
2219   name .str_set_x:N = \l_stex_symdecl_name_str ,
2220   local .bool_set:N = \l_stex_symdecl_local_bool ,
2221   args .str_set_x:N = \l_stex_symdecl_args_str ,
2222   type .tl_set:N = \l_stex_symdecl_type_tl ,
2223   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2224   op .tl_set:N = \l__stex_notation_op_tl ,
2225   lang .str_set_x:N = \l__stex_notation_lang_str ,
2226   variant .str_set_x:N = \l__stex_notation_variant_str ,
2227   prec .str_set_x:N = \l__stex_notation_prec_str ,
2228   unknown .code:n = \str_set:Nx
2229     \l__stex_notation_variant_str \l_keys_key_str
2230 }
2231
2232 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2233   \str_clear:N \l_stex_symdecl_name_str
2234   \str_clear:N \l_stex_symdecl_args_str
2235   \bool_set_false:N \l_stex_symdecl_local_bool
2236   \tl_clear:N \l_stex_symdecl_type_tl
2237   \tl_clear:N \l_stex_symdecl_definiens_tl
2238   \str_clear:N \l__stex_notation_lang_str
2239   \str_clear:N \l__stex_notation_variant_str
2240   \str_clear:N \l__stex_notation_prec_str
2241   \tl_clear:N \l__stex_notation_op_tl
2242
2243   \keys_set:nn { stex / symdef } { #1 }
2244 }
2245
2246 \NewDocumentCommand \symdef { 0{} m } {
2247   \_stex_notation_symdef_args:n { #1 }
2248   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2249   \stex_symdecl_do:n { #2 }
2250   \exp_args:Nx \stex_notation_do:nn {
2251     \prop_item:Nn \l_tmpa_prop { module } ?
2252     \prop_item:Nn \l_tmpa_prop { name }
2253   }
2254 }
2255 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2256 \</package>

```

## Chapter 20

# STEX -Terms Implementation

```
2257 <*package>
2258
2259 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2260
2261 <@@=stex_terms>
2262
2263   Warnings and error messages
2264   \msg_new:nnn{stex}{error/nonotation}{
2265     Symbol~#1~invoked,~but~has~no~notation#2!
2266   }
2267   \msg_new:nnn{stex}{error/notationarg}{
2268     Error~in~parsing~notation~#1
2269   }
2270
```

### 20.1 Symbol Invocations

Arguments:

```
2269 \keys_define:nn { stex / terms } {
2270   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2271   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2272   unknown .code:n = \str_set:Nx
2273     \l__stex_terms_variant_str \l_keys_key_str
2274 }
2275
2276 \cs_new_protected:Nn \__stex_terms_args:n {
2277   \str_clear:N \l__stex_terms_lang_str
2278   \str_clear:N \l__stex_terms_variant_str
2279   \str_clear:N \l__stex_terms_prec_str
2280   \tl_clear:N \l__stex_terms_op_tl
2281
2282   \keys_set:nn { stex / terms } { #1 }
2283 }
```

`\stex_invoke_symbol:n` Invokes a semantic macro



```

2284 \cs_new_protected:Nn \stex_invoke_symbol:n {
2285   \if_mode_math:
2286     \exp_after:wN \__stex_terms_invoke_math:n
2287   \else:
2288     \exp_after:wN \__stex_terms_invoke_text:n
2289   \fi: { #1 }
2290 }

```

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

`\__stex_terms_invoke_math:n`

```

2291 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2292   \peek_charcode_remove:NTF ! {
2293     \peek_charcode:NTF [ {
2294       \__stex_terms_invoke_op:nw { #1 }
2295     }{
2296       \__stex_terms_invoke_op:nw { #1 } []
2297     }
2298   }{
2299     \peek_charcode_remove:NTF * {
2300       \__stex_terms_invoke_text:n { #1 }
2301     }{
2302       \peek_charcode:NTF [ {
2303         \__stex_terms_invoke_math:nw { #1 }
2304       }{
2305         \__stex_terms_invoke_math:nw { #1 } []
2306       }
2307     }
2308   }
2309 }

```

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

`\__stex_terms_invoke_op:nw`

```

2310 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2311   \__stex_terms_args:n { #2 }
2312   \cs_if_exist:cTF {
2313     stex_op_notation_ #1 \c_hash_str
2314     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2315   }{
2316     \csname stex_op_notation_ #1 \c_hash_str
2317     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2318   \endcsname
2319   }{
2320     % TODO throw error
2321   }
2322 }

```

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

`\__stex_terms_invoke_math:nw`

```

2323 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2324   \__stex_terms_args:n { #2 }
2325   \prop_set_eq:Nc \l_tmpa_prop {
2326     g_stex_symdecl_ #1 _prop

```

```

2327 }
2328 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2329 \seq_if_empty:NTF \l_tmpa_seq {
2330   \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2331 } {
2332   \seq_if_in:NxTF \l_tmpa_seq
2333   { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2334     \use:c{
2335       stex_notation_ #1 \c_hash_str
2336       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2337       _cs
2338     }
2339   }{
2340     \str_if_empty:NTF \l__stex_terms_variant_str {
2341       \str_if_empty:NTF \l__stex_terms_lang_str {
2342         \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2343         \use:c{
2344           stex_notation_ #1 \c_hash_str \l_tmpa_str
2345           _cs
2346         }
2347       }{
2348         \msg_error:nn{stex}{error/nonotation}{#1}{
2349           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2350         }
2351       }
2352     }{
2353       \msg_error:nn{stex}{error/nonotation}{#1}{
2354         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2355       }
2356     }
2357   }
2358 }
2359 }

```

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

`\__stex_terms_invoke_text:n`

```

2360 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2361   \peek_charcode_remove:NTF ! {
2362     \stex_term_custom:nn { #1 } { }
2363   }{
2364     \prop_set_eq:Nc \l_tmpa_prop {
2365       g_stex_symdecl_ #1 _prop
2366     }
2367     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2368     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2369   }
2370 }

```

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

## 20.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2371 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2372 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2373 \int_new:N \l__stex_terms_downprec
2374 \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
2375 \tl_set:Nn \l__stex_terms_left_bracket_str (
2376 \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
2377 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2378 \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2379 \bool_if:NTF \l_stex_inarray_bool { #2 }{
2380 \dobrackets { #2 }
2381 }
2382 }{ #2 }
2383 }

(End definition for \__stex_terms_maybe_brackets:nn.)

```

```

\dobrackets
2384 %\RequirePackage{scalerel}
2385 \cs_new_protected:Npn \dobrackets #1 {
2386 %\ThisStyle{\if D@m@switch
2387 % \exp_args:Nnx \use:nn
2388 % { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2389 % { \exp_not:N\right\l__stex_terms_right_bracket_str }
2390 % \else
2391 \exp_args:Nnx \use:nn
2392 { \l__stex_terms_left_bracket_str #1 }
2393 { \l__stex_terms_right_bracket_str }
2394 %\fi}
2395 }

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

```

```

\withbrackets
2396 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2397 \exp_args:Nnx \use:nn
2398 {
2399 \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2400 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2401 #3
2402 }
2403 {
2404 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2405 {\l__stex_terms_left_bracket_str}
2406 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str

```

```

2407     {\l__stex_terms_right_bracket_str}
2408   }
2409 }

```

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

#### `\STEXinvisible`

```

2410 \cs_new_protected:Npn \STEXinvisible #1 {
2411   \stex_annotate_invisible:n { #1 }
2412 }

```

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

OMDoc terms:

#### `\_stex_term_math_oms:nnnn`

```

2413 \cs_new_protected:Nn \_stex_term_oms:nnn {
2414   \stex_annotate:nnn{ OMID }{ #2 }{
2415     \stex_highlight_term:nn { #1 } { #3 }
2416   }
2417 }
2418
2419 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2420   \__stex_terms_maybe_brackets:nn { #3 }{
2421     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2422   }
2423 }

```

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

#### `\_stex_term_math_oma:nnnn`

```

2424 \cs_new_protected:Nn \_stex_term_oma:nnn {
2425   \stex_annotate:nnn{ OMA }{ #2 }{
2426     \stex_highlight_term:nn { #1 } { #3 }
2427   }
2428 }
2429
2430 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2431   \__stex_terms_maybe_brackets:nn { #3 }{
2432     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2433   }
2434 }

```

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

#### `\_stex_term_math_omb:nnnn`

```

2435 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2436   \stex_annotate:nnn{ OMBIND }{ #2 }{
2437     \stex_highlight_term:nn { #1 } { #3 }
2438   }
2439 }
2440
2441 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2442   \__stex_terms_maybe_brackets:nn { #3 }{
2443     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2444   }
2445 }

```

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

`\_stex_term_math_arg:nnn`

```

2446 \cs_new_protected:Nn \_stex_term_arg:nn {
2447   \stex_unhighlight_term:n {
2448     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2449   }
2450 }
2451 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2452   \exp_args:Nnx \use:nn
2453     { \int_set:Nn \l__stex_terms_downprec { #2 }
2454       \_stex_term_arg:nn { #1 }{ #3 }
2455     }
2456   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2457 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

2458 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2459   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2460   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2461     \tl_set:Nn \l_tmpa_tl { #4 }
2462   }{
2463     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2464     \seq_reverse:N \l_tmpa_seq
2465     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2466     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2467
2468     \seq_map_inline:Nn \l_tmpa_seq {
2469       \exp_args:NNo \tl_set:No \l_tmpa_tl {
2470         \exp_args:Nno
2471         \l_tmpa_cs { ##1 } \l_tmpa_tl
2472       }
2473     }
2474
2475   }
2476   \exp_args:Nnno
2477   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2478 }

```

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

`\stex_term_custom:nn`

```

2479 \cs_new_protected:Nn \stex_term_custom:nn {
2480   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2481   \str_set:Nn \l_tmpa_str { #2 }
2482   \tl_clear:N \l_tmpa_tl
2483   \int_zero:N \l_tmpa_int
2484   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2485   \__stex_terms_custom_loop:
2486 }

```

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

\\_stex\_terms\_custom\_loop:

```

2487 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2488   \bool_set_false:N \l_tmpa_bool
2489   \bool_while_do:nn {
2490     \str_if_eq_p:ee X {
2491       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2492     }
2493   }{
2494     \int_incr:N \l_tmpa_int
2495   }
2496
2497   \peek_charcode:NTF [ {
2498     % notation/text component
2499     \_stex_terms_custom_component:w
2500   } {
2501     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2502       % all arguments read => finish
2503       \_stex_terms_custom_final:
2504     } {
2505       % arguments missing
2506       \peek_charcode_remove:NTF * {
2507         % invisible, specific argument position or both
2508         \peek_charcode:NTF [ {
2509           % visible specific argument position
2510           \_stex_terms_custom_arg:wn
2511         } {
2512           % invisible
2513           \peek_charcode_remove:NTF * {
2514             % invisible specific argument position
2515             \_stex_terms_custom_arg_inv:wn
2516           } {
2517             % invisible next argument
2518             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2519           }
2520         }
2521       } {
2522         % next normal argument
2523         \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2524       }
2525     }
2526   }
2527 }

```

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

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

```

2528 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2529   \bool_set_true:N \l_tmpa_bool
2530   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2531 }

```

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

\\_stex\_terms\_custom\_arg:wn

```

2532 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2533   \str_set:Nx \l_tmpb_str {
2534     \str_item:Nn \l_tmpa_str { #1 }
2535   }
2536   \str_case:VnTF \l_tmpb_str {
2537     { X } {
2538       \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2539     }
2540     { i } { \__stex_terms_custom_set_X:n { #1 } }
2541     { b } { \__stex_terms_custom_set_X:n { #1 } }
2542     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2543     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2544   }{}{
2545     \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2546   }
2547
2548   \bool_if:nTF \l_tmpa_bool {
2549     \tl_put_right:Nx \l_tmpa_tl {
2550       \stex_annotate_invisible:n {
2551         \stex_term_arg:nn { \int_eval:n { #1 } }
2552         \exp_not:n { { #2 } }
2553       }
2554     }
2555   } {
2556     \tl_put_right:Nx \l_tmpa_tl {
2557       \stex_term_arg:nn { \int_eval:n { #1 } }
2558       \exp_not:n { { #2 } }
2559     }
2560   }
2561
2562   \__stex_terms_custom_loop:
2563 }

```

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

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

```

2564 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2565   \str_set:Nx \l_tmpa_str {
2566     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2567     X
2568     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2569   }
2570 }

```

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

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

```

2571 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2572   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2573   \__stex_terms_custom_loop:
2574 }

```

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

`\_stex_terms_custom_final:`

```

2575 \cs_new_protected:Nn \_stex_terms_custom_final: {
2576   \int_compare:nNnTF \l_tmpb_int = 0 {
2577     \exp_args:Nnno \_stex_term_oms:nnn
2578   }{
2579     \str_if_in:NnTF \l_tmpa_str {b} {
2580       \exp_args:Nnno \_stex_term_ombind:nnn
2581     } {
2582       \exp_args:Nnno \_stex_term_oma:nnn
2583     }
2584   }
2585   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2586 }

```

(End definition for `\_stex_terms_custom_final:.`)

**`\symref`**

**`\symname`**

```

2587 \NewDocumentCommand \symref { m m }{
2588   \let\compemph_uri_prev:\compemph@uri
2589   \let\compemph@uri\symrefemph@uri
2590   \STEXsymbol{#1}![#2]
2591   \let\compemph@uri\compemph_uri_prev:
2592 }
2593
2594 \keys_define:nn { stex / symname } {
2595   post      .str_set_x:N      = \l_stex_symname_post_str
2596 }
2597
2598 \cs_new_protected:Nn \stex_symname_args:n {
2599   \str_clear:N \l_stex_symname_post_str
2600   \keys_set:nn { stex / symname } { #1 }
2601 }
2602
2603 \NewDocumentCommand \symname { 0{} m }{
2604   \stex_symname_args:n { #1 }
2605   \stex_get_symbol:n { #2 }
2606   \str_set:Nx \l_tmpa_str {
2607     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2608   }
2609   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2610
2611   \let\compemph_uri_prev:\compemph@uri
2612   \let\compemph@uri\symrefemph@uri
2613   \exp_args:NNx \use:nn
2614   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2615     \l_tmpa_str \l_stex_symname_post_str
2616   ] }
2617   \let\compemph@uri\compemph_uri_prev:
2618 }

```

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



## 20.3 Notation Components

2619  $\langle @@=\text{stex\_notationcomps} \rangle$

$\backslash\text{stex\_highlight\_term:nn}$

```

2620
2621 \str_new:N \l__stex_notationcomps_highlight_uri_str
2622 \cs_new_protected:Nn \stex_highlight_term:nn {
2623   \exp_args:Nnx
2624   \use:nn {
2625     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2626     #2
2627   } {
2628     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2629     { \l__stex_notationcomps_highlight_uri_str }
2630   }
2631 }
2632
2633 \cs_new_protected:Nn \stex_unhighlight_term:n {
2634 % \latexml_if:TF {
2635 %   #1
2636 % } {
2637 %   \scalatex_if:TF {
2638 %     #1
2639 %   } {
2640     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2641 %   }
2642 % }
2643 }
```

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

```

\comp
\compemph@uri 2644 \cs_new_protected:Npn \comp #1 {
\compemph 2645   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph 2646     \scalatex_if:TF {
\defemph@uri 2647       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph 2648     }{
\symrefemph@uri 2649       \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2650     }
2651   }
2652 }
2653
2654 \cs_new_protected:Npn \compemph@uri #1 #2 {
2655   \compemph{ #1 }
2656 }
2657
2658
2659 \cs_new_protected:Npn \compemph #1 {
2660   \textcolor{blue}{#1}
2661 }
2662
2663 \cs_new_protected:Npn \defemph@uri #1 #2 {
2664   \defemph{#1}
2665 }
```

```

2666
2667 \cs_new_protected:Npn \defemph #1 {
2668     \textbf{#1}
2669 }
2670
2671 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2672     \symrefemph{#1}
2673 }
2674
2675 \cs_new_protected:Npn \symrefemph #1 {
2676     \textbf{#1}
2677 }

```

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

### `\ellipses`

```

2678 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
2679 \bool_new:N \l_stex_inarray_bool
2680 \bool_set_false:N \l_stex_inarray_bool
2681 \NewDocumentCommand \parray { m m } {
2682     \begingroup
2683     \bool_set_true:N \l_stex_inarray_bool
2684     \begin{array}{#1}
2685         #2
2686     \end{array}
2687 \endgroup
2688 }
2689
2690 \NewDocumentCommand \prmatrix { m } {
2691     \begingroup
2692     \bool_set_true:N \l_stex_inarray_bool
2693     \begin{matrix}
2694         #1
2695     \end{matrix}
2696 \endgroup
2697 }
2698
2699 \def \parrayline #1 #2 {
2700     #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2701 }
2702
2703 \def \parraylineh #1 #2 {
2704     #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2705 }
2706
2707 \def \parraycell #1 {
2708     #1 \bool_if:NT \l_stex_inarray_bool {\&}
2709 }

```

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

```

2710 \endpackage

```

## Chapter 21

# STEX -Structural Features Implementation

```
2711 <*package>
2712
2713 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2714
2715 <@@=stex_features>
      Warnings and error messages
2716
symboldoc
2717 \NewDocumentEnvironment{symboldoc}{m}{
2718   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2719   \seq_clear:N \l_tmpb_seq
2720   \seq_map_inline:Nn \l_tmpa_seq {
2721     \stex_get_symbol:n { ##1 }
2722     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2723       \l_stex_get_symbol_uri_str
2724     }
2725   }
2726   \par
2727   \exp_args:Nnnx
2728   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2729 }{
2730   \end{stex_annotate_env}
2731 }

STEXdefinition
2732
2733 \NewDocumentCommand \definiendum { O{} m m } {
2734   \stex_get_symbol:n { #2 }
2735   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
2736   \scalatex_if:TF {
2737     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
```

```

2738 } {
2739   \exp_args:Nnx \defemph@uri { #3 } { \l_stex_get_symbol_uri_str }
2740 }
2741 }
2742 \stex_deactivate_macro:Nn \definiendum {definition~environments}
2743 \NewDocumentCommand \definame { 0{} m } {
2744   % TODO: root
2745   \stex_get_symbol:n { #2 }
2746   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
2747   \str_set:Nx \l_tmpa_str {
2748     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2749   }
2750   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2751   \scalatex_if:TF {
2752     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
2753       \l_tmpa_str
2754     }
2755   } {
2756     \defemph@uri {
2757       \l_tmpa_str
2758     } { \l_stex_get_symbol_uri_str }
2759   }
2760 }
2761 \stex_deactivate_macro:Nn \definame {definition~environments}
2762
2763 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2764   \stex_reactivate_macro:N \definiendum
2765   \stex_reactivate_macro:N \definame
2766   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2767   \seq_clear:N \l_tmpb_seq
2768   \seq_map_inline:Nn \l_tmpa_seq {
2769     \stex_get_symbol:n { ##1 }
2770     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2771       \l_stex_get_symbol_uri_str
2772     }
2773   }
2774   \exp_args:Nnnx
2775   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2776 }
2777
2778 \cs_new_protected:Nn \__stex_features_defi_end: {
2779   \end{stex_annotate_env}
2780 }
2781
2782 \NewDocumentEnvironment{STEXdefinition}{ m }{
2783   \__stex_features_defi_begin:n { #1 }
2784 }{
2785   \__stex_features_defi_end:
2786 }

```

\setSTEXdefinition

```

2787 \cs_new_protected:Npn \setSTEXdefinition #1 {
2788   \AddToHook{env/#1/before}[stex]{\__stex_features_defi_begin:n{}}
2789   \AddToHook{env/#1/after}[stex]{\__stex_features_defi_end:}

```

2790 }

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

structural@feature

```

2791
2792 \NewDocumentEnvironment{structural@feature}{ m m m }{
2793   \stex_if_in_module:F {
2794     \msg_set:nnn{stex}{error/nomodule}{
2795       Structural~Feature~has~to~occur~in~a~module:\\
2796       Feature~#2~of~type~#1\\
2797       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2798     }
2799     \msg_error:nn{stex}{error/nomodule}
2800   }
2801
2802   \str_set:Nx \l_stex_module_name_str {
2803     \prop_item:Nn \l_stex_current_module_prop
2804       { name } / #2 - feature
2805   }
2806
2807   \str_set:Nx \l_stex_module_ns_str {
2808     \prop_item:Nn \l_stex_current_module_prop
2809       { ns }
2810   }
2811
2812
2813   \str_clear:N \l_tmpa_str
2814   \seq_clear:N \l_tmpa_seq
2815   \tl_clear:N \l_tmpa_tl
2816   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2817     origname = #2,
2818     name     = \l_stex_module_name_str ,
2819     ns       = \l_stex_module_ns_str ,
2820     imports  = \exp_not:o { \l_tmpa_seq } ,
2821     constants = \exp_not:o { \l_tmpa_seq } ,
2822     content  = \exp_not:o { \l_tmpa_tl } ,
2823     file     = \exp_not:o { \g_stex_currentfile_seq } ,
2824     lang     = \l_stex_module_lang_str ,
2825     sig      = \l_tmpa_str ,
2826     meta     = \l_tmpa_str ,
2827     feature  = #1 ,
2828   }
2829
2830   \stex_if_smsmode:TF {
2831     \stex_smsmode_set_codes:
2832   } {
2833     \begin{stex_annotate_env}{ feature:#1 }{}
2834     \stex_annotate_invisible:nnn{header}{}{ #3 }
2835   }
2836 }{
2837   \str_set:Nx \l_tmpa_str {
2838     c_stex_feature_
2839     \prop_item:Nn \l_stex_current_module_prop { ns } ?

```

```

2840     \prop_item:Nn \l_stex_current_module_prop { name }
2841     _prop
2842 }
2843 \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2844 \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2845 \stex_if_smsmode:TF {
2846     \exp_args:Nx \stex_add_to_sms:n {
2847         \prop_gset_from_keyval:cn {
2848             c_stex_feature_
2849             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2850             \prop_item:Nn \l_stex_current_module_prop { name }
2851             _prop
2852         } {
2853             origname = #2,
2854             name      = \prop_item:cn { \l_tmpa_str } { name } ,
2855             ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2856             imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2857             constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2858             content    = \prop_item:cn { \l_tmpa_str } { content } ,
2859             file       = \prop_item:cn { \l_tmpa_str } { file } ,
2860             lang       = \prop_item:cn { \l_tmpa_str } { lang } ,
2861             sig        = \prop_item:cn { \l_tmpa_str } { sig } ,
2862             meta       = \prop_item:cn { \l_tmpa_str } { meta } ,
2863             feature    = \prop_item:cn { \l_tmpa_str } { feature }
2864         }
2865     }
2866 } {
2867     \end{stex_annotate_env}
2868 }
2869 }
2870

```

## structure

```

2871
2872 \prop_new:N \l_stex_all_structures_prop
2873
2874 \keys_define:nn { stex / features / structure } {
2875     name .str_set_x:N = \l__stex_features_structure_name_str ,
2876 }
2877
2878 \cs_new_protected:Nn \__stex_features_structure_args:n {
2879     \str_clear:N \l__stex_features_structure_name_str
2880     \keys_set:nn { stex / features / structure } { #1 }
2881 }
2882
2883 %\stex_new_feature:nnnn { structure } { 0{} m } {
2884 % \__stex_features_structure_args:n { ##1 }
2885 % \str_if_empty:NT \l__stex_features_structure_name_str {
2886 % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2887 % }
2888 %} {
2889 %
2890 %}
2891

```

```

2892 \NewDocumentEnvironment{structure}{0}{m}{
2893   \__stex_features_structure_args:n { #1 }
2894   \str_if_empty:NT \l__stex_features_structure_name_str {
2895     \str_set:Nx \l__stex_features_structure_name_str { #2 }
2896   }
2897   \exp_args:Nnnx
2898   \begin{structural@feature}{structure}
2899     { \l__stex_features_structure_name_str }{}
2900     \seq_clear:N \l_tmpa_seq
2901     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2902
2903   }{
2904     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2905     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2906     \str_set:Nx \l_tmpa_str {
2907       \prop_item:Nn \l_stex_current_module_prop { ns } ?
2908       \prop_item:Nn \l_stex_current_module_prop { name }
2909     }
2910     \seq_map_inline:Nn \l_tmpa_seq {
2911       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2912     }
2913     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2914     \exp_args:Nnx
2915     \AddToHookNext { env / structure / after }{
2916       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2917         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{}{}
2918       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2919     \STEXexport {
2920       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2921         {\prop_item:Nn \l_stex_current_module_prop { origname }}
2922       {\l_tmpa_str}
2923       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2924         {#2}{\l_tmpa_str}
2925       % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2926       %   \prop_item:Nn \l_stex_current_module_prop { origname },
2927       %   \l_tmpa_str
2928       % }
2929       % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2930       %   #2,\l_tmpa_str
2931       % }
2932       % \tl_set:cx { #2 } {
2933       %   \stex_invoke_structure:n { \l_tmpa_str }
2934     }
2935   }
2936
2937   \end{structural@feature}
2938   % \g_stex_last_feature_prop
2939 }

```

\instantiate

```

2940 \seq_new:N \l__stex_features_structure_field_seq
2941 \str_new:N \l__stex_features_structure_field_str
2942 \str_new:N \l__stex_features_structure_def_tl
2943 \prop_new:N \l__stex_features_structure_prop

```

```

2944 \NewDocumentCommand \instantiate { m O{} m }{
2945   \stex_smsmode_set_codes:
2946   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2947   \prop_set_eq:Nc \l__stex_features_structure_prop {
2948     c_stex_feature_\l_tmpa_str _prop
2949   }
2950   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2951   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2952     \seq_set_split:Nnn \l_tmpa_seq={}{ ##1 }
2953     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2954       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2955       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2956         {!} \l_tmpa_tl
2957       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2958         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2959         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2960         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2961       }{
2962         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2963         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2964         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2965           \l_tmpa_tl
2966         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2967           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2968           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2969         }{
2970           \tl_clear:N \l_tmpb_tl
2971         }
2972       }
2973     }{
2974       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2975       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2976         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2977         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2978         \tl_clear:N \l_tmpa_tl
2979       }{
2980         % TODO throw error
2981       }
2982     }
2983     % \l_tmpa_str: name
2984     % \l_tmpa_tl: definiens
2985     % \l_tmpb_tl: notation
2986     \tl_if_empty:NT \l__stex_features_structure_field_str {
2987       % TODO throw error
2988     }
2989     \str_clear:N \l_tmpb_str
2990
2991     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2992     \seq_map_inline:Nn \l_tmpa_seq {
2993       \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2994       \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2995       \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2996         \seq_map_break:n {
2997           \str_set:Nn \l_tmpb_str { ####1 }

```



```

2998     }
2999   }
3000 }
3001 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
3002   \l_tmpb_str
3003
3004 \tl_if_empty:NTF \l_tmpb_tl {
3005   \tl_if_empty:NF \l_tmpa_tl {
3006     \exp_args:Nx \use:n {
3007       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3008     }
3009   }
3010 }{
3011   \tl_if_empty:NTF \l_tmpa_tl {
3012     \exp_args:Nx \use:n {
3013       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3014     }
3015   }
3016 }{
3017   \exp_args:Nx \use:n {
3018     \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3019     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3020   }
3021 }
3022 }
3023 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3024 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3025 % #3/\l__stex_features_structure_field_str
3026 % \par
3027 % \expandafter\present\csname
3028 %   g_stex_symdecl_
3029 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3030 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3031 %   #3/\l__stex_features_structure_field_str
3032 %   _prop
3033 % \endcsname
3034 }
3035
3036 \tl_clear:N \l__stex_features_structure_def_tl
3037
3038 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3039 \seq_map_inline:Nn \l_tmpa_seq {
3040   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3041   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3042   \exp_args:Nx \use:n {
3043     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3044     }
3045   }
3046 }
3047
3048 \prop_if_exist:cF {
3049   g_stex_symdecl_
3050   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3051   \prop_item:Nn \l_stex_current_module_prop {name} ?

```

```

3052     #3/\l_tmpa_str
3053     _prop
3054   }{
3055     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
3056     \l_tmpb_str
3057     \exp_args:Nx \use:n {
3058       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3059     }
3060   }
3061 }
3062
3063 \symdecl*[type={\STEXsymbol{module-type}}{
3064   \_stex_term_math_oms:nnnn {
3065     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3066     \prop_item:Nn \l__stex_features_structure_prop {name}
3067     }{}{0}{}}
3068   ]]{#3}
3069
3070 % TODO: -> sms file
3071
3072 \tl_set:cx{ #3 }{
3073   \stex_invoke_structure:nnn {
3074     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3075     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3076   } {
3077     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3078     \prop_item:Nn \l__stex_features_structure_prop {name}
3079   }
3080 }
3081
3082 }

```

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

\stex\_invoke\_structure:nnn

```

3083 % #1: URI of the instance
3084 % #2: URI of the instantiated module
3085 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3086   \tl_if_empty:nTF{ #3 }{
3087     \prop_set_eq:Nc \l__stex_features_structure_prop {
3088       c_stex_feature_ #2 _prop
3089     }
3090     \tl_clear:N \l_tmpa_tl
3091     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3092     \seq_map_inline:Nn \l_tmpa_seq {
3093       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3094       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3095       \cs_if_exist:cT {
3096         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3097       }{
3098         \tl_if_empty:NF \l_tmpa_tl {
3099           \tl_put_right:Nn \l_tmpa_tl {,}
3100         }
3101         \tl_put_right:Nx \l_tmpa_tl {

```

```

3102         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3103     }
3104 }
3105 }
3106 \exp_args:No \mathstruct \l_tmpa_tl
3107 }{
3108     \stex_invoke_symbol:n{#1/#3}
3109 }
3110 }

```

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

```

3111 </package>

```

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

## Chapter 22

# STEX -Others Implementation

```
3112 <*package>
3113
3114 %%%%%%%%%% others.dtx %%%%%%%%%%
3115
3116 <@@=stex_others>
    Warnings and error messages
3117 % None

\MSC Math subject classifier

3118 \NewDocumentCommand \MSC {m} {
3119 % TODO
3120 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3121 \@ifpackageloaded{tikzinput}{
3122 \RequirePackage{stex-tikzinput}
3123 }{}
3124 </package>
```

## Chapter 23

# STEX -Metatheory Implementation

```
3125 <*package>
3126 <@@=stex_modules>
3127
3128 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3129
3130 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3131 \begingroup
3132 \stex_module_setup:nn{
3133   ns=\c_stex_metatheory_ns_str,
3134   meta=NONE
3135 }{Metatheory}
3136 \stex_reactivate_macro:N \symdecl
3137 \stex_reactivate_macro:N \notation
3138 \stex_reactivate_macro:N \symdef
3139 \ExplSyntaxOff
3140 \csname stex_suppress_html:n\endcsname{
3141   % is-a (a:A, a \in A, a is an A, etc.)
3142   \symdecl[args=ai]{isa}
3143   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3144   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3145   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3146
3147   % bind (\forall, \Pi, \lambda etc.)
3148   \symdecl[args=Bi]{bind}
3149   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3150   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3151   \notation[deffun]{bind}{\comp( #1 \comp)\;\to\;}{#1 \comp, #2}
3152
3153   % dummy variable
3154   \symdecl{dummyvar}
3155   \notation[underscore]{dummyvar}{\comp\_}
3156   \notation[dot]{dummyvar}{\comp\cdot}
3157   \notation[dash]{dummyvar}{\comp{\rm --}}
3158
3159   %fromto (function space, Hom-set, implication etc.)
```

```

3160 \symdecl[args=ai]{fromto}
3161 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3162 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3163
3164 % mapto (lambda etc.)
3165 %\symdecl[args=Bi]{mapto}
3166 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3167 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3168 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3169
3170 % function/operator application
3171 \symdecl[args=ia]{apply}
3172 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3173 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3174
3175 % ‘‘type’’ of all collections (sets, classes, types, kinds)
3176 \symdecl{collection}
3177 \notation[U]{collection}{\comp{\mathcal{U}}}
3178 \notation[set]{collection}{\comp{\textsf{Set}}}
3179
3180 % sequences
3181 \symdecl[args=1]{seqtype}
3182 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3183
3184 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3185 \notation[ui]{sequence-index}{#1^{#2}}
3186
3187 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses},#1_{#3}}
3188 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses},#1^{#3}}
3189 % ^ superceded by \aseqfromto and \livar/\uivar
3190
3191 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3192 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
3193 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\comp,}#2\comp{\,\ellipses}
3194
3195 % letin (‘‘let’’, local definitions, variable substitution)
3196 \symdecl[args=bii]{letin}
3197 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3198 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3199 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3200
3201 % structures
3202 \symdecl*[args=1]{module-type}
3203 \notation{module-type}{\mathtt{MOD} #1}
3204 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3205 \notation[angle,prec=nobrackets]{mathstruct}{\comp\langle #1 \comp\rangle}{#1 \comp, #2}
3206
3207 }
3208 \ExplSyntaxOn
3209 \stex_add_to_current_module:n{
3210   \let\nappa\apply
3211   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3212   \def\livar{\csname sequence-index\endcsname[li]}
3213   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3214     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3215     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3216   }
3217   \__stex_modules_end_module:
3218   \endgroup
3219 \end{package}

```

## Chapter 24

# Tikzinput Implementation

```
3220 <*package>
3221
3222 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3223
3224 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3225 \RequirePackage{l3keys2e}
3226
3227 \keys_define:nn { tikzinput } {
3228   image .bool_set:N = \c_tikzinput_image_bool,
3229   image .default:n = false ,
3230 }
3231
3232 \ProcessKeysOptions { tikzinput }
3233
3234 \bool_if:NTF \c_tikzinput_image_bool {
3235   \RequirePackage{graphicx}
3236
3237   \providecommand\usetikzlibrary[]{}
3238   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
3239 }{
3240   \RequirePackage{tikz}
3241   \RequirePackage{standalone}
3242
3243   \newcommand \tikzinput [2] [] {
3244     \setkeys{Gin}{#1}
3245     \ifx \Gin@ewidth \Gin@exclamation
3246       \ifx \Gin@eheight \Gin@exclamation
3247         \input { #2 }
3248       \else
3249         \resizebox{!}{ \Gin@eheight }{
3250           \input { #2 }
3251         }
3252       \fi
3253     \else
3254       \ifx \Gin@eheight \Gin@exclamation
3255         \resizebox{ \Gin@ewidth }{!}{
3256           \input { #2 }
3257         }
3258       \fi
3259     \fi
3260   }
3261 }
```



```

3258     \else
3259         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3260             \input { #2 }
3261         }
3262     \fi
3263 \fi
3264 }
3265 }
3266
3267 \newcommand \ctikzinput [2] [] {
3268     \begin{center}
3269         \tikzinput [1] {#2}
3270     \end{center}
3271 }
3272
3273 \@ifpackageloaded{stex}{
3274     \RequirePackage{stex-tikzinput}
3275 }{}
3276
3277 </package>
3278 <*stex>
3279 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3280 \RequirePackage{stex}
3281 \RequirePackage{tikzinput}
3282
3283 \newcommand\mhtikzinput[2] [] {%
3284     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3285     \stex_in_repository:nn\Gin@mhrepos{
3286         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
3287     }
3288 }
3289 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
3290 </stex>

```

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

## 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`

```
3291 \*cls)
3292 \RequirePackage{etoolbox}
3293 \RequirePackage{kvoptions}
3294 \SetupKeyvalOptions{family=omdoc@cls,prefix=omdoc@cls@}
3295 \DeclareStringOption[article]{class}
3296 \AddToKeyvalOption*{class}{\PassOptionsToPackage{class=\omdoc@cls@class}{omdoc}}

the following options are deprecated.

3297 \DeclareVoidOption{report}{\def\omdoc@cls@class{report}}%
3298 \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}}
3299 \DeclareVoidOption{book}{\def\omdoc@cls@class{book}}%
3300 \ClassWarning{omdoc}{the option 'part' is deprecated, use 'class=book', instead}}
3301 \DeclareVoidOption{bookpart}{\def\omdoc@cls@class{book}}%
3302 \PassOptionsToPackage{topsect=chapter}{omdoc}%
3303 \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter',
3304 \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`.

```
3305 \def\@omdoc@cls@doopt{}
3306 \DeclareDefaultOption{%
```

```

3307 \ifx\@omdoc@cls@dopt\@empty%
3308 \xdef\@omdoc@cls@dopt{\CurrentOption}%
3309 \else\xappto\@omdoc@cls@dopt{,\CurrentOption}%
3310 \fi}%
3311 \PassOptionsToPackage{\CurrentOption}{omdoc}
3312 \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.

```

3313 \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

```

3314 \ifomdoc@cls@minimal\else%
3315 \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>

```

3316 \srefaddidkey{document}
3317 \newcommand\documentkeys[1]{\metasetkeys{document}{#1}}
3318 \let\orig@document=\document
3319 \renewcommand{\document}[1] [] {\metasetkeys{document}{#1}\orig@document}

```

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

```

3320 \fi% \ifomdoc@cls@minimal
3321 \</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).

```

3322 <*package>
3323 \RequirePackage{kvoptions}
3324 \SetupKeyvalOptions{family=omdoc@sty,prefix=omdoc@sty@}
3325 \DeclareStringOption[article]{class}
3326 \DeclareBoolOption{showignores}
3327 \DeclareStringOption[section]{topsect}
3328 \newcount\section@level
3329 \DeclareDefaultOption{\PassOptionsToPackage{\CurrentOption}{sref}}
3330 \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.

```

3331 \RequirePackage{stex-metakeys}
3332 %\RequirePackage{sref}
3333 \RequirePackage{xspace}
3334 \RequirePackage{comment}
3335 %\RequirePackage{pathsuris}
3336 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}
3337
3338 \def\srefaddidkey#1{\addmetakey{#1}{id}}
3339

```

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

```

3340 \ExplSyntaxOn
3341 \@ifpackageloaded{babel}{
3342   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3343   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3344     \input{omdoc-ngerman.ldf}
3345   }
3346 }{}
3347 \ExplSyntaxOff
3348 %\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.

```

3349 \section@level=2
3350 \ifdefstring{\omdoc@sty@class}{book}{\section@level=0}{}
3351 \ifdefstring{\omdoc@sty@class}{report}{\section@level=0}{}
3352 \ifdefstring{\omdoc@sty@topsect}{part}{\section@level=0}{}
3353 \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>

```

3354 \def\current@section@level{document}%
3355 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
3356 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipomgroup`

```

3357 \newcommand\skipomgroup{%
3358   \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.

```

3359 \or\stepcounter{chapter}%
3360 \or\stepcounter{section}%
3361 \or\stepcounter{subsection}%
3362 \or\stepcounter{subsubsection}%
3363 \or\stepcounter{paragraph}%
3364 \or\stepcounter{subparagraph}%
3365 \fi}% \ifcase

```

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

**blindomgroup**

```

3366 \newcommand\at@begin@blindomgroup[1]{%
3367 \newenvironment{blindomgroup}
3368 {\advance\section@level by 1\at@begin@blindomgroup\section@level}
3369 {\advance\section@level by -1}

```

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

```

3370 \newcommand\omgroup@nonum[2]{%
3371 \ifx\hyper@anchor\@undefined\else\phantomsection\fi%
3372 \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.

```

3373 \newcommand\omgroup@num[2]{%
3374 \ifx\omgroup@short\@empty% no short title
3375 \@nameuse{#1}{#2}%
3376 \else% we have a short title
3377 \@ifundefined{rdfmata@sectioning}%
3378 {\@nameuse{#1}[\omgroup@short]{#2}}%
3379 {\@nameuse{rdfmata@#1@old}[\omgroup@short]{#2}}%
3380 \fi%
3381 %\sref@label@id@arg{\@nameuse{the#1}}\omgroup@id
3382 }

```

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

**omgroup**

```

3383 \def\@true{true}
3384 \def\@false{false}
3385 \srefaddidkey{omgroup}
3386 \addmetakey{omgroup}{date}
3387 \addmetakey{omgroup}{creators}
3388 \addmetakey{omgroup}{contributors}
3389 \addmetakey{omgroup}{srccite}
3390 \addmetakey{omgroup}{type}
3391 \addmetakey*{omgroup}{short}
3392 \addmetakey*{omgroup}{display}
3393 \addmetakey*{omgroup}{intro}% ignored
3394 \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.

```
3395 \newif\if@mainmatter\@mainmattertrue
3396 \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.

```
3397 \addmetakey{omdoc@sect}{name}
3398 \addmetakey[false]{omdoc@sect}{clear}[true]
3399 \addmetakey{omdoc@sect}{ref}
3400 \addmetakey[false]{omdoc@sect}{num}[true]
3401 \newcommand\omdoc@sectioning[3] [] {\metasetkeys{omdoc@sect}{#1}%
3402 \ifx\omdoc@sect@clear\@true\cleardoublepage\fi%
3403 \if@mainmatter% numbering not overridden by frontmatter, etc.
3404 \ifx\omdoc@sect@num\@true\omgroup@num{#2}{#3}\else\omgroup@nonum{#2}{#3}\fi%
3405 \def\current@section@level{\omdoc@sect@name}%
3406 \else\omgroup@nonum{#2}{#3}%
3407 \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.

```
3408 \newcommand\omgroup@redefine@addtocontents[1] {%
3409 %\edef\@import{#1}%
3410 %\@for\@I:=\@import\do{%
3411 %\edef\@path{\csname module@\@I @path\endcsname}%
3412 %\@ifundefined{tf@toc}\relax%
3413 % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
3414 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
3415 %\def\addcontentsline##1##2##3{%
3416 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
3417 %\else% hyperref.sty not loaded
3418 %\def\addcontentsline##1##2##3{%
3419 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
3420 %\fi
3421 }% 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.

```
3422 \newcount\omgroup@level
3423 \newenvironment{omgroup}[2] [] % keys, title
3424 {\metasetkeys{omgroup}{#1}%\sref@target%
3425 \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.

```
3426 \ifx\omgroup@loadmodules\@true%
3427 \omgroup@redefine@addtocontents{\@ifundefined{module@id}\used@modules%
3428 {\@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`.

```
3429 \advance\section@level by 1\relax%
3430 \ifcase\section@level%
```

```

3431 \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}%
3432 \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}%
3433 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}%
3434 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}%
3435 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}%
3436 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}%
3437 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragraph}
3438 \fi% \ifcase
3439 \at@begin@omgroup[#1]\section@level{#2}
3440 \csname stex_ref_new_doc_target:n\endcsname\omgroup@id%
3441 }% for customization
3442 {\advance\section@level by -1\advance\omgroup@level by -1}

```

and finally, we localize the sections

```

3443 \newcommand\omdoc@part@kw{Part}
3444 \newcommand\omdoc@chapter@kw{Chapter}
3445 \newcommand\omdoc@section@kw{Section}
3446 \newcommand\omdoc@subsection@kw{Subsection}
3447 \newcommand\omdoc@subsubsection@kw{Subsubsection}
3448 \newcommand\omdoc@paragraph@kw{paragraph}
3449 \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`

```

3450 \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).

```

3451 \ifcsdef{frontmatter}% to redefine if necessary
3452   {\cslet{orig@frontmatter}{\frontmatter}\cslet{frontmatter}{\relax}}
3453   {\cslet{orig@frontmatter}{\clearpage\@mainmatterfalse\pagenumbering{roman}}}
3454 \ifcsdef{backmatter}% to redefine if necessary
3455   {\cslet{orig@backmatter}{\backmatter}\cslet{backmatter}{\relax}}
3456   {\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.

```

3457 \newenvironment{frontmatter}
3458 {\orig@frontmatter}
3459 {\ifcsdef{mainmatter}{\mainmatter}{\clearpage\@mainmattertrue\pagenumbering{arabic}}}

```

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

```

3460 \newenvironment{backmatter}
3461 {\orig@backmatter}
3462 {\ifcsdef{mainmatter}{\mainmatter}{\clearpage\@mainmattertrue\pagenumbering{arabic}}}

```

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

```
3463 \@mainmattertrue\pagenumbering{arabic}
```

## 25.8 Ignoring Inputs

ignore

```
3464 \ifomdoc@sty@showignores
3465 \addmetakey{ignore}{type}
3466 \addmetakey{ignore}{comment}
3467 \newenvironment{ignore}[1] []
3468 {\metasetkeys{ignore}{#1}\textless\ignore@type\textgreater\bgroup\itshape}
3469 {\egroup\textless\ignore@type\textgreater}
3470 \renewenvironment{ignore}{}{}\else\excludcomment{ignore}\fi
```

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

```
3471 \newcommand\afterprematurestop{}
3472 \def\prematurestop@endomgroup{\ifnum\omgroup@level=0\else%
3473 \end{omgroup}\advance\omgroup@level by -1\prematurestop@endomgroup\fi}
3474 \providecommand\prematurestop{%
3475 \message{Stopping sTeX processing prematurely}
3476 \prematurestop@endomgroup\afterprematurestop
3477 \end{document}}
```

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

## 25.9 Structure Sharing

EdN:10

10

```
3478 \providecommand{\lxDocumentID}[1]{}%
3479 \def\LXMID#1#2{\expandafter\gdef\csname xmarg#1\endcsname{#2}\csname xmarg#1\endcsname}
3480 \def\LXMRef#1{\csname xmarg#1\endcsname}
```

`\STRlabel` The main macro, it it 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.

```
3481 \long\def\STRlabel#1#2{\STRlabeldef{#1}{#2}{#2}}
```

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

EdN:11

`\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>

```
3482 \newcommand\STRcopy[2] []{\expandafter\ifx\csname STR@#2\endcsname\relax
3483 \message{STR warning: reference #2 undefined!}
3484 \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

```

3485 \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`

```

3486 \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

```

3487 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

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

```

`\useSGvar` use a global variable

```

3488 \newrobustcmd\useSGvar[1]{%
3489   \@ifundefined{sTeX@Gvar@#1}
3490   {\PackageError{omdoc}
3491     {The sTeX Global variable #1 is undefined}
3492     {set it with \protect\setSGvar}}
3493   \@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.

```

3494 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
3495   \@ifundefined{sTeX@Gvar@#1}
3496   {\PackageError{omdoc}
3497     {The sTeX Global variable #1 is undefined}
3498     {set it with \protect\setSGvar}}
3499   {\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`

```

3500 \def\black#1{\textcolor{black}{#1}}
3501 \def\gray#1{\textcolor{gray}{#1}}
3502 \def\blue#1{\textcolor{blue}{#1}}
3503 \def\red#1{\textcolor{red}{#1}}
3504 \def\green#1{\textcolor{green}{#1}}
3505 \def\cyan#1{\textcolor{cyan}{#1}}
3506 \def\magenta#1{\textcolor{magenta}{#1}}
3507 \def\brown#1{\textcolor{brown}{#1}}
3508 \def\yellow#1{\textcolor{yellow}{#1}}
3509 \def\orange#1{\textcolor{orange}{#1}}
3510 \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.

```
3511 \*cls
3512 \RequirePackage{kvoptions}
3513 \RequirePackage{stex-metakeys}
3514 \RequirePackage{etoolbox}
3515 \SetupKeyvalOptions{family=mks@cls,prefix=mks@cls@}
3516 \DeclareStringOption[article]{class}
3517 \AddToKeyvalOption*{class}{\PassOptionsToClass{class=\mks@cls@class}{omdoc}
3518 \ifdefstring{\mks@cls@class}{book}{\PassOptionsToPackage{defaulttopsect=part}{mikoslides}}
3519 \ifdefstring{\mks@cls@class}{report}{\PassOptionsToPackage{defaulttopsect=part}{mikoslides}}
3520 \DeclareBoolOption{notes}
3521 \DeclareComplementaryOption{slides}{notes}
3522 \DeclareDefaultOption{%
3523 \PassOptionsToClass{\CurrentOption}{omdoc}
3524 \PassOptionsToClass{\CurrentOption}{beamer}
3525 \PassOptionsToPackage{\CurrentOption}{mikoslides}}
3526 \ProcessKeyvalOptions{mks@cls}
3527 \*cls
```

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

```
3528 \*package
3529 %\RequirePackage{stex-base}
3530 \RequirePackage{kvoptions}
3531 \RequirePackage{stex-metakeys}
3532 \SetupKeyvalOptions{family=mks@sty,prefix=mks@sty@}
3533 \DeclareStringOption{topsect}
3534 \DeclareStringOption{defaulttopsect}
3535 \newif\ifnotes\notesttrue
3536 \DeclareBoolOption{notes}
3537 \AddToKeyvalOption*{notes}{\notesttrue}%\PassOptionsToPackage{notes}{statements}}
3538 \DeclareComplementaryOption{slides}{notes}
3539 \AddToKeyvalOption*{slides}{\notestfalse}%\PassOptionsToPackage{nontheorem}{statements}}
```

```

3540 \DeclareBoolOption{sectocframes}
3541 \DeclareBoolOption{frameimages}
3542 \DeclareBoolOption{fiboxed}
3543 \DeclareBoolOption{noproblems}
3544 %\DeclareDefaultOption{
3545   %\PassOptionsToPackage{\CurrentOption}{stex}
3546   %\PassOptionsToPackage{\CurrentOption}{smglom}
3547   %\PassOptionsToPackage{\CurrentOption}{tikzinput}}
3548 \ProcessKeyvalOptions{mks@sty}

```

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

```

3549 \ifx\mks@sty@topsect\@empty\edef\@@topsect{\mks@sty@defaulttopsect}
3550 \else\edef\@@topsect{\mks@sty@topsect}\fi
3551 \</package>

```

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

```

3552 \<cls>
3553 \ifmks@cls@notes
3554   \LoadClass{omdoc}
3555 \else
3556   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
3557   \newcounter{Item}
3558   \newcounter{paragraph}
3559   \newcounter{subparagraph}
3560   \newcounter{Hfootnote}
3561 \fi

```

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

```

3562 \RequirePackage{mikoslides}
3563 \</cls>

```

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

```

3564 \<package>
3565 %\RequirePackage{stex}
3566 \RequirePackage{stex-compatibility}
3567 \ifmks@sty@notes
3568   \RequirePackage{a4wide}
3569   \RequirePackage{marginnote}
3570   \RequirePackage[dvipsnames,svgnames]{xcolor}
3571   \RequirePackage{mdframed}
3572   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
3573 \fi
3574 \RequirePackage{etoolbox}
3575 \RequirePackage{amssymb}
3576 \RequirePackage{amsmath}
3577 \RequirePackage{comment}
3578 \RequirePackage{textcomp}
3579 \RequirePackage{url}

```

```

3580 \RequirePackage{graphicx}
3581 \RequirePackage{pgf}
3582 %\RequirePackage{omtext}
3583 \ifmks@sty@notes
3584 \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
3585 \fi

```

finally, we require the `metakeys` package from  $\text{\LaTeX}$ , so that we can use the `\addmetakey` mechanism.

```

3586 %\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>

```

3587 \ifmks@sty@notes
3588 \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]{}
3589 \fi

```

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

```

3590 \newcounter{slide}
3591 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
3592 \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.

```

3593 \ifmks@sty@notes%
3594 \renewenvironment{note}{\ignorespaces}{}%
3595 \else%
3596 \excludecomment{note}%
3597 \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.

```

3598 \ifmks@sty@notes
3599 \newlength{\slideframewidth}
3600 \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

3601 \addmetakey{frame}{label}
3602 \addmetakey[yes]{frame}{allowframebreaks}
3603 \addmetakey{frame}{allowdisplaybreaks}
3604 \addmetakey[yes]{frame}{fragile}
3605 \addmetakey[yes]{frame}{shrink}
3606 \addmetakey[yes]{frame}{squeeze}
3607 \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.

```

3608 \renewenvironment{frame}[1][]{%
3609 \metasetkeys{frame}{#1}%
3610 \sffamily%
3611 \stepcounter{slide}%
3612 \def\@currentlabel{\theslide}%
3613 \ifx\frame@label\@empty\else\label{\frame@label}\fi%

```

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

```

3614 \def\itemize@level{outer}%
3615 \def\itemize@outer{outer}%
3616 \def\itemize@inner{inner}%
3617 \renewcommand\newpage{\addtocounter{framenumber}{1}}%
3618 \renewcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}%
3619 \renewenvironment{itemize}{%
3620 \ifx\itemize@level\itemize@outer%
3621 \def\itemize@label{$\rhd$}%
3622 \fi%
3623 \ifx\itemize@level\itemize@inner%
3624 \def\itemize@label{$\scriptstyle\rhd$}%
3625 \fi%
3626 \begin{list}%
3627 {\itemize@label}%
3628 {\setlength{\labelsep}{.3em}%
3629 \setlength{\labelwidth}{.5em}%
3630 \setlength{\leftmargin}{1.5em}%
3631 }%
3632 \edef\itemize@level{\itemize@inner}%
3633 }{%
3634 \end{list}%
3635 }%

```

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

```

3636 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=
3637 ]{%
3638 \medskip\miko@slidelabel\end{mdframed}%
3639 }%

```

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

`\frametitle`

```

3640 \renewcommand{\frametitle}[1]{\Large\bf\sffamily\color{blue}{#1}\medskip}%
3641 \fi %ifmks@sty@notes

```

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

EdN:13

`\pause` 13

```

3642 \ifmks@sty@notes\newcommand\pause{}\fi

```

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

`nomtext`

```

3643 \ifmks@sty@notes\newenvironment{nomtext}[1]{}{\begin{omtext}[#1]}{\end{omtext}}%
3644 \else\excludecomment{nomtext}\fi%

```

---

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

```

nomgroup
3645 \ifmks@sty@notes\newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}%
3646 \else\excludecomment{nomgroup}\fi%

ndefinition
3647 \ifmks@sty@notes\newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}%
3648 \else\excludecomment{ndefinition}\fi%

nassertion
3649 \ifmks@sty@notes\newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}%
3650 \else\excludecomment{nassertion}\fi%

nsproof
3651 \ifmks@sty@notes\newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}%
3652 \else\excludecomment{nsproof}\fi%

nexample
3653 \ifmks@sty@notes\newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}%
3654 \else\excludecomment{nexample}\fi%

\inputref*skip We customize the hooks for in \inputref.
3655 \def\inputref@preskip{\smallskip}
3656 \def\inputref@postskip{\medskip}

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

\inputref*
3657 \let\orig@inputref\inputref
3658 \def\inputref{\@ifstar\ninputref\orig@inputref}
3659 \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>}.
3660 \newlength{\slidelogoheight}
3661 \ifmks@sty@notes%
3662 \setlength{\slidelogoheight}{.4cm}%
3663 \else%
3664 \setlength{\slidelogoheight}{1cm}%
3665 \fi%
3666 \newsavebox{\slidelogo}%
3667 \sbox{\slidelogo}{\TeX}%
3668 \newrobustcmd{\setslidelogo}[1]{%
3669 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}%
3670 }%

(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.

```
3671 \def\source{Michael Kohlhase}% customize locally
3672 \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.

```
3673 \def\copyrightnotice{\footnotesize\copyright:\hspace{.3ex}{\source}}%
3674 \newsavebox{\cclogo}%
3675 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}%
3676 \newif\ifcchref\cchreffalse%
3677 \AtBeginDocument{%
3678   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
3679 }%
3680 \def\licensing{%
3681   \ifcchref%
3682     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}%
3683   \else%
3684     {\usebox{\cclogo}}%
3685   \fi%
3686 }%
3687 \newrobustcmd{\setlicensing}[2][{}]{%
3688   \def@url{#1}%
3689   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}%
3690   \ifx@url@empty%
3691     \def\licensing{{\usebox{\cclogo}}}%
3692   \else%
3693     \def\licensing{%
3694       \ifcchref%
3695         \href{#1}{\usebox{\cclogo}}%
3696       \else%
3697         {\usebox{\cclogo}}%
3698       \fi%
3699     }%
3700   \fi%
3701 }%
```

(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>

```
3702 \newrobustcmd\miko@slidelabel{%
3703   \vbox to \slidelogoheight{%
3704     \vss\hbox to \slidewidth%
3705     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}%
3706   }%
3707 }%
3708 % \subsection{Frame Images}\label{sec:impl:frameimage}
3709 %
```

---

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

```

3710 % \begin{macro}{\frameimage}
3711 % We have to make sure that the width is overwritten, for that we check the
3712 % |\Gin@ewidth| macro from the |graphicx| package. We also add the |label| key.
3713 % \begin{macrocode}
3714 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
3715 \newrobustcmd\frameimage[2] [] {%
3716   \stepcounter{slide}%
3717   \ifmks@sty@frameimages%
3718     \def\Gin@ewidth{}\setkeys{Gin}{#1}%
3719     \ifmks@sty@notes\else\vfill\fi%
3720     \begin{center}
3721       \ifmks@sty@fiboxed%
3722         \fbox{\ifx\Gin@ewidth\@empty\includegraphics[width=\slidewidth,#1]{#2}\else\mygraphics[width=\slidewidth,#1]{#2}\else\mygraphics[width=\slidewidth,#1]{#2}}
3723       \else
3724         \ifx\Gin@ewidth\@empty\includegraphics[width=\slidewidth,#1]{#2}\else\mygraphics[width=\slidewidth,#1]{#2}
3725       \fi% \ifmks@sty@fiboxed
3726     \end{center}
3727     \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
3728     \ifmks@sty@notes\else\vfill\fi%
3729   \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`.

```

3730 \def\Gin@mhrepos{}
3731 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
3732 \newcommand\mhframeimage[2] [] {%
3733   \setkeys{Gin}{#1}%
3734   \edef\mh@crepos{\mh@currentrepos}%
3735   \ifx\Gin@mhrepos\@empty%
3736     \edef\temp@path{\MathHub{\mh@currentrepos/source/#2}}%
3737   \else%
3738     \edef\temp@path{\MathHub{\Gin@mhrepos/source/#2}}%
3739   \fi%
3740   \if@iswindows@ \path@to@windows\temp@path\fi%
3741   \frameimage[#1]{\temp@path}%
3742 }%

```

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

## 26.4 Colors and Highlighting

We first specify sans serif fonts as the default.

```

3743 \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

```

3744 \AtBeginDocument{%
3745   \definecolor{green}{rgb}{0,.5,0}%
3746   \definecolor{purple}{cmyk}{.3,1,0,.17}%
3747 }%

```



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`.

```

3748 % \def\STpresent#1{\textcolor{blue}{#1}}
3749 \def\defemph#1{\textcolor{magenta}{#1}}
3750 \def\symrefemph#1{\textcolor{cyan}{#1}}
3751 \def\compemph#1{\textcolor{magenta}{#1}}
3752 \def\titleemph#1{\textcolor{blue}{#1}}
3753 \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.

```

3754 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
3755 \def\smalltextwarning{%
3756   \pgfuseimage{miko@small@dbend}%
3757   \xspace%
3758 }%
3759 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
3760 \newrobustcmd\textwarning{%
3761   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}%
3762   \xspace%
3763 }%
3764 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}%
3765 \newrobustcmd\bigtextwarning{%
3766   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}%
3767   \xspace%
3768 }%

```

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

```

3769 \newrobustcmd\putgraphicsat[3]{%
3770   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}%
3771 }%
3772 \newrobustcmd\putat[2]{%
3773   \begin{picture}(0,0)\put(#1){#2}\end{picture}%
3774 }%

```

## 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`.

```

3775 \ifmks@sty@sectocframes%
3776 \ifdefstring\@@topsect{part}{%
3777   \newcounter{chapter}\counterwithin*{section}{chapter}}
3778 {\ifdefstring\@@topsect{chapter}{\newcounter{chapter}\counterwithin*{section}{chapter}}{}}
3779 \fi% ifsectocframes

```

Now that we have defined the counters, we can load the  $\text{\TeX}$ -specific packages (in particular `statements` that needs these counters).

```

3780 \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`

```

3781 \section@level=2
3782 \def\part@prefix{}
3783 \ifdefstring{\@@topsect}{part}
3784 {\section@level=0%
3785   \def\thesection{\arabic{chapter}.\arabic{section}}}%
3786   \def\part@prefix{\arabic{chapter}.}}{}
3787 \ifdefstring{\@@topsect}{chapter}
3788 {\section@level=1%
3789   \def\thesection{\arabic{chapter}.\arabic{section}}}%
3790   \def\part@prefix{\arabic{chapter}.}}{}
3791 \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`

```

3792 \renewenvironment{omgroup}[2][]{%
3793   \metasetkeys{omgroup}{#1}%
3794   \advance\section@level by 1%
3795   \advance\omgroup@level by 1%
3796   \ifmks@sty@sectocframes%
3797     \stepcounter{slide}
3798     \begin{frame}[noframenumbering]%
3799     \vfill\Large\centering%
3800     \red{%
3801       \ifcase\section@level\or
3802       \stepcounter{part}
3803       \def\@@label{\omdoc@part@kw~\Roman{part}}
3804       \def\currentsectionlevel{\omdoc@part@kw}
3805       \or%
3806       \stepcounter{chapter}
3807       \def\@@label{\omdoc@chapter@kw~\arabic{chapter}}
3808       \def\currentsectionlevel{\omdoc@chapter@kw}
3809       \or
3810       \stepcounter{section}
3811       \def\@@label{\part@prefix\arabic{section}}
3812       \def\currentsectionlevel{\omdoc@section@kw}
3813       \or
3814       \stepcounter{subsection}
3815       \def\@@label{\part@prefix\arabic{section}.\arabic{subsection}}
3816       \def\currentsectionlevel{\omdoc@subsection@kw}
3817       \or
3818       \stepcounter{subsubsection}
3819       \def\@@label{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
3820       \def\currentsectionlevel{\omdoc@subsubsection@kw}
3821       \or
3822       \stepcounter{mparagraph}
3823       \def\@@label{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{subsubsection}.\arabic{mparagraph}}
3824       \def\currentsectionlevel{\omdoc@paragraph@kw}

```

```

3825 \fi% end ifcase
3826 \@@label\sref@label@id\@@label
3827 \quad #2%
3828 }%
3829 \vfill%
3830 \end{frame}%
3831 \fi %ifmks@sty@sectocframes
3832 \csname stex_ref_new_doc_target:n\endcsname\omgroup@id%
3833 }
3834 {\advance\section@level by -1}%
3835 \fi% ifmks@sty@notes

```

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

```

3836 \def\inserttheorembodyfont{\normalfont}
3837 \ifmks@sty@notes\else% only in slides
3838 \defbeamertemplate{theorem begin}{miko}
3839 {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
3840 \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
3841 \inserttheorempunctuation\inserttheorembodyfont\xspace}
3842 \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

3843 \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.

```

3844 \expandafter\def\csname Parent2\endcsname{}
3845 \fi% ifmks@sty@notes
3846 \ifmks@sty@notes%
3847 \renewenvironment{columns}[1][]{%
3848 \par\noindent%
3849 \begin{minipage}%
3850 \slidewidth\centering\leavevmode%
3851 }{%
3852 \end{minipage}\par\noindent%
3853 }%
3854 \newsavebox\columnbox%
3855 \renewenvironment<>{column}[2][]{%
3856 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
3857 }{%
3858 \end{minipage}\end{lrbox}\usebox\columnbox%
3859 }%
3860 \fi% ifmks@sty@notes
3861 \ifmks@sty@nopproblems%
3862 \newenvironment{problems}{}{}%
3863 \else%
3864 \excludcomment{problems}%
3865 \fi%

```

## 26.6 Excursions

`\excursion` The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is

defined before formatting the file in the argument.

```

3866 \gdef\printexcursions{}
3867 \newcommand\excursionref[2]{% label, text
3868 \ifnotes\begin{omtext}[title=Excursion]#2 \sref[fallback=the appendix]{#1}.\end{omtext}\fi}
3869 \newcommand\activate@excursion[2][]{\gappto\printexcursions{\inputref[#1]{#2}}}
3870 \newcommand\excursion[4][]{% repos, label, path, text
3871 \ifnotes\activate@excursion[#1]{#3}\excursionref{#2}{#4}\fi}%

```

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

**\excursiongroup**

```

3872 \srefaddidkey{excursiongroup}%
3873 \addmetakey{excursiongroup}{intro}%
3874 \addmetakey{excursiongroup}{mhrepos}%
3875 \newcommand\excursiongroup[1][]{\metasetkeys{excursiongroup}{#1}%
3876 \ifdefempty\printexcursions{}% only if there are excursions
3877 {\begin{omgroup}[#1]{Excursions}%
3878 \ifdefempty\excursiongroup@intro-{\inputref[\excursiongroup@mhrepos]{\excursiongroup@
3879 \printexcursions%
3880 \end{omgroup}}}
3881 \end{package}

```

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

## 26.7 Miscellaneous