

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

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

2022-01-24

## **Abstract**

TODO

---

\*Version 3.0 (last revised 2022-01-24)

# 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
9.1.1	Structures . . . . .	30
<b>10</b>	<b>sTeX-Statements</b>	<b>31</b>
10.1	Macros and Environments . . . . .	31

<b>11</b>	<b>STeX-Proofs: Structural Markup for Proofs</b>	<b>32</b>
11.1	Introduction	34
11.2	The User Interface	35
11.2.1	Package Options	35
11.2.2	Proofs and Proof steps	35
11.2.3	Justifications	35
11.2.4	Proof Structure	36
11.2.5	Proof End Markers	37
11.2.6	Configuration of the Presentation	37
11.3	Limitations	37
<b>12</b>	<b>STeX-Metatheory</b>	<b>39</b>
12.1	Symbols	39
<b>III</b>	<b>Extensions</b>	<b>40</b>
<b>13</b>	<b>Tikzinput</b>	<b>41</b>
13.1	Macros and Environments	41
<b>14</b>	<b>document-structure.sty: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>42</b>
14.1	Introduction	42
14.2	The User Interface	43
14.2.1	Package and Class Options	43
14.2.2	Document Structure	43
14.2.3	Ignoring Inputs	44
14.2.4	Structure Sharing	45
14.2.5	Global Variables	45
14.2.6	Colors	46
14.3	Limitations	46
<b>15</b>	<b>Slides and Course Notes</b>	<b>47</b>
15.1	Introduction	47
15.2	The User Interface	47
15.2.1	Package Options	47
15.2.2	Notes and Slides	48
15.2.3	Header and Footer Lines of the Slides	49
15.2.4	Frame Images	49
15.2.5	Colors and Highlighting	50
15.2.6	Front Matter, Titles, etc.	50
15.2.7	Excursions	50
15.2.8	Miscellaneous	50
15.3	Limitations	50

<b>16</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>51</b>
16.1	Introduction . . . . .	51
16.2	The User Interface . . . . .	51
16.2.1	Package Options . . . . .	51
16.2.2	Problems and Solutions . . . . .	52
16.2.3	Multiple Choice Blocks . . . . .	53
16.2.4	Including Problems . . . . .	53
16.2.5	Reporting Metadata . . . . .	53
16.3	Limitations . . . . .	53
<b>17</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>55</b>
17.1	Introduction . . . . .	56
17.2	The User Interface . . . . .	56
17.2.1	Package and Class Options . . . . .	56
17.2.2	Assignments . . . . .	56
17.2.3	Typesetting Exams . . . . .	56
17.2.4	Including Assignments . . . . .	57
17.3	Limitations . . . . .	57
<b>IV</b>	<b>Implementation</b>	<b>59</b>
<b>18</b>	<b>gTeX-Basics Implementation</b>	<b>60</b>
18.1	The gTeXDocument Class . . . . .	60
18.2	Preliminaries . . . . .	60
18.3	Messages and logging . . . . .	61
18.4	Persistence . . . . .	62
18.5	HTML Annotations . . . . .	62
18.6	Languages . . . . .	65
18.7	Activating/Deactivating Macros . . . . .	66
<b>19</b>	<b>gTeX-MathHub Implementation</b>	<b>68</b>
19.1	Generic Path Handling . . . . .	68
19.2	PWD and kpsewhich . . . . .	70
19.3	File Hooks and Tracking . . . . .	71
19.4	MathHub Repositories . . . . .	72
<b>20</b>	<b>gTeX-References Implementation</b>	<b>78</b>
20.1	Document URIs and URLs . . . . .	78
20.2	Setting Reference Targets . . . . .	80
20.3	Using References . . . . .	81
<b>21</b>	<b>gTeX-Modules Implementation</b>	<b>83</b>
21.1	The module environment . . . . .	86
21.2	Invoking modules . . . . .	91
<b>22</b>	<b>gTeX-Module Inheritance Implementation</b>	<b>93</b>
22.1	SMS Mode . . . . .	93
22.2	Inheritance . . . . .	97

<b>23</b>	<b>STEX-Symbols Implementation</b>	<b>102</b>
23.1	Symbol Declarations . . . . .	102
23.2	Notations . . . . .	108
<b>24</b>	<b>STEX-Terms Implementation</b>	<b>116</b>
24.1	Symbol Invocations . . . . .	116
24.2	Terms . . . . .	119
24.3	Notation Components . . . . .	125
<b>25</b>	<b>STEX-Structural Features Implementation</b>	<b>128</b>
25.1	The feature environment . . . . .	128
25.2	Features . . . . .	130
<b>26</b>	<b>STEX-Statements Implementation</b>	<b>135</b>
26.1	Definitions . . . . .	135
26.2	Assertions . . . . .	138
26.3	Examples . . . . .	140
26.4	Logical Paragraphs . . . . .	142
<b>27</b>	<b>The Implementation</b>	<b>145</b>
27.1	Package Options . . . . .	145
27.2	Proofs . . . . .	145
27.3	Justifications . . . . .	151
<b>28</b>	<b>STEX-Others Implementation</b>	<b>153</b>
<b>29</b>	<b>STEX-Metatheory Implementation</b>	<b>154</b>
<b>30</b>	<b>Tikzinput Implementation</b>	<b>157</b>
<b>31</b>	<b>document-structure.sty Implementation</b>	<b>159</b>
31.1	The OMDoc Class . . . . .	159
31.2	Class Options . . . . .	159
31.3	Beefing up the document environment . . . . .	160
31.4	Implementation: OMDoc Package . . . . .	160
31.5	Package Options . . . . .	160
31.6	Document Structure . . . . .	162
31.7	Front and Backmatter . . . . .	165
31.8	Global Variables . . . . .	167
<b>32</b>	<b>MiKoSlides – Implementation</b>	<b>168</b>
32.1	Class and Package Options . . . . .	168
32.2	Notes and Slides . . . . .	170
32.3	Header and Footer Lines . . . . .	174
32.4	Frame Images . . . . .	175
32.5	Colors and Highlighting . . . . .	176
32.6	Sectioning . . . . .	177
32.7	Excursions . . . . .	179

<b>33 The Implementation</b>	<b>181</b>
33.1 Package Options . . . . .	181
33.2 Problems and Solutions . . . . .	182
33.3 Multiple Choice Blocks . . . . .	187
33.4 Including Problems . . . . .	188
33.5 Reporting Metadata . . . . .	189
<b>34 Implementation: The hwexam Class</b>	<b>191</b>
34.1 Class Options . . . . .	191
<b>35 Implementation: The hwexam Package</b>	<b>193</b>
35.1 Package Options . . . . .	193
35.2 Assignments . . . . .	194
35.3 Including Assignments . . . . .	197
35.4 Typesetting Exams . . . . .	198
35.5 Leftovers . . . . .	200

**Part I**  
**Manual**

# Chapter 1

## Stuff

### 1.1 Modules

---

`\sTeX`  
`\stex`

---

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

#### 1.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

##### Example 1

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

$ab$

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

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



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

### Example 2

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

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

.

Not using an explicit option with a semantic macro yields the first declared notation, unless changed<sup>1</sup>.

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

### Example 3

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

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

.

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

### Example 4

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

Multiplying again by  $b$  yields...

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

### Example 5

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

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

<sup>1</sup>EdNOTE: TODO

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

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

### Example 6

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

The operator  $\textcolor{teal}{+}$  adds two elements, as in  $a\textcolor{teal}{+}b$ .

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

### Example 7

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

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

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

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

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

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

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

## Other Argument Types

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

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

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

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

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

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

### Example 8

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

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

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

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

### Example 9

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

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

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

2 3

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

<sup>3</sup>EDNOTE: “decompose” a-type arguments into fixed-arity operators?

## Precedences

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

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

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

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

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

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

For example:

### Example 10

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

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

## 1.1.2 Archives and Imports

### Namespaces

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

Unfortunately,  $\TeX$  only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that  $\TeX$  can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

- If `\begin{module}{Foo}` occurs in a file `/path/to/file/Foo[.<lang>].tex` which does not belong to an archive, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.<lang>].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix<sup>1</sup>.

If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

## Paths in Import-Statements

Conversely, here is how namespaces/URIs and file paths are computed in import statements, exemplary `\importmodule`:

- `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.<lang>].tex` in the same directory.
- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the archive's `source`-folder.
- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

The module `Foo` must either be declared in the file `<top-directory>/some/path/Foo[.<lang>].tex`, or in `<top-directory>/some/path[.<lang>].tex` (which are checked in that order).

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

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

---

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

## Part II

# Documentation

## Chapter 2

# sTeX-Basics

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

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

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

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

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

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

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

### 2.1 Macros and Environments

---

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

---

---

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

---

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

---

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

---

---

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

---

We have four macros for annotating generated HTML (via L<sup>A</sup>T<sub>E</sub>XML or RusT<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 (heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</code> .
<code>\c_stex_pwd_str</code>	
<code>\c_stex_mainfile_seq</code>	
<code>\c_stex_mainfile_str</code>	

---

---

`\g_stex_currentfile_seq`

---

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

### Test 1

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

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

## 3.1.2 MathHub Archives

---

`\mathhub`

---

`\c_stex_mathhub_seq`

`\c_stex_mathhub_str`

---

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

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

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

---

`\l_stex_current_repository_prop`

---

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

<hr/> <hr/> <code>\stex_set_current_repository:n</code>	Sets the current repository to the one with the provided ID. calls <code>\__stex_mathhub_do_manifest:n</code> , so works whether this repository's MANIFEST.MF-file has already been read or not.
<hr/> <hr/> <code>\stex_require_repository:n</code>	Calls <code>\__stex_mathhub_do_manifest:n</code> iff the corresponding archive property list does not already exist, and adds a corresponding definition to the <code>.sms</code> -file.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{&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: >macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

Meaning: >macro:->\protect \bar <

**Module 6.1.2[Importtest]**

modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo Meaning: >macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

**Module 6.1.3[Importtest2]**

modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Importtest Meaning: >macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

---

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

---

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

### Test 9

```

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

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

```

**Module 6.1.4**[UseTest1]

**Module 6.1.5**[UseTest2]  
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextestUseTest1 Meaning: >undefined<

**Module 6.1.6**[UseTest3]  
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2 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?collect>,  
<http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?collection>,  
<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?module-type>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>,  
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar

### Test 10

```

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

```

Circular dependencies:

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

---

---

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {⟨archive-ID⟩} {⟨module-path⟩}`

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

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

1. If `⟨archive-ID⟩` is empty:

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

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

2. Otherwise:

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

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

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

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

---

---

`\stex_import_require_module:nnnn`

`{⟨ns⟩} {⟨archive-ID⟩} {⟨path⟩} {⟨name⟩}`

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

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

# Chapter 7

## STEX-Symbols

Code related to symbol declarations and notations

### 7.1 Macros and Environments

---

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

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

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

---

---

`\stex_symdecl_do:n`

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

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

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

### Test 11

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

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

---

---

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

---

---

`\stex_get_symbol:n`

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

---

---

`\notation`

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

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

---

---

`\stex_notation_do:nn`

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

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

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

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

### Test 12

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

Module 7.1.2[NotationTest]  
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo

---

---

`\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/> <hr/>	$\backslash\infprec$ $\backslashneginfprec$	Maximal and minimal notation precedences.
<hr/> <hr/>	$\backslashdobrackets$	$\backslashdobrackets \{ \langle body \rangle \}$  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 $\backslash\withbrackets$ .
<hr/> <hr/>	$\backslash\withbrackets$	$\backslash\withbrackets \langle left \rangle \langle right \rangle \{ \langle body \rangle \}$  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 $\backslash\left$ and $\backslash\right$ 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]  
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo  $\langle a^b_c \rangle$   
and  $\langle a^b_c \rangle$ .

### 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  $\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$ 
 $\plus{a,\mult{b,c}}$  and  $\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$ 
\displaystyle \plus{a,\mult{b,c}} and
\displaystyle \mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}
\withbrackets[ { $ \displaystyle }
\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}
\end{module}

```

Module 8.1.2[MathTest2]  
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo  $\langle a|[b:c,d:e,f]^9 \rangle$   
and  $\langle a|[b:c]^9 \rangle$  and  $\langle a|[b]^c \rangle$   
 $a+(b \cdot c)$  and  $a \cdot \frac{a}{b} + \frac{a}{c}$   
 $a+(b \cdot c)$  and  $a \cdot \frac{a}{b} + \frac{a}{c}$

---

---

`\stex_term_custom:nn`

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

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

### Test 16

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

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

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

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

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

\bar![bar]

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

\end{module}
```

```
Module 8.1.3[TextTest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo
some aand some band also some here.
some a and some b and also some c here.
bar
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

#### 9.1.1 Structures

mathstructure    TODO

#### Test 17

```

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

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

\present\magma

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

Test: $\mM{op}ab$

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

```

Module 9.1.1[StructureTest1]
a**b**:*M*
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}
feature?op
>macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}
Test: a+b
Test2: (*U*,+)

## Chapter 10

# sTeX-Statements

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

### 10.1 Macros and Environments

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

## Chapter 11

# sTeX-Proofs: Structural Markup for Proofs

The `sproof` package is part of the sTeX collection, a version of T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X that allows to markup T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X documents semantically without leaving the document format, essentially turning T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X into a document format for mathematical knowledge management (MKM).

This package supplies macros and environment that allow to annotate the structure of mathematical proofs in sTeX files. This structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation.

## Contents

## 11.1 Introduction

The **sproof** (semantic proofs) package supplies macros and environment that allow to annotate the structure of mathematical proofs in  $\text{\LaTeX}$  files. This structure can be used by MKM systems for added-value services, either directly from the  $\text{\LaTeX}$  sources, or after translation. Even though it is part of the  $\text{\LaTeX}$  collection, it can be used independently, like it's sister package **statements**.

$\text{\LaTeX}$  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).

```
\begin{sproof}[id=simple-proof,for=sum-over-odds]
  {We prove that  $\sum_{i=1}^n (2i-1) = n^2$  by induction over  $n$ }
  \begin{spfcase}{For the induction we have to consider the following cases:}
    \begin{spfcase}{ $n=1$ }
      \begin{spfstep}[display=flow] then we compute  $1=1^2$ \end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n=2$ }
      \begin{sproofcomment}[display=flow]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[display=flow] We compute  $1+3=2^2=4$ .\end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n>1$ }
      \begin{spfstep}[type=assumption,id=ind-hyp]
        Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
        i.e.  $\sum_{i=1}^k (2i-1) = k^2$ $.
      \end{spfstep}
      \begin{sproofcomment}
        We have to show that we can derive the assertion for  $n=k+1$  from
        this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ $.
      \end{sproofcomment}
      \begin{spfstep}
        We obtain  $\sum_{i=1}^{k+1} (2i-1) = \sum_{i=1}^k (2i-1) + 2(k+1) - 1$ 
        \begin{justification}[method=arith:split-sum]
          by splitting the sum.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}
        Thus we have  $\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1$ 
        \begin{justification}[method=fertilize]
          by inductive hypothesis.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}[type=conclusion]
        We can \begin{justification}[method=simplify]simplify\end{justification}
        the right-hand side to  $(k+1)^2$ , which proves the assertion.
      \end{spfstep}
    \end{spfcase}
    \begin{spfstep}[type=conclusion]
      We have considered all the cases, so we have proven the assertion.
    \end{spfstep}
  \end{spfcase}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).<sup>4</sup>

<sup>4</sup>EdNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.



## 11.2 The User Interface

### 11.2.1 Package Options

`showmeta` The `sproof` package takes a single option: `showmeta`. If this is set, then the metadata keys are shown (see [Kohlhase:metakeys] for details and customization options).

### 11.2.2 Proofs and Proof steps

`sproof` The `proof` environment is the main container for proofs. It takes an optional `KeyVal` argument that allows to specify the `id` (identifier) and `for` (for which assertion is this a proof) keys. The regular argument of the `proof` environment contains an introductory comment, that may be used to announce the proof style. The `proof` environment contains a sequence of `\step`, `proofcomment`, and `pfcases` environments that are used to markup the proof steps. The `proof` environment has a variant `Proof`, which does not use the proof end marker. This is convenient, if a proof ends in a case distinction, which brings it's own proof end marker with it. The `Proof` environment is a variant of `proof` that does not mark the end of a proof with a little box; presumably, since one of the subproofs already has one and then a box supplied by the outer proof would generate an otherwise empty line. The `\spfidea` macro allows to give a one-paragraph description of the proof idea.

`spfsketch` For one-line proof sketches, we use the `\spfsketch` macro, which takes the `KeyVal` argument as `sproof` and another one: a natural language text that sketches the proof.

`spfstep` Regular proof steps are marked up with the `step` environment, which takes an optional `KeyVal` argument for annotations. A proof step usually contains a local assertion (the text of the step) together with some kind of evidence that this can be derived from already established assertions.

Note that both `\premise` and `\justarg` can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.

### 11.2.3 Justifications

`justification` This evidence is marked up with the `justification` environment in the `sproof` package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional `KeyVal` argument, which can have the `method` key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain “premises” (specifications to assertions that were used justify the step) and “arguments” (other information taken into account by the proof method).

`\premise` The `\premise` macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the example in Figure 1 we have used the `\premise` macro to identify the inductive hypothesis.

`\justarg` The `\justarg` macro is very similar to `\premise` with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of `\premise`. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a `\justarg` macro.

<b>Proof:</b>	We prove that $\sum_{i=1}^n 2i - 1 = n^2$ by induction over $n$	
<b>P.1</b>	For the induction we have to consider the following cases:	
<b>P.1.1</b>	$n = 1$ : then we compute $1 = 1^2$	□
<b>P.1.1</b>	$n = 2$ : This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$	□
<b>P.1.1</b>	$n > 1$ :	
<b>P.1.1.1</b>	Now, we assume that the assertion is true for a certain $k \geq 1$ , i.e. $\sum_{i=1}^k (2i - 1) = k^2$ .	
<b>P.1.1.1</b>	We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$ .	
<b>P.1.1.1</b>	We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum	
<b>P.1.1.1</b>	Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.	
<b>P.1.1.1</b>	We can simplify the right-hand side to $(k + 1)^2$ , which proves the assertion.	□
<b>P.1.1</b>	We have considered all the cases, so we have proven the assertion.	□

Example 2: The formatted result of the proof in Figure 1

#### 11.2.4 Proof Structure

<b>subproof</b>	The <b>pfcases</b> environment is used to mark up a subproof. This environment takes an optional <b>KeyVal</b> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <b>proof</b> environment). The <b>method</b> key can be used to give the name of the proof method executed to make this subproof.
<b>spfcases</b>	The <b>pfcases</b> environment is used to mark up a proof by cases. Technically it is a variant of the <b>subproof</b> where the <b>method</b> is <b>by-cases</b> . Its contents are <b>spfcases</b> environments that mark up the cases one by one.
<b>spfcases</b>	The content of a <b>pfcases</b> environment are a sequence of case proofs marked up in the <b>pfcases</b> environment, which takes an optional <b>KeyVal</b> argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a <b>pfcases</b> environment is the same as that of a <b>proof</b> , i.e. <b>steps</b> , <b>proofcomments</b> , and <b>pfcases</b> environments. <b>\spfcasesketch</b> is a variant of the <b>spfcases</b> environment that takes the same arguments, but instead of the <b>spfsteps</b> in the body uses a third argument for a proof sketch.
<b>\spfcasesketch</b>	
<b>sproofcomment</b>	The <b>proofcomment</b> environment is much like a <b>step</b> , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a <b>\premise</b> .



1. The numbering scheme of proofs cannot be changed. It is more geared for teaching proof structures (the author's main use case) and not for writing papers. reported by Tobias Pfeiffer (fixed)
2. currently proof steps are formatted by the `LATEX description` environment. We would like to configure this, e.g. to use the `inparaenum` environment for more condensed proofs. I am just not sure what the best user interface would be I can imagine redefining an internal environment `spf@proofstep@list` or adding a key `prooflistenv` to the `proof` environment that allows to specify the environment directly. Maybe we should do both.

## Chapter 12

# 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.

### 12.1 Symbols

**Part III**  
**Extensions**

## Chapter 13

# Tikzinput

### 13.1 Macros and Environments

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

## Chapter 14

# 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  $\text{\texttt{sTeX}}$  collection, a version of  $\text{\texttt{TeX/LaTeX}}$  that allows to markup  $\text{\texttt{TeX/LaTeX}}$  documents semantically without leaving the document format, essentially turning  $\text{\texttt{TeX/LaTeX}}$  into a document format for mathematical knowledge management (MKM).

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

### 14.1 Introduction

$\text{\texttt{sTeX}}$  is a version of  $\text{\texttt{TeX/LaTeX}}$  that allows to markup  $\text{\texttt{TeX/LaTeX}}$  documents semantically without leaving the document format, essentially turning  $\text{\texttt{TeX/LaTeX}}$  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  $\text{\texttt{sTeX}}$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\text{\texttt{sTeX}}$  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>6</sup>

## 14.2 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`.

### 14.2.1 Package and Class Options

The `omdoc` class accept the following options:

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

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

### 14.2.2 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>6</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 3 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 3: 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.

### 14.2.3 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].

#### 14.2.4 Structure Sharing

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

`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in  $\text{\LaTeX}$ . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.<sup>7</sup>

#### 14.2.5 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>7</sup>EdNOTE: document LMID und LMXRef here if we decide to keep them.

### 14.2.6 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.

## 14.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `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 15

## Slides and Course Notes

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

### 15.1 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`.

### 15.2 The User Interface

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

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

#### 15.2.1 Package Options

The `mikoslides` class takes a variety of class options:<sup>8</sup>

- |                           |  |
|---------------------------|--|
| <code>slides</code>       | <ul style="list-style-type: none"><li>• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 15.2.2).</li></ul>   |
| <code>notes</code>        |  |
| <code>sectocframes</code> | <ul style="list-style-type: none"><li>• If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.</li></ul> |

EdN:8

<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 15.2.4). 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>

### 15.2.2 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 4: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 4.

`\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>8</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.

### 15.2.3 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.

### 15.2.4 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>9</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>9</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}`

## 15.2.5 Colors and Highlighting

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

## 15.2.6 Front Matter, Titles, etc.

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

## 15.2.8 Miscellaneous

## 15.3 Limitations

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

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



## Chapter 16

# problem.sty: An Infrastructure for formatting Problems

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

### 16.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions<sup>5</sup>. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the `problem` package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

### 16.2 The User Interface

#### 16.2.1 Package Options

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in
	a test situation, so this option does not show the solutions (of course), but leaves space
	for the students to solve them.
<code>mh</code>	The <code>mh</code> option turns on MathHub support; see [ <code>Kohlhase:mss</code> ].
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [ <code>Kohlhase:metakeys</code> ]
	for details and customization options).

---

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

## 16.2.2 Problems and Solutions

**problem** The main environment provided by the **problem** package is (surprise surprise) the **problem** environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys **id** as an identifier that can be reference later, **pts** for the points to be gained from this exercise in homework or quiz situations, **min** for the estimated minutes needed to solve the problem, and finally **title** for an informative title of the problem. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{problem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants can you fit into a Volkswagen beetle?
  \begin{hint}
    Think positively, this is simple!
  \end{hint}
  \begin{exnote}
    Justify your answer
  \end{exnote}
  \begin{solution}[for=elephants,height=3cm]
    Four, two in the front seats, and two in the back.
  \begin{gnote}
    if they do not give the justification deduct 5 pts
  \end{gnote}
  \end{solution}
  \end{problem}
\end{document}
```

Example 5: A marked up Problem

**solution** The **solution** environment can be to specify a solution to a problem. If the **solutions** option is set or **\solutionstrue** is set in the text, then the solution will be presented in the output. The **solution** environment takes an optional KeyVal argument with the keys **id** for an identifier that can be reference **for** to specify which problem this is a solution for, and **height** that allows to specify the amount of space to be left in test situations (i.e. if the **test** option is set in the **\usepackage** statement).

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

may arise in grading.

Sometimes we would like to locally override the `solutions` option we have given to the package. To turn on solutions we use the `\startsolutions`, to turn them off, `\stopsolutions`. These two can be used at any point in the documents.

Also, sometimes, we want content (e.g. in an exam with master solutions) conditional on whether solutions are shown. This can be done with the `\ifsolutions` conditional.

### 16.2.3 Multiple Choice Blocks

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc[⟨keyvals⟩]{⟨text⟩}` macro, which takes an optional key/value argument `⟨keyvals⟩` for choice metadata and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

- `T` • `T` for true answers, `F` for false ones,
- `F` • `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `Ttext` • `feedback` for a short feedback text given to the student.
- `Ftext`
- `feedback`

See Figure ?? for an example

### 16.2.4 Including Problems

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

### 16.2.5 Reporting Metadata

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

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

## 16.3 Limitations

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

1. none reported yet

```

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

```

---

**Problem0.0 ()**

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

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

---

**Problem0.0 ()**

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

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

Example 7: A Problem with a multiple choice block

## Chapter 17

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

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

### Contents

## 17.1 Introduction

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

## 17.2 The User Interface

### 17.2.1 Package and Class Options

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

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

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

### 17.2.2 Assignments

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

### 17.2.3 Typesetting Exams

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

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the L<sup>A</sup>T<sub>E</sub>X source.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical  
`\testnewpage` space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage`  
`\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

`testheading` Finally, the `\testheading` takes an optional keyword argument where the keys  
`duration` `duration` specifies a string that specifies the duration of the test, `min` specifies the equiv-  
`min` alent in number of minutes, and `reqpts` the points that are required for a perfect grade.  
`reqpts`

### 17.2.4 Including Assignments

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

### 17.3 Limitations

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

1. none reported yet.

---

Name:
MatriculationNumber:

2022-01-24

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

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

[illegible]

Example 8: A generated test heading.



**Part IV**  
**Implementation**

## Chapter 18

# ST<sub>E</sub>X -Basics Implementation

### 18.1 The ST<sub>E</sub>XDocument 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>
```

### 18.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%%% basics.dtx %%%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-compat}
22 \RequirePackage{morewrites}
23 \RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
27   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 ,
28 mathhub .tl_set_x:N = \mathhub ,
29 sms .bool_set:N = \c_stex_persist_mode_bool ,
30 image .bool_set:N = \c_tikzinput_image_bool ,
31 unknown .code:n = {}
32 }
33 \ProcessKeysOptions { stex }

```

**\stex** The  $\text{\TeX}$  logo:

**\sTeX**

```

34 \protected\def\stex{%
35 \ifundefined{texorpdfstring}%
36 {\let\texorpdfstring\@firstoftwo}%
37 {}%
38 \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
39 }
40 \def\sTeX{\stex}

```

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

## 18.3 Messages and logging

```

41 <@@=stex_log>

Warnings and error messages
42 \msg_new:nnn{stex}{error/unknownlanguage}{
43   Unknown~language:~#1
44 }
45 \msg_new:nnn{stex}{warning/nomathhub}{
46   MATHHUB~system~variable~not~found~and~no~
47   \detokenize{\mathhub}~value~set!
48 }
49 \msg_new:nnn{stex}{error/deactivated-macro}{
50   The~\detokenize{#1}~command~is~only~allowed~in~#2!
51 }

```

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

```

52 \cs_new_protected:Nn \stex_debug:nn {
53   \clist_if_in:NnTF \c_stex_debug_clist { all } {
54     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
55       \\Debug~#1:~#2\\
56     }
57     \msg_none:nn{stex}{debug / #1}
58   }{
59     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
60       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
61         \\Debug~#1:~#2\\
62       }
63       \msg_none:nn{stex}{debug / #1}
64     }
65   }
66 }

```

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

Redirecting messages:

```

67 \clist_if_in:NnTF \c_stex_debug_clist {all} {
68   \msg_redirect_module:nnn{ stex }{ none }{ term }
69 }{
70   \clist_map_inline:Nn \c_stex_debug_clist {
71     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
72   }
73 }
74
75 \stex_debug:nn{log}{debug~mode~on}

```

## 18.4 Persistence

76 `<@=stex_persist>`

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

```

77 \iow_new:N \c__stex_persist_sms_iow
78 \AddToHook{begindocument}{
79   \bool_if:NTF \c_stex_persist_mode_bool {
80     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
81   } {
82     \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
83   }
84 }
85 \AddToHook{enddocument}{
86   \bool_if:NF \c_stex_persist_mode_bool {
87     \iow_close:N \c__stex_persist_sms_iow
88   }
89 }

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

90 \cs_new_protected:Nn \stex_add_to_sms:n {
91   \bool_if:NF \c_stex_persist_mode_bool {
92     \iow_now:Nn \c__stex_persist_sms_iow { #1 }
93   }
94 }

```

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

## 18.5 HTML Annotations

95 `<@=stex_annotate>`  
96 `\RequirePackage{rustex}`

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

```

97 \rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}

```

`\if@latexml` Conditionals for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L:

```

\latexml_if_p:
\latexml_if:TF
98 \ifcsname if@latexml\endcsname\else

```

```

99     \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
100 \fi
101
102 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
103   \if@latexml
104     \prg_return_true:
105   \else:
106     \prg_return_false:
107   \fi:
108 }

```

(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`

```

109 \tl_new:N \l__stex_annotate_arg_tl
110 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
111   \rustex_if:TF {
112     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
113   }{-}
114 }

```

(End definition for \l\_\_stex\_annotate\_arg\_tl and \c\_\_stex\_annotate\_emptyarg\_tl.)

`\_stex_annotate_checkempty:n`

```

115 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
116   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
117   \tl_if_empty:NT \l__stex_annotate_arg_tl {
118     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
119   }
120 }

```

(End definition for \\_stex\_annotate\_checkempty:n.)

`\l_stex_html_do_output_bool` Whether to (locally) produce HTML output

```

\stex_if_do_html:
121 \bool_new:N \l_stex_html_do_output_bool
122 \bool_set_true:N \l_stex_html_do_output_bool
123 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
124   \bool_if:nTF \l_stex_html_do_output_bool
125     \prg_return_true: \prg_return_false:
126 }

```

(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

```

127 \cs_new_protected:Nn \stex_suppress_html:n {
128   \exp_args:Nne \use:nn {
129     \bool_set_false:N \l_stex_html_do_output_bool
130     #1
131   }{
132     \stex_if_do_html:T {
133       \bool_set_true:N \l_stex_html_do_output_bool
134     }
135   }
136 }

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

137 \rustex_if:TF{
138   \cs_new_protected:Nn \stex_annotate:nnn {
139     \__stex_annotate_checkempty:n { #3 }
140     \rustex_annotate_HTML:nn {
141       property="stex:#1" ~
142       resource="#2"
143     } {
144       \mode_if_vertical:TF{
145         \tl_use:N \l__stex_annotate_arg_tl\par
146       }{
147         \tl_use:N \l__stex_annotate_arg_tl
148       }
149     }
150   }
151   \cs_new_protected:Nn \stex_annotate_invisible:n {
152     \__stex_annotate_checkempty:n { #1 }
153     \rustex_annotate_HTML:nn {
154       stex:visible="false" ~
155       style:display="none"
156     } {
157       \mode_if_vertical:TF{
158         \tl_use:N \l__stex_annotate_arg_tl\par
159       }{
160         \tl_use:N \l__stex_annotate_arg_tl
161       }
162     }
163   }
164   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
165     \__stex_annotate_checkempty:n { #3 }
166     \rustex_annotate_HTML:nn {
167       property="stex:#1" ~
168       resource="#2" ~
169       stex:visible="false" ~
170       style:display="none"
171     } {
172       \mode_if_vertical:TF{
173         \tl_use:N \l__stex_annotate_arg_tl\par
174       }{
175         \tl_use:N \l__stex_annotate_arg_tl
176       }
177     }
178   }
179   \NewDocumentEnvironment{stex_annotate_env} { m m } {
180     \par
181     \rustex_annotate_HTML_begin:n {
182       property="stex:#1" ~
183       resource="#2"
184     }

```

```

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

```

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

## 18.6 Languages

```

232 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

```

## 18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

270 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
271   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
272   \def#1{
273     \msg_error:nxxx{stex}{error/deactivated-macro}{#1}{#2}
274   }
275 }

```



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

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

```
276 \cs_new_protected:Nn \stex_reactivate_macro:N {  
277   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname  
278 }
```

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

```
279 \</package>
```

## Chapter 19

# STEX -MathHub Implementation

```
280 <*package>
281
282 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
283
284 <@@=stex_path>
285
286 Warnings and error messages
287 \msg_new:nnn{stex}{error/norepository}{
288   No~archive~#1~found~in~#2
289 }
290 \msg_new:nnn{stex}{error/notinarchive}{
291   Not~currently~in~an~archive,~but~\detokenize{#1}~
292   needs~one!
293 }
294 \msg_new:nnn{stex}{error/nofile}{
295   \detokenize{#1}~could~not~find~file~#2
296 }
```

### 19.1 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
295 \cs_new_protected:Nn \stex_path_from_string:Nn {
296   \str_set:Nx \l_tmpa_str { #2 }
297   \str_if_empty:NTF \l_tmpa_str {
298     \seq_clear:N #1
299   }{
300     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
301     \sys_if_platform_windows:T{
302       \seq_clear:N \l_tmpa_tl
303       \seq_map_inline:Nn #1 {
304         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
305         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
306       }
307     }
308   }
```

```

306     }
307     \seq_set_eq:NN #1 \l_tmpa_tl
308   }
309   \stex_path_canonicalize:N #1
310 }
311 }
312 \cs_generate_variant:Nn \stex_path_from_string:Nn
313 { 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
314 \cs_new_protected:Nn \stex_path_to_string:NN {
315   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
316 }
317
318 \cs_new:Nn \stex_path_to_string:N {
319   \seq_use:Nn #1 /
320 }

```

(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
321 \str_const:Nn \c__stex_path_dot_str {.}
322 \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.

```

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

```

```

347     }
348   }{
349     \str_if_empty:NF \l_tmpa_tl {
350       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
351     }
352   }
353 }
354 }
355 \seq_gset_eq:NN #1 \l_tmpa_seq
356 }
357 }

```

(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`

```

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

```

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

## 19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(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`

```

376 \sys_if_platform_windows:TF{
377   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
378 }{
379   \stex_kpsewhich:n{-var-value~PWD}
380 }
381
382 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
383 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
384 \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.)

## 19.3 File Hooks and Tracking

385 `<@@=stex_files>`

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

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

`\g__stex_files_stack` keeps track of file changes

386 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

388 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

389 `\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`.

```

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

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

## 19.4 MathHub Repositories

```

414 <@@=stex_mathhub>

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

```

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

```

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

```

\l__stex_mathhub_manifest_file_seq
451 \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`:

```

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

```

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

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

487 \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:

```

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

```

```

496 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
497 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
498 }
499 \exp_args:No \str_case:nnTF \l_tmpa_tl {
500 {id} {
501 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
502 { id } \l_tmpb_tl
503 }
504 {narration-base} {
505 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
506 { narr } \l_tmpb_tl
507 }
508 {url-base} {
509 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
510 { docurl } \l_tmpb_tl
511 }
512 {source-base} {
513 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
514 { ns } \l_tmpb_tl
515 }
516 {ns} {
517 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
518 { ns } \l_tmpb_tl
519 }
520 {dependencies} {
521 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
522 { deps } \l_tmpb_tl
523 }
524 }{}{}
525 }{}
526 }
527 \ior_close:N \c__stex_mathhub_manifest_ior
528 }

```

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

`\stex_set_current_repository:n`

```

529 \cs_new_protected:Nn \stex_set_current_repository:n {
530 \stex_require_repository:n { #1 }
531 \prop_set_eq:Nc \l_stex_current_repository_prop {
532 c_stex_mathhub_#1_manifest_prop
533 }
534 }

```

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

`\stex_require_repository:n`

```

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

```



```

543     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
544     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
545   }
546 }
547 }
548 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

549 \prop_new:N \l_stex_current_repository_prop
550
551 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
552 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
553   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
554 } {
555   \__stex_mathhub_parse_manifest:n { main }
556   \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
557   \l_tmpa_str
558   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
559   \c_stex_mathhub_main_manifest_prop
560   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
561   \stex_debug:nn{mathhub}{Current~repository:~
562     \prop_item:Nn \l_stex_current_repository_prop {id}
563   }
564 }

```

(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.

```

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

```

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

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn

585 \newif \ifinputref \inputreffalse
586
587 \cs_new_protected:Nn \stex_mhinput:nn {
588   \stex_in_repository:nn {#1} {
589     \ifinputref
590       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
591     \else
592       \inputreftrue
593       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
594       \inputreffalse
595     \fi
596   }
597 }
598 \NewDocumentCommand \mhinput { 0{} m}{
599   \stex_mhinput:nn{ #1 }{ #2 }
600 }
601
602 \cs_new_protected:Nn \stex_inputref:nn {
603   \stex_in_repository:nn {#1} {
604     \bool_lazy_any:nTF {
605       {\rustex_if_p:} {\latexml_if_p:}
606     } {
607       \str_clear:N \l_tmpa_str
608       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
609         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
610       }
611       \stex_annotate_invisible:nnn{inputref}{
612         \l_tmpa_str / #2
613       }{}
614     }{
615       \begingroup
616         \inputreftrue
617         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
618       \endgroup
619     }
620   }
621 }
622
623 \NewDocumentCommand \inputref { 0{} m}{
624   \stex_inputref:nn{ #1 }{ #2 }
625 }
626
627 \cs_new_protected:Nn \stex_mhbibresource:nn {
628   \stex_in_repository:nn {#1} {
629     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
630   }
631 }
632 \newcommand\addmhbibresource[2] []{
633   \stex_mhbibresource:nn{ #1 }{ #2 }
634 }

```

(End definition for `\inputref`, `\stex_inputref:nn`, and `\mhinput\stex_mhinput:nn`. These functions are documented on page 13.)

**\mhpath**

```
635 \def \mhpath #1 #2 {
636   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
637     \c_stex_mathhub_str /
638     \prop_item:Nn \l_stex_current_repository_prop { id }
639     / source / #2
640   }{
641     \c_stex_mathhub_str / #1 / source / #2
642   }
643 }
```

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

**\libinput**

```
644 \cs_new_protected:Npn \libinput #1 {
645   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
646     \msg_error:nnn{stex}{error/notinarchive}\libinput
647   }
648   \bool_set_false:N \l_tmpa_bool
649   \tl_clear:N \l_tmpa_tl
650   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
651   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
652   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
653   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
654     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
655     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
656       / meta-inf / lib / #1.tex}{
657       \bool_set_true:N \l_tmpa_bool
658       \tl_put_right:Nx \l_tmpa_tl {
659         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
660           / meta-inf / lib / #1.tex}
661       }
662     }{}
663   }
664   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
665     / \l_tmpa_str / lib / #1.tex
666   }{
667     \bool_set_true:N \l_tmpa_bool
668     \tl_put_right:Nx \l_tmpa_tl {
669       \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
670         / \l_tmpa_str / lib / #1.tex}
671     }
672   }{}
673   \bool_if:NF \l_tmpa_bool {
674     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
675   }
676   \l_tmpa_tl
677 }
```

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

```
678 \</package>
```

## Chapter 20

# STEX -References Implementation

```
679 <*package>
680
681 %%%%%%%%%% references.dtx %%%%%%%%%%
682
683 %\RequirePackage{hyperref}
684 %\RequirePackage{cleveref}
685 <@@=stex_refs>
686
687 Warnings and error messages
688
689 \iow_new:N \c__stex_refs_refs_iow
690 \AddToHook{begindocument}{
691   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
692 }
693 \AddToHook{enddocument}{
694   \iow_close:N \c__stex_refs_refs_iow
695 }
696
697 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
698
699 \NewDocumentCommand \STEXreftitle { m } {
700   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
701 }
702
```

### 20.1 Document URIs and URLs

```
700 \seq_new:N \g__stex_refs_all_refs_seq
701
702 \str_new:N \l_stex_current_docns_str
703
704 \cs_new_protected:Nn \stex_get_document_uri: {
705   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
706   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
707   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
708   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
709 }
```

```

709 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
710
711 \str_clear:N \l_tmpa_str
712 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
713   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
714 }
715
716 \str_if_empty:NTF \l_tmpa_str {
717   \str_set:Nx \l_stex_current_docns_str {
718     file:/\stex_path_to_string:N \l_tmpa_seq
719   }
720 }{
721   \bool_set_true:N \l_tmpa_bool
722   \bool_while_do:Nn \l_tmpa_bool {
723     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
724     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
725       {source} { \bool_set_false:N \l_tmpa_bool }
726     }{}{
727       \seq_if_empty:NT \l_tmpa_seq {
728         \bool_set_false:N \l_tmpa_bool
729       }
730     }
731   }
732
733   \seq_if_empty:NTF \l_tmpa_seq {
734     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
735   }{
736     \str_set:Nx \l_stex_current_docns_str {
737       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
738     }
739   }
740 }
741 }
742
743 \str_new:N \l_stex_current_docurl_str
744 \cs_new_protected:Nn \stex_get_document_url: {
745   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
746   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
747   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
748   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
749   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
750
751   \str_clear:N \l_tmpa_str
752   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
753     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
754       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
755     }
756   }
757
758   \str_if_empty:NTF \l_tmpa_str {
759     \str_set:Nx \l_stex_current_docurl_str {
760       file:/\stex_path_to_string:N \l_tmpa_seq
761     }
762   }{
763     \bool_set_true:N \l_tmpa_bool

```

```

763 \bool_while_do:Nn \l_tmpa_bool {
764   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
765   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
766     {source} { \bool_set_false:N \l_tmpa_bool }
767   }{}{
768     \seq_if_empty:NT \l_tmpa_seq {
769       \bool_set_false:N \l_tmpa_bool
770     }
771   }
772 }
773
774 \seq_if_empty:NTF \l_tmpa_seq {
775   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
776 }{
777   \str_set:Nx \l_stex_current_docurl_str {
778     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
779   }
780 }
781 }
782 }

```

## 20.2 Setting Reference Targets

```

783 \str_const:Nn \c__stex_refs_url_str{URL}
784 \str_const:Nn \c__stex_refs_ref_str{REF}
785 % @currentlabel -> number
786 % @currentlabelname -> title
787 % @currentHref -> name.number <- id of some kind
788 % \theH# -> \arabic{section}
789 % \the# -> number
790 % \hyper@makecurrent{#}
791 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
792   \stex_get_document_uri:
793   \str_set:Nx \l_tmpa_str { #1 }
794   \str_if_empty:NT \l_tmpa_str {
795     \int_zero:N \l_tmpa_int
796     \bool_set_true:N \l_tmpa_bool
797     \bool_while_do:Nn \l_tmpa_bool {
798       \cs_if_exist:cTF {
799         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
800       }{
801         \int_incr:N \l_tmpa_int
802       }{
803         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
804         \bool_set_false:N \l_tmpa_bool
805       }
806     }
807   }
808   \str_set:Nx \l_tmpa_str {
809     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
810   }
811   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
812   \stex_if_smsmode:TF {
813     \stex_get_document_url:

```

```

814 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
815 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
816 }{
817 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
818 \exp_args:Nx\label{sref_\l_tmpa_str}
819 \str_gset:cx {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
820 }
821 }

822 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
823 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
824 }

```

## 20.3 Using References

```

825 \str_new:N \l__stex_refs_indocument_str
826 \keys_define:nn { stex / sref } {
827 linktext .tl_set:N = \l__stex_refs_linktext_tl ,
828 fallback .tl_set:N = \l__stex_refs_fallback_tl ,
829 pre .tl_set:N = \l__stex_refs_pre_tl ,
830 post .tl_set:N = \l__stex_refs_post_tl ,
831 %indoc .str_set_x:N = \l__stex_refs_repo_str ,
832 }
833
834 \bool_new:N \c__stex_refs_hyperref_bool
835 \bool_set_false:N \c__stex_refs_hyperref_bool
836 \AddToHook{begindocument}{
837 \ifpackageloaded{hyperref}{
838 \bool_set_true:N \c__stex_refs_hyperref_bool
839 }{}
840 }
841
842
843 \cs_new_protected:Nn \__stex_refs_args:n {
844 \tl_clear:N \l__stex_refs_linktext_tl
845 \tl_clear:N \l__stex_refs_fallback_tl
846 \tl_clear:N \l__stex_refs_pre_tl
847 \tl_clear:N \l__stex_refs_post_tl
848 \str_clear:N \l__stex_refs_repo_str
849 \keys_set:nn { stex / sref } { #1 }
850 }
851
852 \NewDocumentCommand \sref { 0{} m}{
853 \__stex_refs_args:n { #1 }
854 \str_if_empty:NTF \l__stex_refs_indocument_str {
855 \str_set:Nn \l_tmpa_str { #2 }
856 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
857 \tl_set:Nn \l_tmpa_tl {
858 \l__stex_refs_fallback_tl
859 }
860 \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
861 \str_set:Nn \l_tmpb_str { ##1 }
862 \str_if_eq:eeT { \l_tmpa_str } {
863 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
864 } {

```

```

865 \seq_map_break:n {
866   \tl_set:Nn \l_tmpa_tl {
867     % doc uri in \l_tmpb_str
868     \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str_type}}
869     \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
870       % reference
871       \cs_if_exist:cTF{autoref}{
872         \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
873       }{
874         \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
875       }
876     }{
877       % URL
878       \if_bool:N \c__stex_refs_hyperref_bool {
879         \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str_str}}{\l__stex_refs_fallback
880       }{
881         \l__stex_refs_fallback_tl
882       }
883     }
884   }
885 }
886 }
887 }
888 \l_tmpa_tl
889 }{
890   % TODO
891 }
892 }
893
894 \end{package}

```



## Chapter 21

# STEX -Modules Implementation

```
895 <*package>
896
897 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
898
899 <@@=stex_modules>
    Warnings and error messages
900 \msg_new:nnn{stex}{error/unknownmodule}{
901   No~module~#1~found
902 }
903 \msg_new:nnn{stex}{error/syntax}{
904   Syntax~error:~#1
905 }
906 \msg_new:nnn{stex}{error/siglanguage}{
907   Module~#1~declares~signature~#2,~but~does~not~
908   declare~its~language
909 }
```

`\l_stex_current_module_prop` The current module:

```
910 \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

```
911 \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`

```
912 \seq_new:N \g_stex_modules_in_file_seq
913 \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
914 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
915   \prop_if_empty:NTF \l_stex_current_module_prop
916   \prg_return_false: \prg_return_true:
917 }

```

(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
918 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
919   \prop_if_exist:cTF { c_stex_module_#1_prop }
920   \prg_return_true: \prg_return_false:
921 }

```

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

```

\stex_add_to_current_module:n
\STEXexport
922 \cs_new_protected:Nn \stex_add_to_current_module:n {
923   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
924   \tl_put_right:Nn \l_tmpa_tl { #1 }
925   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
926 }
927 \cs_new_protected:Npn \STEXexport {
928   \begingroup
929   \newlinechar=-1\relax
930   \endlinechar=-1\relax
931   %\catcode'\ = 9\relax
932   \expandafter\endgroup\STEXexport:n
933 }
934 \cs_new_protected:Nn \STEXexport:n {
935   \ignorespaces #1
936   \stex_add_to_current_module:n { \ignorespaces #1 }
937   \stex_smsmode_set_codes:
938 }
939 \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
940 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
941   \str_set:Nx \l_tmpa_str { #1 }
942   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
943   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
944   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
945 }

```

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

```

\stex_add_import_to_current_module:n
946 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
947   \str_set:Nx \l_tmpa_str { #1 }
948   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
949   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
950   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
951 }

```

(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).

```

952 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
953   \str_set:Nx \l_tmpa_str { #1 }
954   \seq_set_eq:NN \l_tmpa_seq #2
955   % split off file extension
956   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
957   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
958   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
959   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
960
961   \bool_set_true:N \l_tmpa_bool
962   \bool_while_do:Nn \l_tmpa_bool {
963     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
964     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
965       {source} { \bool_set_false:N \l_tmpa_bool }
966     }{}{
967       \seq_if_empty:NT \l_tmpa_seq {
968         \bool_set_false:N \l_tmpa_bool
969       }
970     }
971   }
972
973   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
974   \str_if_empty:NTF \l_stex_modules_subpath_str {
975     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
976   }{
977     \str_set:Nx \l_stex_modules_ns_str {
978       \l_tmpa_str/\l_stex_modules_subpath_str
979     }
980   }
981 }

```

(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`

```

982 \str_new:N \l_stex_modules_ns_str
983 \str_new:N \l_stex_modules_subpath_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.

```

984 \cs_new_protected:Nn \stex_modules_current_namespace: {
985   \str_clear:N \l_stex_modules_subpath_str
986   \prop_get:NnTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
987     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
988   }{
989     % split off file extension
990     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
991     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str

```

```

992     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
993     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
994     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
995     \str_set:Nx \l_stex_modules_ns_str {
996         file:/\stex_path_to_string:N \l_tmpa_seq
997     }
998 }
999 }

```

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

## 21.1 The module environment

module arguments:

```

1000 \keys_define:nn { stex / module } {
1001     title          .str_set_x:N = \l_stex_module_title_str ,
1002     ns             .str_set_x:N = \l_stex_module_ns_str ,
1003     lang           .str_set_x:N = \l_stex_module_lang_str ,
1004     sig            .str_set_x:N = \l_stex_module_sig_str ,
1005     creators       .str_set_x:N = \l_stex_module_creators_str ,
1006     contributors   .str_set_x:N = \l_stex_module_contributors_str ,
1007     meta           .str_set_x:N = \l_stex_module_meta_str ,
1008     srccite        .str_set_x:N = \l_stex_module_srccite_str
1009 }
1010
1011 \cs_new_protected:Nn \__stex_modules_args:n {
1012     \str_clear:N \l_stex_module_title_str
1013     \str_clear:N \l_stex_module_ns_str
1014     \str_clear:N \l_stex_module_lang_str
1015     \str_clear:N \l_stex_module_sig_str
1016     \str_clear:N \l_stex_module_creators_str
1017     \str_clear:N \l_stex_module_contributors_str
1018     \str_clear:N \l_stex_module_meta_str
1019     \str_clear:N \l_stex_module_srccite_str
1020     \keys_set:nn { stex / module } { #1 }
1021 }
1022
1023 % module parameters here? In the body?
1024

```

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

```

1025 \cs_new_protected:Nn \stex_module_setup:nn {
1026     \str_set:Nx \l_stex_module_name_str { #2 }
1027     \__stex_modules_args:n { #1 }

```

First, we set up the name and namespace of the module.  
Are we in a nested module?

```

1028     \stex_if_in_module:TF {
1029         % Nested module
1030         \prop_get:NnN \l_stex_current_module_prop
1031             { ns } \l_stex_module_ns_str
1032         \str_set:Nx \l_stex_module_name_str {
1033             \prop_item:Nn \l_stex_current_module_prop

```

```

1034     { name } / \l_stex_module_name_str
1035   }
1036 }{
1037   % not nested:
1038   \str_if_empty:NT \l_stex_module_ns_str {
1039     \stex_modules_current_namespace:
1040     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1041     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1042       / { \l_stex_module_ns_str }
1043     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1044     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1045       \str_set:Nx \l_stex_module_ns_str {
1046         \stex_path_to_string:N \l_tmpa_seq
1047       }
1048     }
1049   }
1050 }

```

Next, we determine the language of the module:

```

1051 \str_if_empty:NT \l_stex_module_lang_str {
1052   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1053   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1054   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1055   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1056   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1057     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1058       inferred~from~file~name}
1059     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1060   }
1061 }
1062
1063 \str_if_empty:NF \l_stex_module_lang_str {
1064   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1065     \l_tmpa_str {
1066     \ltx@ifpackageloaded{babel}{
1067       \exp_args:Nx \selectlanguage { \l_tmpa_str }
1068     }{}
1069   } {
1070     \msg_error:nmx{stex}{error/unknownlanguage}{\l_tmpa_str}
1071   }
1072 }

```

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

```

1073 \str_if_empty:NTF \l_stex_module_sig_str {
1074   \str_clear:N \l_tmpa_str
1075   \seq_clear:N \l_tmpa_seq
1076   \tl_clear:N \l_tmpa_tl
1077   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
1078     name      = \l_stex_module_name_str ,
1079     ns        = \l_stex_module_ns_str ,
1080     imports   = \exp_not:o { \l_tmpa_seq } ,
1081     constants = \exp_not:o { \l_tmpa_seq } ,
1082     content   = \exp_not:o { \l_tmpa_tl } ,

```

```

1083     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1084     lang      = \l_stex_module_lang_str ,
1085     sig       = \l_stex_module_sig_str ,
1086     meta      = \l_stex_module_meta_str
1087   }
1088 }{
1089   \str_if_empty:NT \l_stex_module_lang_str {
1090     \msg_error:nnxx{stex}{error/siglanguage}{
1091       \l_stex_module_ns_str?\l_stex_module_name_str
1092     }\l_stex_module_sig_str}
1093   }
1094
1095   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1096   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1097   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1098   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1099   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1100   \str_set:Nx \l_tmpa_str {
1101     \stex_path_to_string:N \l_tmpa_seq /
1102     \l_tmpa_str . \l_stex_module_sig_str .tex
1103   }
1104   \IfFileExists \l_tmpa_str {
1105     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1106       \seq_clear:N \l_stex_all_modules_seq
1107       \prop_clear:N \l_stex_current_module_prop
1108       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1109       \input { \l_tmpa_str }
1110     }
1111   }{
1112     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1113   }
1114   \stex_activate_module:n {
1115     \l_stex_module_ns_str ? \l_stex_module_name_str
1116   }
1117   \prop_set_eq:Nc \l_stex_current_module_prop {
1118     c_stex_module_
1119     \l_stex_module_ns_str ?
1120     \l_stex_module_name_str
1121     _prop
1122   }
1123 }

```

We load the metatheory:

```

1124   \str_if_empty:NT \l_stex_module_meta_str {
1125     \str_set:Nx \l_stex_module_meta_str {
1126       \c_stex_metatheory_ns_str ? Metatheory
1127     }
1128   }
1129   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1130     \exp_args:Nx \stex_add_to_current_module:n {
1131       \stex_activate_module:n {\l_stex_module_meta_str}
1132     }
1133     \stex_activate_module:n {\l_stex_module_meta_str}
1134   }

```

1135 }

(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}

1136 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1137   \stex_reactivate_macro:N \STEXexport
1138   \stex_reactivate_macro:N \importmodule
1139   \stex_reactivate_macro:N \symdecl
1140   \stex_reactivate_macro:N \notation
1141   \stex_reactivate_macro:N \symdef
1142   \stex_module_setup:nn{#1}{#2}
1143
1144   \stex_debug:nn{modules}{
1145     New~module:\\
1146     Namespace:~\l_stex_module_ns_str\\
1147     Name:~\l_stex_module_name_str\\
1148     Language:~\l_stex_module_lang_str\\
1149     Signature:~\l_stex_module_sig_str\\
1150     Metatheory:~\l_stex_module_meta_str\\
1151     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1152   }
1153
1154   \seq_put_right:Nx \l_stex_all_modules_seq {
1155     \l_stex_module_ns_str ? \l_stex_module_name_str
1156   }
1157
1158   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1159     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1160
1161   \stex_if_smsmode:TF {
1162     \stex_smsmode_set_codes:
1163   } {
1164     \begin{stex_annotate_env} {theory} {
1165       \l_stex_module_ns_str ? \l_stex_module_name_str
1166     }
1167
1168     \stex_annotate_invisible:nnn{header}{} {
1169       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1170       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1171       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1172         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1173       }
1174     }
1175   }
1176   % TODO: Inherit metatheory for nested modules?
1177 }
1178 \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}

```

1179 \cs_new_protected:Nn \__stex_modules_end_module: {
1180   \str_set:Nx \l_tmpa_str {
1181     c_stex_module_
1182     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1183     \prop_item:Nn \l_stex_current_module_prop { name }
1184     _prop
1185   }
1186   %^^A \prop_new:c { \l_tmpa_str }
1187   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1188   \stex_debug:nn{modules}{Closing module~\prop_item:Nn \l_stex_current_module_prop { name }}
1189 }

```

(End definition for \\_\_stex\_modules\_end\_module:.)

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

```

1190 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1191 \NewDocumentEnvironment { @module } { 0{} m } {
1192   \par
1193   \__stex_modules_begin_module:nn{#1}{#2}
1194 } {
1195   \__stex_modules_end_module:
1196   \stex_if_smsmode:TF {
1197     \exp_args:Nx \stex_add_to_sms:n {
1198       \prop_gset_from_keyval:cn {
1199         c_stex_module_
1200         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1201         \prop_item:Nn \l_stex_current_module_prop { name }
1202         _prop
1203       } {
1204         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1205         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1206         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1207         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1208         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1209         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1210         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1211         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1212         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1213       }
1214     }
1215   }{
1216     \end{stex_annotate_env}
1217   }
1218 }

```

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

```

1219 \cs_if_exist:NTF \thesection {
1220   \newcounter{module}[section]
1221 }{
1222   \newcounter{module}
1223 }
1224
1225 \bool_if:NT \c_stex_showmods_bool {
1226   \latexml_if:F { \RequirePackage{mdframed} }

```



```

1227 }
1228
1229 \cs_new_protected:Nn \stex_modules_heading: {
1230   \stepcounter{module}
1231   \par
1232   \bool_if:NT \c_stex_showmods_bool {
1233     \noindent{\textbf{Module} ~
1234       \cs_if_exist:NT \thesection {\thesection.}
1235       \themodule ~ [\l_stex_module_name_str]
1236     }
1237     \str_if_empty:NTF \l_stex_module_title_str {
1238       }{
1239         \quad(\l_stex_module_title_str)\hfill
1240       }\par
1241     }
1242     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1243     % TODO
1244     \stex_ref_new_doc_target:n \l_stex_module_name_str
1245   }

```

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

Finally:

```

1246 \NewDocumentEnvironment { module } { 0{} m } {
1247   \bool_if:NT \c_stex_showmods_bool {
1248     \begin{mdframed}
1249   }
1250   \begin{@module}[#1]{#2}
1251     \stex_modules_heading:
1252   }{
1253     \end{@module}
1254     \bool_if:NT \c_stex_showmods_bool {
1255       \end{mdframed}
1256     }
1257   }

```

## 21.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1258 \NewDocumentCommand \STEXModule { m } {
1259   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1260   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1261   \tl_set:Nn \l_tmpa_tl {
1262     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1263   }
1264   \seq_map_inline:Nn \l_stex_all_modules_seq {
1265     \str_set:Nn \l_tmpb_str { ##1 }
1266     \str_if_eq:eeT { \l_tmpa_str } {
1267       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1268     } {
1269       \seq_map_break:n {
1270         \tl_set:Nn \l_tmpa_tl {
1271           \stex_invoke_module:n { ##1 }
1272         }

```

```

1273     }
1274   }
1275 }
1276 \l_tmpa_tl
1277 }
1278
1279 \cs_new_protected:Nn \stex_invoke_module:n {
1280   \stex_debug:nn{modules}{Invoking~module~#1}
1281   \peek_charcode_remove:NTF ! {
1282     \__stex_modules_invoke_uri:nN { #1 }
1283   } {
1284     \peek_charcode_remove:NTF ? {
1285       \__stex_modules_invoke_symbol:nn { #1 }
1286     } {
1287       \msg_error:nnx{stex}{error/syntax}{
1288         ?~or~!~expected~after~
1289         \c_backslash_str STEXModule{#1}
1290       }
1291     }
1292   }
1293 }
1294
1295 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1296   \str_set:Nn #2 { #1 }
1297 }
1298
1299 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1300   \stex_invoke_symbol:n{#1?#2}
1301 }

```

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

`\stex_activate_module:n`

```

1302 \cs_new_protected:Nn \stex_activate_module:n {
1303   \stex_debug:nn{modules}{Activating~module~#1}
1304   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1305     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1306     \prop_item:cn { c_stex_module_#1_prop } { content }
1307   }
1308 }

```

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

```

1309 </package>

```

## Chapter 22

# STEX -Module Inheritance Implementation

```
1310 <*package>
1311
1312 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1313
```

### 22.1 SMS Mode

```
1314 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1315 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1316 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1317 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1318
1319 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1320   \makeatletter
1321   \makeatother
1322   \ExplSyntaxOn
1323   \ExplSyntaxOff
1324 }
1325
1326 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1327   \symdef
1328   \importmodule
1329   \notation
1330   \symdecl
1331   \STEXexport
1332 }
1333
1334 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1335   \tl_to_str:n {
1336     module,
1337     @module
```

```

1338 }
1339 }

```

(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
1340 \bool_new:N \g__stex_smsmode_bool
1341 \bool_set_false:N \g__stex_smsmode_bool
1342 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1343   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1344 }

```

(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
1345 \bool_new:N \g__stex_smsmode_catcode_bool
1346 \bool_set_false:N \g__stex_smsmode_catcode_bool
1347 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1348   \bool_if:NTF \g__stex_smsmode_catcode_bool
1349   \prg_return_true: \prg_return_false:
1350 }

```

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

```

\stex_smsmode_set_codes:
1351 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1352   \stex_if_smsmode:T {
1353     \__stex_smsmode_if_catcodes:F {
1354       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1355       \exp_after:wN \char_gset_active_eq:NN
1356       \c_backslash_str \__stex_smsmode_cs:
1357       \tex_global:D \char_set_catcode_active:N \
1358       \tex_global:D \char_set_catcode_other:N $
1359       \tex_global:D \char_set_catcode_other:N ^
1360       \tex_global:D \char_set_catcode_other:N _
1361       \tex_global:D \char_set_catcode_other:N &
1362       \tex_global:D \char_set_catcode_other:N ##
1363     }
1364   }
1365 } \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.
1366 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1367   \__stex_smsmode_if_catcodes:T {
1368     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1369     \exp_after:wN \tex_global:D \exp_after:wN
1370     \char_set_catcode_escape:N \c_backslash_str
1371     \tex_global:D \char_set_catcode_math_toggle:N $
1372     \tex_global:D \char_set_catcode_math_superscript:N ^
1373     \tex_global:D \char_set_catcode_math_subscript:N _
1374     \tex_global:D \char_set_catcode_alignment:N &
1375     \tex_global:D \char_set_catcode_parameter:N ##
1376   }
1377 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1378 \cs_new_protected:Nn \stex_in_smsmode:nn {
1379   \vbox_set:Nn \l_tmpa_box {
1380     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1381     \bool_gset_true:N \g__stex_smsmode_bool
1382     \stex_smsmode_set_codes:
1383     #2
1384     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1385     \stex_if_smsmode:F {
1386       \__stex_smsmode_unset_codes:
1387     }
1388   }
1389   \box_clear:N \l_tmpa_box
1390 }

```

(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:

```

1391 \cs_new_protected:Nn \_stex_smsmode_cs: {
1392   \str_clear:N \l_tmpa_str
1393   \peek_analysis_map_inline:n {
1394     % #1: token (one expansion)
1395     % #2: charcode
1396     % #3 catcode
1397     \token_if_eq_charcode:NNTF ##3 B {
1398       % token is a letter
1399       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1400     } {
1401       \str_if_empty:NTF \l_tmpa_str {
1402         % we don't allow (or need) single non-letter CSs
1403         % for now
1404         \peek_analysis_map_break:
1405       }{
1406         \str_if_eq:onTF \l_tmpa_str { begin } {
1407           \peek_analysis_map_break:n {
1408             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1409           }
1410         } {
1411           \str_if_eq:onTF \l_tmpa_str { end } {
1412             \peek_analysis_map_break:n {
1413               \exp_after:wN \_stex_smsmode_checkend:n ##1
1414             }
1415           } {
1416             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1417             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1418               \g_stex_smsmode_allowedmacros_tl
1419               { \use:c{\l_tmpa_str} } {
1420               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1421               \peek_analysis_map_break:n {
1422                 \exp_after:wN \l_tmpa_tl ##1
1423               }

```

```

1424     } {
1425         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1426         \g_stex_smsmode_allowedmacros_escape_tl
1427         { \use:c{\l_tmpa_str} } {
1428             \__stex_smsmode_unset_codes:
1429             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1430             % TODO \__stex_smsmode_rescan_cs:
1431             % \int_compare:nNnTF {##2} = {92} {
1432             %     \peek_analysis_map_break:n {
1433             %         \__stex_smsmode_unset_codes:
1434             %         \__stex_smsmode_rescan_cs:
1435             %     }
1436             % } {
1437             %     \peek_analysis_map_break:n {
1438             %         \exp_after:wN \l_tmpa_tl ##1
1439             %     }
1440             % }
1441             } {
1442                 \int_compare:nNnTF {##2} = {92} {
1443                     \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1444                 }{
1445                     \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1446                 }
1447             }
1448         }
1449     }
1450 }
1451 }
1452 }
1453 }
1454 }

```

(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:

```

1455 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1456     \str_clear:N \l_tmpb_str
1457     \peek_analysis_map_inline:n {
1458         \token_if_eq_charcode:NNTF ##3 B {
1459             % token is a letter
1460             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1461         } {
1462             \peek_analysis_map_break:n {
1463                 \exp_after:wN \use:c \exp_after:wN {
1464                     \exp_after:wN \l_tmpa_str\exp_after:wN
1465                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1466             }
1467         }
1468     }
1469 }

```

(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.

```

1470 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1471   \str_set:Nn \l_tmpa_str { #1 }
1472   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1473     \__stex_smsmode_unset_codes:
1474     \begin{#1}
1475   }
1476 }
```

(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.

```

1477 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1478   \str_set:Nn \l_tmpa_str { #1 }
1479   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1480     \end{#1}
1481   }
1482 }
```

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

## 22.2 Inheritance

1483 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1484 \cs_new_protected:Nn \stex_import_module_uri:nn {
1485   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1486   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1487
1488   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1489   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1490   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1491
1492   \stex_modules_current_namespace:
1493   \bool_lazy_all:nTF {
1494     {\str_if_empty_p:N \l__stex_importmodule_archive_str}
1495     {\str_if_empty_p:N \l__stex_importmodule_path_str}
1496     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
1497   }{
1498     \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
1499     \str_set_eq:NN \l_stex_module_ns
1500   }{
1501     \str_if_empty:NT \l__stex_importmodule_archive_str {
1502       \prop_if_empty:NF \l_stex_current_repository_prop {
1503         \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
1504       }
1505     }
1506     \str_if_empty:NTF \l__stex_importmodule_archive_str {
1507       \str_if_empty:NF \l__stex_importmodule_path_str {
1508         \str_set:Nx \l_stex_module_ns_str {
1509           \l_stex_module_ns_str / \l__stex_importmodule_path_str
1510         }
1511       }
1512     }
```

```

1512   }{
1513     \stex_require_repository:n \l__stex_importmodule_archive_str
1514     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str _manifest_prop } { ns
1515       \l_stex_module_ns_str
1516     \str_if_empty:NF \l__stex_importmodule_path_str {
1517       \str_set:Nx \l_stex_module_ns_str {
1518         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1519       }
1520     }
1521   }
1522 }
1523 }

```

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

<code>\l__stex_importmodule_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l__stex_importmodule_archive_str</code>	1524 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l__stex_importmodule_path_str</code>	1525 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l__stex_importmodule_file_str</code>	1526 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1527 <code>\str_new:N \g__stex_importmodule_file_str</code>

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

```

\stex_import_require_module:nnnnn    {<ns>} {<archive-ID>} {<path>} {<name>}
1528 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1529   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1530
1531     % archive
1532     \str_set:Nx \l_tmpa_str { #2 }
1533     \str_if_empty:NTF \l_tmpa_str {
1534       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1535     } {
1536       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1537       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1538       \seq_put_right:Nn \l_tmpa_seq { source }
1539     }
1540
1541     % path
1542     \str_set:Nx \l_tmpb_str { #3 }
1543     \str_if_empty:NTF \l_tmpb_str {
1544       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1545
1546       \ltx@ifpackageloaded{babel} {
1547         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1548           { \language } \l_tmpb_str {
1549           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1550         }
1551       } {
1552         \str_clear:N \l_tmpb_str
1553       }
1554
1555       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1556       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1557         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```



```

1558 }{
1559   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1560   \IfFileExists{ \l_tmpa_str.tex }{
1561     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1562   }{
1563     % try english as default
1564     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1565     \IfFileExists{ \l_tmpa_str.en.tex }{
1566       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1567     }{
1568       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1569     }
1570   }
1571 }
1572
1573 } {
1574   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1575   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1576
1577   \ltx@ifpackageloaded{babel} {
1578     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1579       { \language } \l_tmpb_str {
1580       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1581     }
1582   } {
1583     \str_clear:N \l_tmpb_str
1584   }
1585
1586   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1587
1588   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1589   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1590     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1591   }{
1592     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1593     \IfFileExists{ \l_tmpa_str/#4.tex }{
1594       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1595     }{
1596       % try english as default
1597       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1598       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1599         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1600       }{
1601         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1602         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1603           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1604         }{
1605           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1606           \IfFileExists{ \l_tmpa_str.tex }{
1607             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1608           }{
1609             % try english as default
1610             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1611             \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1612         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1613     }{
1614         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1615     }
1616 }
1617 }
1618 }
1619 }
1620 }
1621 }
1622
1623 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1624 \seq_clear:N \g_stex_modules_in_file_seq
1625 % \exp_args:Nnx \use:nn {
1626     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1627         \seq_clear:N \l_stex_all_modules_seq
1628         \prop_clear:N \l_stex_current_module_prop
1629         \str_set:Nx \l_tmpb_str { #2 }
1630         \str_if_empty:NF \l_tmpb_str {
1631             \stex_set_current_repository:n { #2 }
1632         }
1633         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1634         \input { \g__stex_importmodule_file_str }
1635     }
1636 % }{
1637
1638 % }
1639 \prop_gput:Noo \g_stex_module_files_prop
1640 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1641 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1642
1643 \stex_if_module_exists:nF { #1 ? #4 } {
1644     \msg_error:nnx{stex}{error/unknownmodule}{
1645         #1?#4~(in~file~\g__stex_importmodule_file_str)
1646     }
1647 }
1648 }
1649 \stex_activate_module:n { #1 ? #4 }
1650 }

```

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

## `\importmodule`

```

1651 \NewDocumentCommand \importmodule { 0{} m } {
1652     \stex_import_module_uri:nn { #1 } { #2 }
1653     \stex_debug:nn{modules}{Importing~module:~
1654         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1655     }
1656     \stex_if_smsmode:F {
1657         \stex_import_require_module:nnnn
1658         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1659         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1660         \stex_annotate_invisible:nnn
1661         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}

```

```

1662 }
1663 \exp_args:Nx \stex_add_to_current_module:n {
1664   \stex_import_require_module:nnnn
1665   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1666   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1667 }
1668 \exp_args:Nx \stex_add_import_to_current_module:n {
1669   \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1670 }
1671 \stex_smsmode_set_codes:
1672 }
1673 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

### `\usemodule`

```

1674 \NewDocumentCommand \usemodule { 0{} m } {
1675   \stex_if_smsmode:F {
1676     \stex_import_module_uri:nn { #1 } { #2 }
1677     \stex_import_require_module:nnnn
1678     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1679     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1680     \stex_annotate_invisible:nnn
1681     {usemodule} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1682   }
1683   \stex_smsmode_set_codes:
1684 }

```

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

```

1685 \endpackage

```

## Chapter 23

# STEX -Symbols Implementation

```
1686 <*package>
1687
1688 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1689
1690
Warnings and error messages
```

### 23.1 Symbol Declarations

```
1691 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1692 \seq_new:N \l_stex_all_symbols_seq

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

\STEXsymbol

1693 \NewDocumentCommand \STEXsymbol { m } {
1694   \stex_get_symbol:n { #1 }
1695   \exp_args:No
1696   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1697 }

(End definition for \STEXsymbol. This function is documented on page 27.)
symdecl arguments:

1698 \keys_define:nn { stex / symdecl } {
1699   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1700   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1701   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1702   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1703   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1704   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1705   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1706   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1707 }
```

```

1708
1709 \bool_new:N \l_stex_symdecl_make_macro_bool
1710
1711 \cs_new_protected:Nn \__stex_symdecl_args:n {
1712   \str_clear:N \l_stex_symdecl_name_str
1713   \str_clear:N \l_stex_symdecl_args_str
1714   \bool_set_false:N \l_stex_symdecl_local_bool
1715   \tl_clear:N \l_stex_symdecl_type_tl
1716   \tl_clear:N \l_stex_symdecl_definiens_tl
1717
1718   \keys_set:nn { stex / symdecl } { #1 }
1719 }

```

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

```

1720
1721 \NewDocumentCommand \symdecl { s O{} m } {
1722   \__stex_symdecl_args:n { #2 }
1723   \IfBooleanTF #1 {
1724     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1725   } {
1726     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1727   }
1728   \stex_symdecl_do:n { #3 }
1729   \stex_smsmode_set_codes:
1730 }
1731 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1732 \cs_new_protected:Nn \stex_symdecl_do:n {
1733   \stex_if_in_module:F {
1734     % TODO throw error? some default namespace?
1735   }
1736
1737   \str_if_empty:NT \l_stex_symdecl_name_str {
1738     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1739   }
1740
1741   \prop_if_exist:cT { g_stex_symdecl_
1742     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1743     \prop_item:Nn \l_stex_current_module_prop {name} ?
1744     \l_stex_symdecl_name_str
1745     _prop
1746   }{
1747     % TODO throw error (beware of circular dependencies)
1748   }
1749
1750   \prop_clear:N \l_tmpa_prop
1751   \prop_put:Nnx \l_tmpa_prop { module } {
1752     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1753     \prop_item:Nn \l_stex_current_module_prop {name}
1754   }

```

```

1755 \seq_clear:N \l_tmpa_seq
1756 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1757 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1758 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1759 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1760
1761 \exp_args:No \stex_add_constant_to_current_module:n {
1762   \l_stex_symdecl_name_str
1763 }
1764
1765 % arity/args
1766 \int_zero:N \l_tmpb_int
1767
1768 \bool_set_true:N \l_tmpa_bool
1769 \str_map_inline:Nn \l_stex_symdecl_args_str {
1770   \token_case_meaning:NnF ##1 {
1771     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1772     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1773     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1774     {\tl_to_str:n a} {
1775       \bool_set_false:N \l_tmpa_bool
1776       \int_incr:N \l_tmpb_int
1777     }
1778     {\tl_to_str:n B} {
1779       \bool_set_false:N \l_tmpa_bool
1780       \int_incr:N \l_tmpb_int
1781     }
1782   }{
1783     \msg_set:nnn{stex}{error/wrongargs}{
1784       args~value~in~symbol~declaration~for~
1785       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1786       \prop_item:Nn \l_stex_current_module_prop {name} ?
1787       \l_stex_symdecl_name_str ~
1788       needs~to~be~
1789       i,~a,~b~or~B,~but~##1~given
1790     }
1791     \msg_error:nn{stex}{error/wrongargs}
1792   }
1793 }
1794 \bool_if:NTF \l_tmpa_bool {
1795   % possibly numeric
1796   \str_if_empty:NTF \l_stex_symdecl_args_str {
1797     \prop_put:Nnn \l_tmpa_prop { args } {}
1798     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1799   }{
1800     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1801     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1802     \str_clear:N \l_tmpa_str
1803     \int_step_inline:nn \l_tmpa_int {
1804       \str_put_right:Nn \l_tmpa_str i
1805     }
1806     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1807   }
1808 } {

```

```

1809 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1810 \prop_put:Nnx \l_tmpa_prop { arity }
1811 { \str_count:N \l_stex_symdecl_args_str }
1812 }
1813 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1814
1815
1816 % semantic macro
1817
1818 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1819 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1820 \prop_item:Nn \l_tmpa_prop { module } ?
1821 \prop_item:Nn \l_tmpa_prop { name }
1822 } }
1823
1824 \bool_if:NF \l_stex_symdecl_local_bool {
1825 \exp_args:Nx \stex_add_to_current_module:n {
1826 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1827 \prop_item:Nn \l_tmpa_prop { module } ?
1828 \prop_item:Nn \l_tmpa_prop { name }
1829 } }
1830 }
1831 }
1832 }
1833
1834 % add to all symbols
1835
1836 \bool_if:NF \l_stex_symdecl_local_bool {
1837 \exp_args:Nx \stex_add_to_current_module:n {
1838 \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1839 \prop_item:Nn \l_tmpa_prop { module } ?
1840 \prop_item:Nn \l_tmpa_prop { name }
1841 }
1842 }
1843 }
1844
1845 \stex_debug:nn{symbols}{New~symbol:~
1846 \prop_item:Nn \l_tmpa_prop { module } ?
1847 \prop_item:Nn \l_tmpa_prop { name } ^^J
1848 Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1849 Args:~\prop_item:Nn \l_tmpa_prop { args }
1850 }
1851
1852 % circular dependencies require this:
1853
1854 \prop_if_exist:cF {
1855 g_stex_symdecl_
1856 \prop_item:Nn \l_tmpa_prop { module } ?
1857 \prop_item:Nn \l_tmpa_prop { name }
1858 _prop
1859 } {
1860 \prop_gset_eq:cN {
1861 g_stex_symdecl_
1862 \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1863     \prop_item:Nn \l_tmpa_prop { name }
1864     _prop
1865   } \l_tmpa_prop
1866 }
1867
1868 \stex_if_smsmode:TF {
1869   \bool_if:NF \l_stex_symdecl_local_bool {
1870     \exp_args:Nx \stex_add_to_sms:n {
1871       \prop_gset_from_keyval:cn {
1872         g_stex_symdecl_
1873         \prop_item:Nn \l_tmpa_prop { module } ?
1874         \prop_item:Nn \l_tmpa_prop { name }
1875         _prop
1876       } {
1877         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1878         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1879         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1880         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1881         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1882         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1883         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1884         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1885       }
1886       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1887         \prop_item:Nn \l_tmpa_prop { module } ?
1888         \prop_item:Nn \l_tmpa_prop { name }
1889       }
1890     }
1891   }
1892 }{
1893   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1894     \prop_item:Nn \l_tmpa_prop { module } ?
1895     \prop_item:Nn \l_tmpa_prop { name }
1896   }
1897   \stex_if_do_html:T {
1898     \stex_annotate_invisible:nnn {symdecl} {
1899       \prop_item:Nn \l_tmpa_prop { module } ?
1900       \prop_item:Nn \l_tmpa_prop { name }
1901     } {
1902       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
1903       \stex_annotate_invisible:nnn{args}{}}{
1904         \prop_item:Nn \l_tmpa_prop { args }
1905       }
1906       \stex_annotate_invisible:nnn{macroname}{}{#1}
1907       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1908         \stex_annotate_invisible:nnn{definiens}{}
1909         {\l_stex_symdecl_definiens_tl$}
1910       }
1911     }
1912   }
1913 }
1914 }

```

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



`\stex_get_symbol:n`

```
1915 \str_new:N \l_stex_get_symbol_uri_str
1916
1917 \cs_new_protected:Nn \stex_get_symbol:n {
1918   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1919     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1920   }{
1921     % argument is a string
1922     % is it a command name?
1923     \cs_if_exist:cTF { #1 }{
1924       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1925       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1926       \str_if_empty:NTF \l_tmpa_str {
1927         \exp_args:Nx \cs_if_eq:NNTF {
1928           \tl_head:N \l_tmpa_tl
1929         } \stex_invoke_symbol:n {
1930           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1931         }{
1932           \__stex_symdecl_get_symbol_from_string:n { #1 }
1933         }
1934       } {
1935         \__stex_symdecl_get_symbol_from_string:n { #1 }
1936       }
1937     }{
1938       % argument is not a command name
1939       \__stex_symdecl_get_symbol_from_string:n { #1 }
1940       % \l_stex_all_symbols_seq
1941     }
1942   }
1943 }
1944
1945 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1946   \str_set:Nn \l_tmpa_str { #1 }
1947   \bool_set_false:N \l_tmpa_bool
1948   \stex_if_in_module:T {
1949     \prop_get:NnN \l_stex_current_module_prop
1950     { constants } \l_tmpa_seq
1951     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1952       \bool_set_true:N \l_tmpa_bool
1953       \str_set:Nx \l_stex_get_symbol_uri_str {
1954         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1955         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1956       }
1957     }
1958   }
1959   \bool_if:NF \l_tmpa_bool {
1960     \tl_set:Nn \l_tmpa_tl {
1961       \msg_set:nnn{stex}{error/unknownsymbol}{
1962         No~symbol~#1~found!
1963       }
1964     }
1965     \msg_error:nn{stex}{error/unknownsymbol}
1966   }
1967   \str_set:Nn \l_tmpa_str { #1 }
1968   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1968 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1969   \str_set:Nn \l_tmpb_str { ##1 }
1970   \str_if_eq:eeT { \l_tmpa_str } {
1971     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1972   } {
1973     \seq_map_break:n {
1974       \tl_set:Nn \l_tmpa_tl {
1975         \str_set:Nn \l_stex_get_symbol_uri_str {
1976           ##1
1977         }
1978       }
1979     }
1980   }
1981 }
1982 \l_tmpa_tl
1983 }
1984 }
1985
1986 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1987   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1988   { \tl_tail:N \l_tmpa_tl }
1989   \tl_if_single:NTF \l_tmpa_tl {
1990     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1991       \exp_after:wN \str_set:Nn \exp_after:wN
1992       \l_stex_get_symbol_uri_str \l_tmpa_tl
1993     }{
1994       % TODO
1995       % tail is not a single group
1996     }
1997   }{
1998     % TODO
1999     % tail is not a single group
2000   }
2001 }

```

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

## 23.2 Notations

```

2002 <@@=stex_notation>
2003 notation arguments:
2004 \keys_define:nn { stex / notation } {
2005   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2006   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2007   prec .str_set_x:N = \l__stex_notation_prec_str ,
2008   op .tl_set:N = \l__stex_notation_op_tl ,
2009   unknown .code:n = \str_set:Nx
2010     \l__stex_notation_variant_str \l_keys_key_str
2011 }
2012
2013 \cs_new_protected:Nn \__stex_notation_args:n {
2014   \str_clear:N \l__stex_notation_lang_str
2015   \str_clear:N \l__stex_notation_variant_str

```

```

2015 \str_clear:N \l__stex_notation_prec_str
2016 \tl_clear:N \l__stex_notation_op_tl
2017
2018 \keys_set:nn { stex / notation } { #1 }
2019 }

```

## **\notation**

```

2020 \NewDocumentCommand \notation { 0{ } m } {
2021   \__stex_notation_args:n { #1 }
2022   \tl_clear:N \l_stex_symdecl_definiens_tl
2023   \stex_get_symbol:n { #2 }
2024   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2025 }
2026 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

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

```

2027 \cs_new_protected:Nn \stex_notation_do:nn {
2028   \prop_set_eq:Nc \l_tmpa_prop {
2029     g_stex_symdecl_ #1 _prop
2030   }
2031
2032   \prop_clear:N \l_tmpb_prop
2033   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2034   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2035   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2036
2037   % precedences
2038   \seq_clear:N \l_tmpb_seq
2039   \exp_args:NNno
2040   \str_if_empty:NTF \l__stex_notation_prec_str {
2041     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2042     \int_compare:nNnTF \l_tmpa_str = 0 {
2043       \exp_args:NNnx
2044       \prop_put:Nno \l_tmpb_prop { opprec }
2045       { \neginfprec }
2046     }{
2047       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2048     }
2049   } {
2050     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2051       \exp_args:NNnx
2052       \prop_put:Nno \l_tmpb_prop { opprec }
2053       { \neginfprec }
2054       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2055       \int_step_inline:nn { \l_tmpa_str } {
2056         \exp_args:NNx
2057         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2058       }
2059     }{
2060       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2061       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2062         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2063         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

2064         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2065         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2066         \seq_map_inline:Nn \l_tmpa_seq {
2067             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2068         }
2069     }
2070     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2071 }{
2072     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2073     \int_compare:nNnTF \l_tmpa_str = 0 {
2074         \exp_args:NNnx
2075         \prop_put:Nno \l_tmpb_prop { opprec }
2076         { \infprec }
2077     }{
2078         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2079     }
2080 }
2081 }
2082 }
2083
2084 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2085 \int_step_inline:nn { \l_tmpa_str } {
2086     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2087         \exp_args:NNx
2088         \seq_put_right:Nn \l_tmpb_seq {
2089             \prop_item:Nn \l_tmpb_prop { opprec }
2090         }
2091     }
2092 }
2093
2094 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2095 \tl_clear:N \l_tmpa_tl
2096
2097 \int_compare:nNnTF \l_tmpa_str = 0 {
2098     \exp_args:NNe
2099     \cs_set:Npn \l__stex_notation_macrocode_cs {
2100         \_stex_term_math_oms:nnnn { #1 }
2101         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2102         { \prop_item:Nn \l_tmpb_prop { opprec } }
2103         { \exp_not:n { #2 } }
2104     }
2105     \__stex_notation_final:
2106 }{
2107     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2108     \str_if_in:NnTF \l_tmpb_str b {
2109         \exp_args:Nne \use:nn
2110         {
2111             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2112             \cs_set:Npn \l_tmpa_str { {
2113                 \_stex_term_math_omb:nnnn { #1 }
2114                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2115                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2116                 { \exp_not:n { #2 } }
2117             }}

```

```

2118   }{
2119     \str_if_in:NnTF \l_tmpb_str B {
2120       \exp_args:Nne \use:nn
2121       {
2122         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2123         \cs_set:Npn \l_tmpa_str } { {
2124           \stex_term_math_omb:nnnn { #1 }
2125           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2126           { \prop_item:Nn \l_tmpb_prop { opprec } }
2127           { \exp_not:n { #2 } }
2128         } }
2129       }{
2130         \exp_args:Nne \use:nn
2131         {
2132           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2133           \cs_set:Npn \l_tmpa_str } { {
2134             \stex_term_math_oma:nnnn { #1 }
2135             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2136             { \prop_item:Nn \l_tmpb_prop { opprec } }
2137             { \exp_not:n { #2 } }
2138           } }
2139         }
2140       }
2141
2142       \int_zero:N \l_tmpa_int
2143       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2144       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2145       \__stex_notation_arguments:
2146     }
2147   }

```

(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

```

2148   \cs_new_protected:Nn \__stex_notation_arguments: {
2149     \int_incr:N \l_tmpa_int
2150     \str_if_empty:NnTF \l_tmpa_str {
2151       \__stex_notation_final:
2152     }{
2153       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2154       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2155       \str_if_eq:VnTF \l_tmpb_str a {
2156         \__stex_notation_argument_assoc:n
2157       }{
2158         \str_if_eq:VnTF \l_tmpb_str B {
2159           \__stex_notation_argument_assoc:n
2160         }{
2161           \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2162           \tl_put_right:Nx \l_tmpa_tl {
2163             { \stex_term_math_arg:nnn
2164               { \int_use:N \l_tmpa_int }
2165               { \l_tmpb_str }
2166               { ####\int_use:N \l_tmpa_int }
2167             }

```

```

2168     }
2169     \__stex_notation_arguments:
2170   }
2171 }
2172 }
2173 }

```

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

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

```

2174 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2175   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2176   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2177   \tl_put_right:Nx \l_tmpa_tl {
2178     { \stex_term_math_assoc_arg:nnnn
2179       { \int_use:N \l_tmpa_int }
2180       { \l_tmpb_str }
2181       \exp_args:No \exp_not:n
2182       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2183       { ####\int_use:N \l_tmpa_int }
2184     }
2185   }
2186   \__stex_notation_arguments:
2187 }

```

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

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

```

2188 \cs_new_protected:Nn \__stex_notation_final: {
2189   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2190   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2191   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2192   \exp_args:Nne \use:nn
2193   {
2194     \cs_generate_from_arg_count:cNnn {
2195       stex_notation_ \l_tmpa_str \c_hash_str
2196       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2197       _cs
2198     }
2199     \cs_gset:Npn \l_tmpb_str } { {
2200       \exp_after:wN \exp_after:wN \exp_after:wN
2201       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2202       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2203     } }
2204
2205     \tl_if_empty:NF \l__stex_notation_op_tl {
2206       \cs_gset:cpx {
2207         stex_op_notation_ \l_tmpa_str \c_hash_str
2208         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2209         _cs
2210       } {
2211         \stex_term_oms:nnn {
2212           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2213           \l__stex_notation_lang_str

```

```

2214     }{
2215         \l_tmpa_str
2216     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2217 }
2218 }
2219
2220
2221
2222 \stex_debug:nn{symbols}{
2223     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2224     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2225     Operator~precedence:~
2226     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2227     Argument~precedences:~
2228     \seq_use:Nn \l_tmpa_seq {,~}^^J
2229     Notation: \cs_meaning:c {
2230         stex_notation_ \l_tmpa_str \c_hash_str
2231         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2232         _cs
2233     }
2234 }
2235
2236 \prop_gset_eq:cN {
2237     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2238     \c_hash_str \l__stex_notation_lang_str _prop
2239 } \l_tmpb_prop
2240
2241 \exp_args:Nx
2242 \stex_add_to_current_module:n {
2243     \prop_get:cnN {
2244         g_stex_symdecl_
2245         \prop_item:Nn \l_tmpb_prop { symbol }
2246         _prop
2247     } { notations } \exp_not:N \l_tmpa_seq
2248     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2249         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2250     }
2251     \prop_put:cno {
2252         g_stex_symdecl_
2253         \prop_item:Nn \l_tmpb_prop { symbol }
2254         _prop
2255     } { notations } \exp_not:N \l_tmpa_seq
2256 }
2257
2258 \stex_if_smsmode:TF {
2259     \stex_smsmode_set_codes:
2260     \exp_args:Nx \stex_add_to_sms:n {
2261         \prop_gset_from_keyval:cn {
2262             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2263             \c_hash_str \l__stex_notation_lang_str _prop
2264         } {
2265             symbol = \prop_item:Nn \l_tmpb_prop { symbol } ,
2266             language = \prop_item:Nn \l_tmpb_prop { language } ,
2267             variant = \prop_item:Nn \l_tmpb_prop { variant } ,

```

```

2268         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2269         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2270     }
2271 }
2272 }{
2273   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2274   \seq_put_right:Nx \l_tmpa_seq {
2275     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2276   }
2277   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2278   \prop_set_eq:cN {
2279     g_stex_symdecl_ \l_tmpa_str _prop
2280   } \l_tmpa_prop
2281
2282   % HTML annotations
2283   \stex_if_do_html:T {
2284     \stex_annotate_invisible:nnn { notation }
2285     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2286       \stex_annotate_invisible:nnn { notationfragment }
2287       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2288       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2289       \stex_annotate_invisible:nnn { precedence }
2290       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2291         \seq_use:Nn \l_tmpa_seq { x }
2292       }{}
2293
2294       \int_zero:N \l_tmpa_int
2295       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2296       \tl_clear:N \l_tmpa_tl
2297       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2298         \int_incr:N \l_tmpa_int
2299         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2300         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2301         \str_if_eq:VnTF \l_tmpb_str a {
2302           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2303             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2304             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2305           } }
2306         }{
2307           \str_if_eq:VnTF \l_tmpb_str B {
2308             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2309               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2310               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2311             } }
2312           }{
2313             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2314               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2315             } }
2316           }
2317         }
2318       }
2319       \stex_annotate_invisible:nnn { notationcomp }{}{
2320         $ \exp_args:Nno \use:nn { \use:c {
2321           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```



```

2322         \c_hash_str \l__stex_notation_variant_str
2323         \c_hash_str \l__stex_notation_lang_str _cs
2324     } } { \l_tmpa_tl } $
2325 }
2326 }
2327 }
2328 }
2329 }

```

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

**\symdef**

```

2330 \keys_define:nn { stex / symdef } {
2331   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2332   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2333   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2334   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2335   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2336   op        .tl_set:N   = \l__stex_notation_op_tl ,
2337   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2338   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2339   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2340   unknown   .code:n     = \str_set:Nx
2341             \l__stex_notation_variant_str \l_keys_key_str
2342 }
2343
2344 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2345   \str_clear:N \l_stex_symdecl_name_str
2346   \str_clear:N \l_stex_symdecl_args_str
2347   \bool_set_false:N \l_stex_symdecl_local_bool
2348   \tl_clear:N \l_stex_symdecl_type_tl
2349   \tl_clear:N \l_stex_symdecl_definiens_tl
2350   \str_clear:N \l__stex_notation_lang_str
2351   \str_clear:N \l__stex_notation_variant_str
2352   \str_clear:N \l__stex_notation_prec_str
2353   \tl_clear:N \l__stex_notation_op_tl
2354
2355   \keys_set:nn { stex / symdef } { #1 }
2356 }
2357
2358 \NewDocumentCommand \symdef { 0{} m } {
2359   \_stex_notation_symdef_args:n { #1 }
2360   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2361   \stex_symdecl_do:n { #2 }
2362   \exp_args:Nx \stex_notation_do:nn {
2363     \prop_item:Nn \l_tmpa_prop { module } ?
2364     \prop_item:Nn \l_tmpa_prop { name }
2365   }
2366 }
2367 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2368 \</package>

```

## Chapter 24

# STEX -Terms Implementation

```
2369 <*package>
2370
2371 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2372
2373 <@@=stex_terms>
2374
2375 Warnings and error messages
2376 \msg_new:nnn{stex}{error/nonotation}{
2377   Symbol~#1~invoked,~but~has~no~notation~#2!
2378 }
2379 \msg_new:nnn{stex}{error/notationarg}{
2380   Error~in~parsing~notation~#1
2381 }
2382 \msg_new:nnn{stex}{error/noop}{
2383   Symbol~#1~has~no~operator~notation~for~notation~#2
2384 }
2385
```

### 24.1 Symbol Invocations

Arguments:

```
2384 \keys_define:nn { stex / terms } {
2385   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2386   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2387   unknown .code:n = \str_set:Nx
2388     \l__stex_terms_variant_str \l_keys_key_str
2389 }
2390
2391 \cs_new_protected:Nn \__stex_terms_args:n {
2392   \str_clear:N \l__stex_terms_lang_str
2393   \str_clear:N \l__stex_terms_variant_str
2394   \str_clear:N \l__stex_terms_prec_str
2395   \tl_clear:N \l__stex_terms_op_tl
2396
2397   \keys_set:nn { stex / terms } { #1 }
```

2398 }

**\stex\_invoke\_symbol:n** Invokes a semantic macro

```
2399 \cs_new_protected:Nn \stex_invoke_symbol:n {
2400   \if_mode_math:
2401     \exp_after:wN \__stex_terms_invoke_math:n
2402   \else:
2403     \exp_after:wN \__stex_terms_invoke_text:n
2404   \fi: { #1 }
2405 }
```

(End definition for \stex\_invoke\_symbol:n. This function is documented on page 27.)

**\\_\_stex\_terms\_invoke\_math:n**

```
2406 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2407   \peek_charcode_remove:NTF ! {
2408     \peek_charcode:NTF [ {
2409       \__stex_terms_invoke_op:nw { #1 }
2410     }{
2411       \peek_charcode_remove:NTF ! {
2412         \peek_charcode:NTF [ {
2413           \__stex_terms_invoke_op_custom:nw
2414         }{
2415           % TODO throw error
2416         }
2417       }{
2418         \__stex_terms_invoke_op:nw { #1 } []
2419       }
2420     }
2421   }{
2422     \peek_charcode_remove:NTF * {
2423       \__stex_terms_invoke_text:n { #1 }
2424     }{
2425       \peek_charcode:NTF [ {
2426         \__stex_terms_invoke_math:nw { #1 }
2427       }{
2428         \__stex_terms_invoke_math:nw { #1 } []
2429       }
2430     }
2431   }
2432 }
```

(End definition for \\_\_stex\_terms\_invoke\_math:n.)

**\\_\_stex\_terms\_invoke\_op\_custom:nw**

```
2433 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2434   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2435     \stex_highlight_term:nn{#1}{#2}
2436   }
2437 }
```

(End definition for \\_\_stex\_terms\_invoke\_op\_custom:nw.)

\\_stex\_terms\_invoke\_op:nw

```

2438 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2439   \_stex_terms_args:n { #2 }
2440   \cs_if_exist:cTF {
2441     stex_op_notation_ #1 \c_hash_str
2442     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2443   }{
2444     \csname stex_op_notation_ #1 \c_hash_str
2445       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2446     \endcsname
2447   }{
2448     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2449   }
2450 }

```

(End definition for \\_stex\_terms\_invoke\_op:nw.)

\\_stex\_terms\_invoke\_math:nw

```

2451 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2452   \_stex_terms_args:n { #2 }
2453   \prop_set_eq:Nc \l_tmpa_prop {
2454     g_stex_symdecl_ #1 _prop
2455   }
2456   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2457   \seq_if_empty:NTF \l_tmpa_seq {
2458     \msg_error:nnxn{stex}{error/nonotation}{#1}{s}
2459   } {
2460     \seq_if_in:NxTF \l_tmpa_seq
2461       { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2462       \use:c{
2463         stex_notation_ #1 \c_hash_str
2464         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2465         _cs
2466       }
2467     }{
2468       \str_if_empty:NTF \l__stex_terms_variant_str {
2469         \str_if_empty:NTF \l__stex_terms_lang_str {
2470           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2471           \use:c{
2472             stex_notation_ #1 \c_hash_str \l_tmpa_str
2473             _cs
2474           }
2475         }{
2476           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2477             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2478           }
2479         }
2480       }{
2481         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2482           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2483         }
2484       }
2485     }
2486   }

```

```
2487 }
(End definition for \_stex_terms_invoke_math:nw.)
```

```
\_stex_terms_invoke_text:n
2488 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2489   \peek_charcode_remove:NTF ! {
2490     \stex_term_custom:nn { #1 } { }
2491   }{
2492     \prop_set_eq:Nc \l_tmpa_prop {
2493       g_stex_symdecl_ #1 _prop
2494     }
2495     \prop_get:Nn \l_tmpa_prop { args } \l_tmpa_str
2496     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2497   }
2498 }
(End definition for \_stex_terms_invoke_text:n.)
```

## 24.2 Terms

Precedences:

```
\infprec
\neginfprec
\l__stex_terms_downprec
2499 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2500 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2501 \int_new:N \l__stex_terms_downprec
2502 \int_set_eq:NN \l__stex_terms_downprec \infprec
```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 28.)

Bracketing:

```
\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2503 \tl_set:Nn \l__stex_terms_left_bracket_str (
2504 \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

```
2505 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2506   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2507     \bool_set_false:N \l__stex_terms_brackets_done_bool
2508     #2
2509   } {
2510     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2511       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2512         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2513         \dobrackets { #2 }
2514       }
2515     }{ #2 }
2516   }
2517 }
```

(End definition for `\_stex_terms_maybe_brackets:nn`.)

### **\dobrackets**

```
2518 \bool_new:N \l__stex_terms_brackets_done_bool
2519 %\RequirePackage{scalerel}
2520 \cs_new_protected:Npn \dobrackets #1 {
2521   %\ThisStyle{\if D\m@switch
2522   %   \exp_args:Nnx \use:nn
2523   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2524   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2525   % \else
2526   \exp_args:Nnx \use:nn
2527   {
2528     \bool_set_true:N \l__stex_terms_brackets_done_bool
2529     \int_set:Nn \l__stex_terms_downprec \infpref
2530     \l__stex_terms_left_bracket_str
2531     #1
2532   }
2533   {
2534     \bool_set_false:N \l__stex_terms_brackets_done_bool
2535     \l__stex_terms_right_bracket_str
2536     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2537   }
2538   %\fi}
2539 }
```

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

### **\withbrackets**

```
2540 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2541   \exp_args:Nnx \use:nn
2542   {
2543     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2544     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2545     #3
2546   }
2547   {
2548     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2549     {\l__stex_terms_left_bracket_str}
2550     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2551     {\l__stex_terms_right_bracket_str}
2552   }
2553 }
```

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

### **\STEXinvisible**

```
2554 \cs_new_protected:Npn \STEXinvisible #1 {
2555   \stex_annotate_invisible:n { #1 }
2556 }
```

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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```

2557 \cs_new_protected:Nn \_stex_term_oms:nnn {
2558   \stex_annotate:nnn{ OMID }{ #2 }{
2559     \stex_highlight_term:nn { #1 } { #3 }
2560   }
2561 }
2562
2563 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2564   \__stex_terms_maybe_brackets:nn { #3 }{
2565     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2566   }
2567 }

```

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

`\_stex_term_math_oma:nnnn`

```

2568 \cs_new_protected:Nn \_stex_term_oma:nnn {
2569   \stex_annotate:nnn{ OMA }{ #2 }{
2570     \stex_highlight_term:nn { #1 } { #3 }
2571   }
2572 }
2573
2574 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2575   \__stex_terms_maybe_brackets:nn { #3 }{
2576     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2577   }
2578 }

```

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

`\_stex_term_math_omb:nnnn`

```

2579 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2580   \stex_annotate:nnn{ OMBIND }{ #2 }{
2581     \stex_highlight_term:nn { #1 } { #3 }
2582   }
2583 }
2584
2585 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2586   \__stex_terms_maybe_brackets:nn { #3 }{
2587     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2588   }
2589 }

```

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

`\_stex_term_math_arg:nnn`

```

2590 \cs_new_protected:Nn \_stex_term_arg:nn {
2591   \stex_unhighlight_term:n {
2592     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2593   }
2594 }
2595 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2596   \exp_args:Nnx \use:nn
2597     { \int_set:Nn \l__stex_terms_downprec { #2 }

```

```

2598     \stex_term_arg:nn { #1 }{ #3 }
2599   }
2600   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2601 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2602 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2603   \clist_set:Nn \l_tmpa_clist{ #4 }
2604   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2605     \tl_set:Nn \l_tmpa_tl { #4 }
2606   }{
2607     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2608     \clist_reverse:N \l_tmpa_clist
2609     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2610
2611     \clist_map_inline:Nn \l_tmpa_clist {
2612       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2613         \exp_args:Nno
2614         \l_tmpa_cs { ##1 } \l_tmpa_tl
2615       }
2616     }
2617
2618   }
2619   \exp_args:Nnno
2620   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2621 }

```

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

`\stex_term_custom:nn`

```

2622 \cs_new_protected:Nn \stex_term_custom:nn {
2623   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2624   \str_set:Nn \l_tmpa_str { #2 }
2625   \tl_clear:N \l_tmpa_tl
2626   \int_zero:N \l_tmpa_int
2627   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2628   \__stex_terms_custom_loop:
2629 }

```

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

`\__stex_terms_custom_loop:`

```

2630 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2631   \bool_set_false:N \l_tmpa_bool
2632   \bool_while_do:nn {
2633     \str_if_eq_p:ee X {
2634       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2635     }
2636   }{
2637     \int_incr:N \l_tmpa_int
2638   }
2639
2640   \peek_charcode:NNTF [ {

```



```

2641 % notation/text component
2642 \__stex_terms_custom_component:w
2643 } {
2644 \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2645 % all arguments read => finish
2646 \__stex_terms_custom_final:
2647 } {
2648 % arguments missing
2649 \peek_charcode_remove:NTF * {
2650 % invisible, specific argument position or both
2651 \peek_charcode:NTF [ {
2652 % visible specific argument position
2653 \__stex_terms_custom_arg:wn
2654 } {
2655 % invisible
2656 \peek_charcode_remove:NTF * {
2657 % invisible specific argument position
2658 \__stex_terms_custom_arg_inv:wn
2659 } {
2660 % invisible next argument
2661 \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2662 }
2663 }
2664 } {
2665 % next normal argument
2666 \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2667 }
2668 }
2669 }
2670 }

```

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

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

```

2671 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2672 \bool_set_true:N \l_tmpa_bool
2673 \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2674 }

```

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

\\_\_stex\_terms\_custom\_arg:wn

```

2675 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2676 \str_set:Nx \l_tmpb_str {
2677 \str_item:Nn \l_tmpa_str { #1 }
2678 }
2679 \str_case:VnTF \l_tmpb_str {
2680 { X } {
2681 \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2682 }
2683 { i } { \__stex_terms_custom_set_X:n { #1 } }
2684 { b } { \__stex_terms_custom_set_X:n { #1 } }
2685 { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2686 { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2687 }{}{

```

```

2688 \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2689 }
2690
2691 \bool_if:nTF \l_tmpa_bool {
2692   \tl_put_right:Nx \l_tmpa_tl {
2693     \stex_annotate_invisible:n {
2694       \stex_term_arg:nn { \int_eval:n { #1 } }
2695       \exp_not:n { { #2 } }
2696     }
2697   }
2698 } {
2699   \tl_put_right:Nx \l_tmpa_tl {
2700     \stex_term_arg:nn { \int_eval:n { #1 } }
2701     \exp_not:n { { #2 } }
2702   }
2703 }
2704
2705 \__stex_terms_custom_loop:
2706 }

```

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

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

```

2707 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2708   \str_set:Nx \l_tmpa_str {
2709     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2710     X
2711     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2712   }
2713 }

```

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

\\_stex\_terms\_custom\_component:

```

2714 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2715   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2716   \__stex_terms_custom_loop:
2717 }

```

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

\\_\_stex\_terms\_custom\_final:

```

2718 \cs_new_protected:Nn \__stex_terms_custom_final: {
2719   \int_compare:nNnTF \l_tmpb_int = 0 {
2720     \exp_args:Nnno \stex_term_oms:nnn
2721   } {
2722     \str_if_in:NnTF \l_tmpa_str {b} {
2723       \exp_args:Nnno \stex_term_ombind:nnn
2724     } {
2725       \exp_args:Nnno \stex_term_oma:nnn
2726     }
2727   }
2728   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2729 }

```

(End definition for `\_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

2730 \NewDocumentCommand \symref { m m }{
2731   \let\compemph_uri_prev:\compemph@uri
2732   \let\compemph@uri\symrefemph@uri
2733   \STEXsymbol{#1}![#2]
2734   \let\compemph@uri\compemph_uri_prev:
2735 }
2736
2737 \keys_define:nn { stex / symname } {
2738   post      .str_set_x:N   = \l_stex_symname_post_str
2739 }
2740
2741 \cs_new_protected:Nn \stex_symname_args:n {
2742   \str_clear:N \l_stex_symname_post_str
2743   \keys_set:nn { stex / symname } { #1 }
2744 }
2745
2746 \NewDocumentCommand \symname { 0{} m }{
2747   \stex_symname_args:n { #1 }
2748   \stex_get_symbol:n { #2 }
2749   \str_set:Nx \l_tmpa_str {
2750     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2751   }
2752   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2753
2754   \let\compemph_uri_prev:\compemph@uri
2755   \let\compemph@uri\symrefemph@uri
2756   \exp_args:NNx \use:nn
2757   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2758     \l_tmpa_str \l_stex_symname_post_str
2759   ] }
2760   \let\compemph@uri\compemph_uri_prev:
2761 }

```

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

## 24.3 Notation Components

2762 `<@@=stex_notationcomps>`

`\stex_highlight_term:nn`

```

2763
2764 \str_new:N \l__stex_notationcomps_highlight_uri_str
2765 \cs_new_protected:Nn \stex_highlight_term:nn {
2766   \exp_args:Nnx
2767   \use:nn {
2768     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2769     #2
2770   } {
2771     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2772     { \l__stex_notationcomps_highlight_uri_str }
2773   }

```

```

2774 }
2775
2776 \cs_new_protected:Nn \stex_unhighlight_term:n {
2777 % \latexml_if:TF {
2778 %   #1
2779 % } {
2780 %   \rustex_if:TF {
2781 %     #1
2782 %   } {
2783     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2784 %   }
2785 % }
2786 }

```

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

```

\comp
\compemph@uri 2787 \cs_new_protected:Npn \comp #1 {
\compemph 2788 \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph 2789 \rustex_if:TF {
\defemph@uri 2790 \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph 2791 }{
\symrefemph@uri 2792 \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2793 }
2794 }
2795 }
2796
2797 \cs_new_protected:Npn \compemph@uri #1 #2 {
2798 \compemph{ #1 }
2799 }
2800
2801
2802 \cs_new_protected:Npn \compemph #1 {
2803 \textcolor{blue}{#1}
2804 }
2805
2806 \cs_new_protected:Npn \defemph@uri #1 #2 {
2807 \defemph{#1}
2808 }
2809
2810 \cs_new_protected:Npn \defemph #1 {
2811 \textbf{#1}
2812 }
2813
2814 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2815 \symrefemph{#1}
2816 }
2817
2818 \cs_new_protected:Npn \symrefemph #1 {
2819 \textbf{#1}
2820 }

```

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

**\ellipses**

```
2821 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```
\parray
\prmatrix 2822 \bool_new:N \l_stex_inarray_bool
\parrayline 2823 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2824 \NewDocumentCommand \parray { m m } {
\parraycell 2825 \begingroup
2826 \bool_set_true:N \l_stex_inarray_bool
2827 \begin{array}{#1}
2828 #2
2829 \end{array}
2830 \endgroup
2831 }
2832
2833 \NewDocumentCommand \prmatrix { m } {
2834 \begingroup
2835 \bool_set_true:N \l_stex_inarray_bool
2836 \begin{matrix}
2837 #1
2838 \end{matrix}
2839 \endgroup
2840 }
2841
2842 \def \maybepline {
2843 \bool_if:NT \l_stex_inarray_bool {\hline}
2844 }
2845
2846 \def \parrayline #1 #2 {
2847 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2848 }
2849
2850 \def \pmrow #1 { \parrayline{}{ #1 } }
2851
2852 \def \parraylineh #1 #2 {
2853 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2854 }
2855
2856 \def \parraycell #1 {
2857 #1 \bool_if:NT \l_stex_inarray_bool {&}
2858 }
```

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

```
2859 \endpackage
```

## Chapter 25

# STEX -Structural Features Implementation

```
2860 <*package>
2861
2862 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2863
2864 <@@=stex_features>
      Warnings and error messages
2865
```

### 25.1 The feature environment

structural@feature

```
2866
2867 \NewDocumentEnvironment{structural@feature}{ m m m }{
2868   \stex_if_in_module:F {
2869     \msg_set:nnn{stex}{error/nomodule}{
2870       Structural~Feature~has~to~occur~in~a~module:\\
2871       Feature~#2~of~type~#1\\
2872       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2873     }
2874     \msg_error:nn{stex}{error/nomodule}
2875   }
2876
2877   \str_set:Nx \l_stex_module_name_str {
2878     \prop_item:Nn \l_stex_current_module_prop
2879     { name } / #2 - feature
2880   }
2881
2882   \str_set:Nx \l_stex_module_ns_str {
2883     \prop_item:Nn \l_stex_current_module_prop
2884     { ns }
2885   }
2886
```

```

2887 \str_clear:N \l_tmpa_str
2888 \seq_clear:N \l_tmpa_seq
2889 \tl_clear:N \l_tmpa_tl
2890 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2891   origname = #2,
2892   name      = \l_stex_module_name_str ,
2893   ns        = \l_stex_module_ns_str ,
2894   imports   = \exp_not:o { \l_tmpa_seq } ,
2895   constants = \exp_not:o { \l_tmpa_seq } ,
2896   content   = \exp_not:o { \l_tmpa_tl } ,
2897   file      = \exp_not:o { \g_stex_currentfile_seq } ,
2898   lang      = \l_stex_module_lang_str ,
2899   sig       = \l_tmpa_str ,
2900   meta      = \l_tmpa_str ,
2901   feature   = #1 ,
2902 }
2903
2904
2905 \stex_if_smsmode:TF {
2906   \stex_smsmode_set_codes:
2907 } {
2908   \begin{stex_annotate_env}{ feature:#1 }{}
2909   \stex_annotate_invisible:nnn{header}{}{ #3 }
2910 }
2911 }{
2912   \str_set:Nx \l_tmpa_str {
2913     c_stex_feature_
2914     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2915     \prop_item:Nn \l_stex_current_module_prop { name }
2916     _prop
2917   }
2918   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2919   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2920   \stex_if_smsmode:TF {
2921     \exp_args:Nx \stex_add_to_sms:n {
2922       \prop_gset_from_keyval:cn {
2923         c_stex_feature_
2924         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2925         \prop_item:Nn \l_stex_current_module_prop { name }
2926         _prop
2927       } {
2928         origname = #2,
2929         name      = \prop_item:cn { \l_tmpa_str } { name } ,
2930         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2931         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2932         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2933         content   = \prop_item:cn { \l_tmpa_str } { content } ,
2934         file      = \prop_item:cn { \l_tmpa_str } { file } ,
2935         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
2936         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
2937         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2938         feature   = \prop_item:cn { \l_tmpa_str } { feature }
2939       }
2940     }

```

```

2941 } {
2942     \end{stex_annotate_env}
2943 }
2944 }
2945

```

## 25.2 Features

structure

```

2946
2947 \prop_new:N \l_stex_all_structures_prop
2948
2949 \keys_define:nn { stex / features / structure } {
2950     name .str_set_x:N = \l__stex_features_structure_name_str ,
2951 }
2952
2953 \cs_new_protected:Nn \__stex_features_structure_args:n {
2954     \str_clear:N \l__stex_features_structure_name_str
2955     \keys_set:nn { stex / features / structure } { #1 }
2956 }
2957
2958 %\stex_new_feature:nnnn { structure } { 0{} m } {
2959 % \__stex_features_structure_args:n { ##1 }
2960 % \str_if_empty:NT \l__stex_features_structure_name_str {
2961 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2962 % }
2963 %} {
2964 %
2965 %}
2966
2967 \NewDocumentEnvironment{mathstructure}{ 0{} m }{
2968     \__stex_features_structure_args:n { #1 }
2969     \str_if_empty:NT \l__stex_features_structure_name_str {
2970         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2971     }
2972     \exp_args:Nnnx
2973     \begin{structural@feature}{ structure }
2974         { \l__stex_features_structure_name_str }{}
2975         \seq_clear:N \l_tmpa_seq
2976         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2977     }{
2978         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2979         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2980         \str_set:Nx \l_tmpa_str {
2981             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2982             \prop_item:Nn \l_stex_current_module_prop { name }
2983         }
2984         \seq_map_inline:Nn \l_tmpa_seq {
2985             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2986         }
2987         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2988         \exp_args:Nnx

```



```

2990 \AddToHookNext { env / mathstructure / after }{
2991 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2992 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2993 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2994 \STEXexport {
2995 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2996 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2997 {\l_tmpa_str}
2998 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2999 {#2}{\l_tmpa_str}
3000 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3001 % \prop_item:Nn \l_stex_current_module_prop { origname },
3002 % \l_tmpa_str
3003 % }
3004 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3005 % #2,\l_tmpa_str
3006 % }
3007 % \tl_set:cx { #2 } {
3008 % \stex_invoke_structure:n { \l_tmpa_str }
3009 }
3010 }
3011
3012 \end{structural@feature}
3013 % \g_stex_last_feature_prop
3014 }

```

\instantiate

```

3015 \seq_new:N \l__stex_features_structure_field_seq
3016 \str_new:N \l__stex_features_structure_field_str
3017 \str_new:N \l__stex_features_structure_def_tl
3018 \prop_new:N \l__stex_features_structure_prop
3019 \NewDocumentCommand \instantiate { m O{} m }{
3020 \stex_smsmode_set_codes:
3021 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3022 \prop_set_eq:Nc \l__stex_features_structure_prop {
3023 c_stex_feature_\l_tmpa_str _prop
3024 }
3025 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3026 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3027 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3028 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3029 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3030 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3031 {!} \l_tmpa_tl
3032 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3033 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3034 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3035 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3036 }{
3037 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3038 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3039 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3040 \l_tmpa_tl
3041 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

3042         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3043         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3044     }{
3045         \tl_clear:N \l_tmpb_tl
3046     }
3047 }
3048 }{
3049     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3050     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3051         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3052         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3053         \tl_clear:N \l_tmpa_tl
3054     }{
3055         % TODO throw error
3056     }
3057 }
3058 % \l_tmpa_str: name
3059 % \l_tmpa_tl: definiens
3060 % \l_tmpb_tl: notation
3061 \tl_if_empty:NT \l__stex_features_structure_field_str {
3062     % TODO throw error
3063 }
3064 \str_clear:N \l_tmpb_str
3065
3066 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3067 \seq_map_inline:Nn \l_tmpa_seq {
3068     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3069     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3070     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3071         \seq_map_break:n {
3072             \str_set:Nn \l_tmpb_str { ####1 }
3073         }
3074     }
3075 }
3076 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
3077     \l_tmpb_str
3078
3079 \tl_if_empty:NNTF \l_tmpb_tl {
3080     \tl_if_empty:NF \l_tmpa_tl {
3081         \exp_args:Nx \use:n {
3082             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3083         }
3084     }
3085 }{
3086     \tl_if_empty:NNTF \l_tmpa_tl {
3087         \exp_args:Nx \use:n {
3088             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3089         }
3090     }
3091 }{
3092     \exp_args:Nx \use:n {
3093         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3094         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3095     }

```

```

3096     }
3097   }
3098   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3099   % \prop_item:Nn \l_stex_current_module_prop {name} ?
3100   % #3/\l_stex_features_structure_field_str
3101   % \par
3102   % \expandafter\present\csname
3103   %   g_stex_symdecl_
3104   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3105   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3106   %   #3/\l_stex_features_structure_field_str
3107   %   _prop
3108   % \endcsname
3109 }
3110
3111 \tl_clear:N \l__stex_features_structure_def_tl
3112
3113 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3114 \seq_map_inline:Nn \l_tmpa_seq {
3115   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3116   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3117   \exp_args:Nx \use:n {
3118     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3119
3120     }
3121   }
3122
3123   \prop_if_exist:cF {
3124     g_stex_symdecl_
3125     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3126     \prop_item:Nn \l_stex_current_module_prop {name} ?
3127     #3/\l_tmpa_str
3128     _prop
3129   }{
3130     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
3131     \l_tmpb_str
3132     \exp_args:Nx \use:n {
3133       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3134     }
3135   }
3136 }
3137
3138 \symdecl*[type={\STEXsymbol{module-type}}{
3139   \_stex_term_math_oms:nnnn {
3140     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3141     \prop_item:Nn \l__stex_features_structure_prop {name}
3142     }{}{0}{}
3143   }{}{#3}
3144
3145   % TODO: -> sms file
3146
3147   \tl_set:cx{ #3 }{
3148     \stex_invoke_structure:nnn {
3149       \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

3150     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3151   } {
3152     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3153     \prop_item:Nn \l__stex_features_structure_prop {name}
3154   }
3155 }
3156
3157 }

```

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

\stex\_invoke\_structure:nnn

```

3158 % #1: URI of the instance
3159 % #2: URI of the instantiated module
3160 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3161   \tl_if_empty:nTF{ #3 }{
3162     \prop_set_eq:Nc \l__stex_features_structure_prop {
3163       c_stex_feature_ #2 _prop
3164     }
3165     \tl_clear:N \l_tmpa_tl
3166     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3167     \seq_map_inline:Nn \l_tmpa_seq {
3168       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3169       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3170       \cs_if_exist:cT {
3171         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3172       }{
3173         \tl_if_empty:NF \l_tmpa_tl {
3174           \tl_put_right:Nn \l_tmpa_tl {,}
3175         }
3176         \tl_put_right:Nx \l_tmpa_tl {
3177           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3178         }
3179       }
3180     }
3181     \exp_args:No \mathstrut \l_tmpa_tl
3182   }{
3183     \stex_invoke_symbol:n{#1/#3}
3184   }
3185 }

```

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

```

3186 </package>

```

## Chapter 26

# STEX -Statements Implementation

```
3187 <*package>
3188
3189 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3190
3191 \protected\def\ignorespacesandpars{
3192   \begingroup\catcode13=10\relax
3193   \@ifnextchar\par{
3194     \endgroup\expandafter\ignorespacesandpars\@gobble
3195   }{
3196     \endgroup
3197   }
3198 }
3199
3200 <@@=stex_statements>
3201
3202   Warnings and error messages
```

`\titleemph`

```
3202 \def\titleemph#1{\textbf{#1}}
```

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

### 26.1 Definitions

`definiendum`

```
3203 \keys_define:nn {stex / definiendum }{
3204   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3205   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3206   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3207 }
3208 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3209   \str_clear:N \l__stex_statements_definiendum_root_str
3210   \tl_clear:N \l__stex_statements_definiendum_post_tl
3211   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3212 \keys_set:nn { stex / definiendum } { #1 }
3213 }
3214 \NewDocumentCommand \definiendum { 0{} m m } {
3215   \__stex_statements_definiendum_args:n { #1 }
3216   \stex_get_symbol:n { #2 }
3217   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3218   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3219     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3220       \tl_set:Nn \l_tmpa_tl { #3 }
3221     } {
3222       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3223       \tl_set:Nn \l_tmpa_tl {
3224         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3225       }
3226     }
3227   } {
3228     \tl_set:Nn \l_tmpa_tl { #3 }
3229   }
3230
3231   % TODO root
3232   \rustex_if:TF {
3233     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3234   } {
3235     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3236   }
3237 }
3238 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

#### definame

```

3239 \NewDocumentCommand \definame { 0{} m } {
3240   \__stex_statements_definiendum_args:n { #1 }
3241   % TODO: root
3242   \stex_get_symbol:n { #2 }
3243   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3244   \str_set:Nx \l_tmpa_str {
3245     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3246   }
3247   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3248   \rustex_if:TF {
3249     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3250       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3251     }
3252   } {
3253     \defemph@uri {
3254       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3255     } { \l_stex_get_symbol_uri_str }
3256   }
3257 }
3258 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3259
3260 \keys_define:nn {stex / sdefinition }{
3261   type      .str_set_x:N = \sdefinitiontype,
3262   id        .str_set_x:N = \sdefinitionid,
3263   title     .tl_set:N    = \sdefinitiontitle
3264 }
3265 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3266   \str_clear:N \sdefinitiontype
3267   \str_clear:N \sdefinitionid
3268   \tl_clear:N \sdefinitiontitle
3269   \keys_set:nn { stex / sdefinition }{ #1 }
3270 }
3271
3272 \NewDocumentEnvironment{sdefinition}{0{}}{
3273   \__stex_statements_sdefinition_args:n{ #1 }
3274   \stex_reactivate_macro:N \definiendum
3275   \stex_reactivate_macro:N \definame
3276   \stex_smsmode_set_codes:
3277   \clist_set:No \l_tmpa_clist \sdefinitiontype
3278   \tl_clear:N \l_tmpa_tl
3279   \clist_map_inline:Nn \l_tmpa_clist {
3280     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3281       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3282     }
3283   }
3284   \tl_if_empty:NTF \l_tmpa_tl {
3285     \__stex_statements_sdefinition_start:
3286   }{
3287     \l_tmpa_tl
3288   }
3289   \stex_ref_new_doc_target:n \sdefinitionid
3290   \stex_if_smsmode:F {
3291     \exp_args:Nnnx
3292     \begin{stex_annotate_env}{definition}{}
3293     \str_if_empty:NF \sdefinitiontype {
3294       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3295     }
3296   }
3297 }{
3298   \stex_if_smsmode:F {
3299     \end{stex_annotate_env}
3300   }
3301   \clist_set:No \l_tmpa_clist \sdefinitiontype
3302   \tl_clear:N \l_tmpa_tl
3303   \clist_map_inline:Nn \l_tmpa_clist {
3304     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3305       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3306     }
3307   }
3308   \tl_if_empty:NTF \l_tmpa_tl {
3309     \__stex_statements_sdefinition_end:
3310   }{
3311     \l_tmpa_tl

```

```

3312 }
3313 }

```

`\stexpatchdefinition`

```

3314 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3315   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3316     ~(\sdefinitiontitle)
3317   }~}
3318 }
3319 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3320
3321 \newcommand\stexpatchdefinition[3] [] {
3322   \str_set:Nx \l_tmpa_str{ #1 }
3323   \str_if_empty:NTF \l_tmpa_str {
3324     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3325     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3326   }{
3327     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3328     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3329   }
3330 }

```

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

`\inlinedef inline:`

```

3331 \NewDocumentCommand \inlinedef { m } {
3332   \begingroup
3333   \stex_reactivate_macro:N \definiendum
3334   \stex_reactivate_macro:N \definame
3335   \stex_ref_new_doc_target:n{
3336     #1
3337   }
3338 }

```

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

## 26.2 Assertions

`sassertion`

```

3339
3340 \keys_define:nn {stex / sassertion }{
3341   type      .str_set_x:N = \sassertiontype,
3342   id        .str_set_x:N = \sassertionid,
3343   title     .tl_set:N     = \sassertiontitle ,
3344   name      .str_set_x:N = \sassertionname
3345 }
3346 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3347   \str_clear:N \sassertiontype
3348   \str_clear:N \sassertionid
3349   \str_clear:N \sassertionname
3350   \tl_clear:N \sassertiontitle
3351   \keys_set:nn { stex / sassertion }{ #1 }
3352 }

```



```

3353
3354 \tl_new:N \g__stex_statements_aftergroup_tl
3355
3356 \NewDocumentEnvironment{sassertion}{0{}}{
3357   \__stex_statements_sassertion_args:n{ #1 }
3358   \stex_smsmode_set_codes:
3359   \clist_set:No \l_tmpa_clist \sassertiontype
3360   \tl_clear:N \l_tmpa_tl
3361   \clist_map_inline:Nn \l_tmpa_clist {
3362     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3363       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3364     }
3365   }
3366   \tl_if_empty:NTF \l_tmpa_tl {
3367     \__stex_statements_sassertion_start:
3368   }{
3369     \l_tmpa_tl
3370   }
3371   \stex_ref_new_doc_target:n \sassertionid
3372   \stex_if_smsmode:F {
3373     \exp_args:Nnnx
3374     \begin{stex_annotate_env}{assertion}{}
3375     \str_if_empty:NF \sassertiontype {
3376       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3377     }
3378   }
3379   }{
3380     \stex_if_smsmode:F {
3381       \end{stex_annotate_env}
3382     }
3383     \clist_set:No \l_tmpa_clist \sassertiontype
3384     \tl_clear:N \l_tmpa_tl
3385     \clist_map_inline:Nn \l_tmpa_clist {
3386       \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3387         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3388       }
3389     }
3390     \tl_if_empty:NTF \l_tmpa_tl {
3391       \__stex_statements_sassertion_end:
3392     }{
3393       \l_tmpa_tl
3394     }
3395     \str_if_empty:NF \sassertionname {
3396       \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3397         \symdecl*{\sassertionname}
3398       }
3399       \aftergroup\g__stex_statements_aftergroup_tl
3400     }
3401   }

```

\stexpatchassertion

```

3402
3403 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3404   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

3405     (\sassertiontitle)
3406   }~}
3407 }
3408 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3409
3410 \newcommand\stexpatchassertion[3] [] {
3411   \str_set:Nx \l_tmpa_str{ #1 }
3412   \str_if_empty:NTF \l_tmpa_str {
3413     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3414     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3415   }{
3416     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3417     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3418   }
3419 }

```

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

`\inlineass` inline:

```

3420 \NewDocumentCommand \inlineass { m } {
3421   \begingroup
3422   \stex_ref_new_doc_target:n{
3423     #1
3424   \endgroup
3425 }

```

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

## 26.3 Examples

`sexample`

```

3426
3427 \keys_define:nn {stex / sexample }{
3428   type      .str_set_x:N = \exampletype,
3429   id        .str_set_x:N = \sexampleid,
3430   title     .tl_set:N = \sexampletile,
3431   for       .clist_set:N = \sexamplefor,
3432 }
3433 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3434   \str_clear:N \sexampletype
3435   \str_clear:N \sexampleid
3436   \tl_clear:N \sexampletile
3437   \clist_clear:N \sexamplefor
3438   \keys_set:nn { stex / sexample }{ #1 }
3439 }
3440
3441 \NewDocumentEnvironment{sexample}{0{}}{
3442   \__stex_statements_sexample_args:n{ #1 }
3443   \stex_smsmode_set_codes:
3444   \clist_set:Nn \l_tmpa_clist \sexampletype
3445   \tl_clear:N \l_tmpa_tl
3446   \clist_map_inline:Nn \l_tmpa_clist {
3447     \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{

```

```

3448     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3449   }
3450 }
3451 \tl_if_empty:NTF \l_tmpa_tl {
3452   \__stex_statements_sexample_start:
3453 }{
3454   \l_tmpa_tl
3455 }
3456 \stex_ref_new_doc_target:n \sexampleid
3457 \stex_if_smsmode:F {
3458   \seq_clear:N \l_tmpa_seq
3459   \clist_map_inline:Nn \sexamplefor {
3460     \str_if_eq:nnF{ ##1 }{{
3461       \stex_get_symbol:n { ##1 }
3462       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3463         \l_stex_get_symbol_uri_str
3464       }
3465     }
3466   }
3467   \exp_args:Nnnx
3468   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3469   \str_if_empty:NF \sexamplotype {
3470     \stex_annotate_invisible:nnn{type}{\sexampletype}{
3471   }
3472 }
3473 }{
3474   \stex_if_smsmode:F {
3475     \end{stex_annotate_env}
3476   }
3477   \clist_set:Nn \l_tmpa_clist \sexamplotype
3478   \tl_clear:N \l_tmpa_tl
3479   \clist_map_inline:Nn \l_tmpa_clist {
3480     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
3481       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
3482     }
3483   }
3484   \tl_if_empty:NTF \l_tmpa_tl {
3485     \__stex_statements_sexample_end:
3486   }{
3487     \l_tmpa_tl
3488   }
3489 }

```

\stexpatchexample

```

3490
3491 \cs_new_protected:Nn \__stex_statements_sexample_start: {
3492   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
3493     (\sexamplotype)
3494   }~}
3495 }
3496 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
3497
3498 \newcommand\stexpatchexample[3][ ] {
3499   \str_set:Nx \l_tmpa_str{ #1 }

```

```

3500   \str_if_empty:NTF \l_tmpa_str {
3501     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
3502     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
3503   }{
3504     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
3505     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
3506   }
3507 }

```

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

`\inlineex` inline:

```

3508 \NewDocumentCommand \inlineex { m } {
3509   \begingroup
3510   \stex_ref_new_doc_target:n{
3511     #1
3512   \endgroup
3513 }

```

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

## 26.4 Logical Paragraphs

`sparagraph`

```

3514 \keys_define:nn { stex / sparagraph } {
3515   id      .str_set_x:N = \sparagraphid ,
3516   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
3517   type    .str_set_x:N = \sparagraphtype ,
3518   for     .str_set_x:N = \sparagraphfor ,
3519   from    .tl_set_x:N  = \sparagraphfrom ,
3520   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
3521   name    .str_set:N   = \sparagraphname
3522 }
3523
3524 \cs_new_protected:Nn \stex_sparagraph_args:n {
3525   \tl_clear:N \l_stex_sparagraph_title_tl
3526   \tl_clear:N \sparagraphfrom
3527   \tl_clear:N \l_stex_sparagraph_start_tl
3528   \str_clear:N \sparagraphid
3529   \str_clear:N \sparagraphtype
3530   \str_clear:N \sparagraphfor
3531   \str_clear:N \sparagraphname
3532   \keys_set:nn { stex / sparagraph } { #1 }
3533 }
3534 \newif\if@in@omtext\@in@omtextfalse
3535
3536 \NewDocumentEnvironment {sparagraph} { 0{} } {
3537   \stex_sparagraph_args:n { #1 }
3538   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3539     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
3540   }{
3541     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
3542   }

```

```

3543 \@in@omtexttrue
3544 \stex_smsmode_set_codes:
3545 \clist_set:No \l_tmpa_clist \sparagraphtype
3546 \tl_clear:N \l_tmpa_tl
3547 \clist_map_inline:Nn \l_tmpa_clist {
3548   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
3549     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
3550   }
3551 }
3552 \tl_if_empty:NTF \l_tmpa_tl {
3553   \__stex_statements_sparagraph_start:
3554 }{
3555   \l_tmpa_tl
3556 }
3557 \stex_ref_new_doc_target:n \sparagraphid
3558 \stex_if_smsmode:F {
3559   \exp_args:Nnnx
3560   \begin{stex_annotate_env}{paragraph}{}
3561   \str_if_empty:NF \sparagraphtype {
3562     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
3563   }
3564 }
3565 \ignorespacesandpars
3566 }{
3567   \stex_if_smsmode:F {
3568     \end{stex_annotate_env}
3569   }
3570   \clist_set:No \l_tmpa_clist \sparagraphtype
3571   \tl_clear:N \l_tmpa_tl
3572   \clist_map_inline:Nn \l_tmpa_clist {
3573     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
3574       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
3575     }
3576   }
3577   \tl_if_empty:NTF \l_tmpa_tl {
3578     \__stex_statements_sparagraph_end:
3579   }{
3580     \l_tmpa_tl
3581   }
3582   \str_if_empty:NF \sparagraphname {
3583     \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3584       \symdecl*{\sparagraphname}
3585     }
3586     \aftergroup\g__stex_statements_aftergroup_tl
3587   }
3588 }

```

\stexpatchparagraph

```

3589
3590 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
3591   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3592     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
3593       \titleemph{\l_stex_sparagraph_title_tl}:~
3594     }

```

```

3595   }{
3596   \titleemph{\l_stex_sparagraph_start_tl}~
3597   }
3598 }
3599 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
3600
3601 \newcommand\stexpatchparagraph[3] [] {
3602   \str_set:Nx \l_tmpa_str{ #1 }
3603   \str_if_empty:NTF \l_tmpa_str {
3604     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
3605     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
3606   }{
3607     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
3608     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
3609   }
3610 }

```

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

#### symboldoc

```

3611 \NewDocumentEnvironment{symboldoc}{ m }{
3612   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3613   \seq_clear:N \l_tmpb_seq
3614   \seq_map_inline:Nn \l_tmpa_seq {
3615     \str_if_eq:nnF{ ##1 }{}{
3616       \stex_get_symbol:n { ##1 }
3617       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3618         \l_stex_get_symbol_uri_str
3619       }
3620     }
3621   }
3622   \par
3623   \exp_args:Nnnx
3624   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3625   }{
3626     \end{stex_annotate_env}
3627   }
3628 \end{package}

```

# Chapter 27

## The Implementation

### 27.1 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).<sup>10</sup>

```
3629 <*package>
3630 <@@=stex_sproof>
3631
3632 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3633
```

### 27.2 Proofs

We first define some keys for the proof environment.

```
3634 \keys_define:nn { stex / spf } {
3635   id          .str_set:N = \l__stex_sproof_spf_id_str,
3636   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3637   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3638   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3639   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3640   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3641   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3642   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3643   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3644   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3645 }
3646 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3647   \str_clear:N \l__stex_sproof_spf_id_str
3648   \tl_clear:N \l__stex_sproof_spf_display_tl
3649   \tl_clear:N \l__stex_sproof_spf_for_tl
3650   \tl_clear:N \l__stex_sproof_spf_from_tl
3651   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3652   \tl_clear:N \l__stex_sproof_spf_type_tl
3653   \tl_clear:N \l__stex_sproof_spf_title_tl

```

---

<sup>10</sup>EDNOTE: need an implementation for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L

```

3654 \tl_clear:N \l__stex_sproof_spf_continues_tl
3655 \tl_clear:N \l__stex_sproof_spf_functions_tl
3656 \tl_clear:N \l__stex_sproof_spf_method_tl
3657 \keys_set:nn { stex / spf }{ #1 }
3658 }

```

`\spf@flow` We define this macro, so that we can test whether the `display` key has the value `flow`

```

3659 \def\spf@flow{flow}

```

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

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, L<sup>A</sup>T<sub>E</sub>X only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

`pst@with@label` This environment manages<sup>6</sup> the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by T<sub>E</sub>X for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```

3660 \newcount\count_ten
3661 \newenvironment{pst@with@label}[1]{
3662   \edef\pst@label{#1}
3663   \advance\count_ten by 1\relax
3664   \count_ten=1
3665 }{
3666   \advance\count_ten by -1\relax
3667 }

```

`\the@pst@label` `\the@pst@label` evaluates to the current step label.

```

3668 \def\the@pst@label{
3669   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3670 }

```

(End definition for `\the@pst@label`. This function is documented on page ??.)

`\setpstlabelstyle` `\setpstlabelstyle{metaKey-Val pairs}` makes the labeling style customizable. `\setpstlabelstyle{pr}` will change the labeling style from **P.1.2.3** to **Pr-1-2-3†**. `\setpstlabelstyledefault` will set the labeling style back to default.

```

3671 \keys_define:nn { stex / pstlabel }{
3672   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3673   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3674   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3675 }
3676 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

<sup>6</sup>This gets the labeling right but only works 8 levels deep



```

3677 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3678 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3679 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3680 }
3681 \__stex_sproof_pstlabel_args:n {}
3682 \newcommand\setpstlabelstyle[1]{
3683   \__stex_sproof_pstlabel_args:n {#1}
3684 }
3685 \newcommand\setpstlabelstyledefault{%
3686   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3687 }

```

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

**\pstlabelstyle** \pstlabelstyle just sets the \pst@make@label macro according to the style.

```

3688 \ExplSyntaxOff
3689 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3690 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3691 \def\pst@make@label@short#1#2{#2}
3692 \def\pst@make@label@empty#1#2{}
3693 \ExplSyntaxOn
3694 \def\pstlabelstyle#1{%
3695   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3696 }%
3697 \pstlabelstyle{long}%

```

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

**\next@pst@label** \next@pst@label increments the step label at the current level.

```

3698 \def\next@pst@label{%
3699   \global\advance\count\count10 by 1%
3700 }%

```

(End definition for \next@pst@label. This function is documented on page ??.)

**\sproofend** This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```

3701 \def\sproof@box{
3702   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3703 }
3704 \def\spf@proofend{\sproof@box}
3705 \def\sproofend{
3706   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3707     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3708   }
3709 }
3710 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

3711 \def\spf@proofsketch@kw{Proof Sketch}
3712 \def\spf@proof@kw{Proof}
3713 \def\spf@step@kw{Step}

```

(End definition for `spf@*kw`. This function is documented on page ??.)

For the other languages, we set up triggers

```

3714 \cs_if_exist:NT \bbl@loaded {
3715   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3716   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3717     \input{proof-ngerman.lda}
3718   }
3719   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3720     \input{proof-finnish.lda}
3721   }
3722   \clist_if_in:NnT \l_tmpa_clist {french}{
3723     \input{proof-french.lda}
3724   }
3725   \clist_if_in:NnT \l_tmpa_clist {russian}{
3726     \input{proof-russian.lda}
3727   }
3728 }
3729

```

**spfsketch**

```

3730 \newcommand\spfsketch[2][]{
3731   \__stex_sproof_spf_args:n{#1}
3732   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3733     \titleemph{
3734       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3735         \spf@proofsketch@kw
3736       }{
3737         \l__stex_sproof_spf_type_tl
3738       }
3739     }:
3740   }
3741   {-#2}
3742   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
3743   \sproofend
3744 }

```

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

**spfeq** This is very similar to `\spfsketch`, but uses a computation array<sup>1112</sup>

```

3745 \newenvironment{spfeq}[2][]{
3746   \__stex_sproof_spf_args:n{#1}
3747   %\sref@target
3748   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3749     \titleemph{
3750       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3751         \spf@proof@kw
3752       }{
3753         \l__stex_sproof_spf_type_tl
3754       }
3755     }:

```

<sup>11</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>12</sup>EDNOTE: document above

```

3756 }
3757 {~#2}
3758 \begin{displaymath}\begin{array}{rcll}
3759 }{
3760 \end{array}\end{displaymath}
3761 }

```

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

**sproof** In this environment, we initialize the proof depth counter `\count10` to 10, and set up the description environment that will take the proof steps. At the end of the proof, we position the proof end into the last line.

```

3762 \newenvironment{spf@proof}[2][]{
3763   \__stex_sproof_spf_args:n{#1}
3764   %\sref@target
3765   \count_ten=10
3766   \par\noindent
3767   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3768     \titleemph{
3769       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3770         \spf@proof@kw
3771       }{
3772         \l__stex_sproof_spf_type_tl
3773       }
3774     }:
3775   }
3776   {~#2}
3777   %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
3778   \def\pst@label{}
3779   \newcount\pst@count% initialize the labeling mechanism
3780   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3781   }{
3782     \end{pst@with@label}\end{description}
3783   }
3784   \newenvironment{sproof}[2][{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3785   \newenvironment{sProof}[2][{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

3786 \newcommand\spfidea[2][]{
3787   \__stex_sproof_spf_args:n{#1}
3788   \titleemph{
3789     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3790       \l__stex_sproof_spf_type_tl
3791     }:
3792   }~#2
3793   \sproofend
3794 }

```

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

The next two environments (proof steps) and comments, are mostly semantical, they take `KeyVal` arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep 13

```

3795 \newenvironment{spfstep}[1][]{
3796   \_stex_sproof_spf_args:n{#1}
3797   \@in@omtexttrue
3798   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3799     \item[\the@pst@label]
3800   }
3801   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3802     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3803   }
3804   %\sref@label@id{\pst@label}
3805   \ignorespacesandpars
3806 }{
3807   \next@pst@label\ignorespacesandpars
3808 }

```

sproofcomment

```

3809 \newenvironment{sproofcomment}[1][]{
3810   \_stex_sproof_spf_args:n{#1}
3811   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3812     \item[\the@pst@label]
3813   }
3814 }{
3815   \next@pst@label
3816 }

```

The next two environments also take a `KeyVal` argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

**subproof** In the `subproof` environment, a new (lower-level) `proproofof` environment is started.

```

3817 \newenvironment{subproof}[2][]{
3818   \_stex_sproof_spf_args:n{#1}
3819   \def\@test{#2}
3820   \ifx\@test\empty\else
3821     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3822       \item[\the@pst@label]
3823     }{#2}
3824   \fi
3825   \begin{pst@with@label}{\pst@label,\number\count_ten}
3826 }{
3827   \end{pst@with@label}\next@pst@label
3828 }

```

**spfcases** In the `pfcases` environment, the start text is displayed as the first comment of the proof.

```

3829 \newenvironment{spfcases}[2][]{
3830   \def\@test{#1}
3831   \ifx\@test\empty
3832     \begin{subproof}[method=by-cases]{#2}
3833   \else
3834     \begin{subproof}[#1,method=by-cases]{#2}
3835   \fi
3836 }{

```

---

<sup>13</sup>EdNOTE: MK: labeling of steps does not work yet.

```

3837 \end{subproof}
3838 }

```

**spfcase** In the **pfcase** environment, the start text is displayed specification of the case after the **\item**

```

3839 \newenvironment{spfcase}[2] [] {
3840   \__stex_sproof_spf_args:n{#1}
3841   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3842     \item[\the@pst@label]
3843   }
3844   \def\@test{#2}
3845   \ifx\@test\@empty
3846   \else
3847     {\titleemph{#2}:~}
3848   \fi
3849   \begin{pst@with@label}{\pst@label,\number\count_ten}
3850 }{
3851   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3852     \sproofend
3853   }
3854   \end{pst@with@label}
3855   \next@pst@label
3856 }

```

**spfcase** similar to **spfcase**, takes a third argument.

```

3857 \newcommand\spfcasesketch[3] [] {
3858   \__stex_sproof_spf_args:n{#1}
3859   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3860     \item[\the@pst@label]
3861   }
3862   \def\@test{#2}
3863   \ifx\@test\@empty
3864   \else
3865     {\titleemph{#2}:~}
3866   \fi#3
3867   \next@pst@label
3868 }%

```

## 27.3 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```

3869 \keys_define:nn { stex / just }{
3870   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3871   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
3872   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
3873   args        .tl_set:N    = \l__stex_sproof_just_args_tl
3874 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>14</sup>

<sup>14</sup>EDNOTE: need to do something about the premise in draft mode.

**justification**

```
3875 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
3876 \newcommand\premise[2] [] {#2}
```

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

**\justarg** the **\justarg** macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```
3877 \newcommand\justarg[2] [] {#2}
```

```
3878 \end{package}
```

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

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

## Chapter 28

# STEX -Others Implementation

```
3879 <*package>
3880
3881 %%%%%%%%%% others.dtx %%%%%%%%%%
3882
3883 <@@=stex_others>
    Warnings and error messages
3884 % None

\MSC Math subject classifier

3885 \NewDocumentCommand \MSC {m} {
3886 % TODO
3887 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3888 \@ifpackageloaded{tikzinput}{
3889 \RequirePackage{stex-tikzinput}
3890 }{}
3891 </package>
```

## Chapter 29

# STEX -Metatheory Implementation

```
3892 \*package>
3893 \@@=stex_modules>
3894
3895 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3896
3897 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3898 \begingroup
3899 \stex_module_setup:nn{
3900   ns=\c_stex_metatheory_ns_str,
3901   meta=NONE
3902 }{Metatheory}
3903 \stex_reactivate_macro:N \symdecl
3904 \stex_reactivate_macro:N \notation
3905 \stex_reactivate_macro:N \symdef
3906 \ExplSyntaxOff
3907 \csname stex_suppress_html:n\endcsname{
3908   % is-a (a:A, a \in A, a is an A, etc.)
3909   \symdecl[args=ai]{isa}
3910   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3911   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3912   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3913
3914   % bind (\forall, \Pi, \lambda etc.)
3915   \symdecl[args=Bi]{bind}
3916   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3917   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3918   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
3919
3920   % dummy variable
3921   \symdecl{dummyvar}
3922   \notation[underscore]{dummyvar}{\comp\_}
3923   \notation[dot]{dummyvar}{\comp\cdot}
3924   \notation[dash]{dummyvar}{\comp{\rm --}}
3925
3926   %fromto (function space, Hom-set, implication etc.)
```



```

3927 \symdecl[args=ai]{fromto}
3928 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3929 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3930
3931 % mapto (lambda etc.)
3932 %\symdecl[args=Bi]{mapto}
3933 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3934 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3935 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3936
3937 % function/operator application
3938 \symdecl[args=ia]{apply}
3939 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3940 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3941
3942 % ‘type’ of all collections (sets, classes, types, kinds)
3943 \symdecl{collection}
3944 \notation[U]{collection}{\comp{\mathcal{U}}}
3945 \notation[set]{collection}{\comp{\textsf{Set}}}
3946
3947 % sequences
3948 \symdecl[args=1]{seqtype}
3949 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3950
3951 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
3952 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
3953
3954 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
3955 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
3956 % ^ superceded by \aseqfromto and \livar/\uivar
3957
3958 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
3959 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
3960 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
3961
3962 % letin (‘let’, local definitions, variable substitution)
3963 \symdecl[args=bii]{letin}
3964 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
3965 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3966 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3967
3968 % structures
3969 \symdecl*[args=1]{module-type}
3970 \notation{module-type}{\mathtt{MOD} #1}
3971 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3972 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3973
3974 }
3975 \ExplSyntaxOn
3976 \stex_add_to_current_module:n{
3977   \let\nappa\apply
3978   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3979   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
3980   \def\livar{\csname sequence-index\endcsname[li]}

```

```

3981 \def\uivar{\csname sequence-index\endcsname[ui]}
3982 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3983 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3984 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3985 }
3986 \__stex_modules_end_module:
3987 \endgroup
3988 \endpackage

```

## Chapter 30

# Tikzinput Implementation

```
3989 <*package>
3990
3991 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3992
3993 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3994 \RequirePackage{l3keys2e}
3995
3996 \keys_define:nn { tikzinput } {
3997   image .bool_set:N = \c_tikzinput_image_bool,
3998   image .default:n = false ,
3999   unknown .code:n = {}
4000 }
4001
4002 \ProcessKeysOptions { tikzinput }
4003
4004 \bool_if:NTF \c_tikzinput_image_bool {
4005   \RequirePackage{graphicx}
4006
4007   \providecommand\usetikzlibrary[]{}
4008   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4009 }{
4010   \RequirePackage{tikz}
4011   \RequirePackage{standalone}
4012
4013   \newcommand \tikzinput [2] [] {
4014     \setkeys{Gin}{#1}
4015     \ifx \Gin@ewidth \Gin@exclamation
4016       \ifx \Gin@eheight \Gin@exclamation
4017         \input { #2 }
4018       \else
4019         \resizebox{!}{ \Gin@eheight }{
4020           \input { #2 }
4021         }
4022       \fi
4023     \else
4024       \ifx \Gin@eheight \Gin@exclamation
4025         \resizebox{ \Gin@ewidth }{!}{
4026           \input { #2 }
```

```

4027     }
4028     \else
4029         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4030             \input { #2 }
4031         }
4032     \fi
4033 \fi
4034 }
4035 }
4036
4037 \newcommand \ctikzinput [2] [] {
4038     \begin{center}
4039         \tikzinput [1] {#2}
4040     \end{center}
4041 }
4042
4043 \@ifpackageloaded{stex}{
4044     \RequirePackage{stex-tikzinput}
4045 }{}
4046
4047 </package>
4048 <*stex>
4049 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4050 \RequirePackage{stex}
4051 \RequirePackage{tikzinput}
4052
4053 \newcommand\mhtikzinput [2] [] {%
4054     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4055     \stex_in_repository:nn\Gin@mhrepos{
4056         \tikzinput [1]{\mhpath{##1}{#2}}
4057     }
4058 }
4059 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4060 </stex>

```

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

## Chapter 31

# document-structure.sty Implementation

### 31.1 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.

```
4061 \*cls)
4062 \@@=document_structure)
4063 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4064 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

### 31.2 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`

```
4065 \keys_define:nn{ document-structure / pkg }{
4066   class      .str_set_x:N = \c_document_structure_class_str,
4067   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4068   report     .code:n      = {
4069     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4070     \str_set:Nn \c_document_structure_class_str {report}
4071   },
4072   book       .code:n      = {
4073     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4074     \str_set:Nn \c_document_structure_class_str {book}
4075   },
4076   bookpart   .code:n      = {
4077     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4078     \str_set:Nn \c_document_structure_class_str {book}
4079     \str_set:Nn \c_document_structure_topsect_str {chapter}
4080   },
```

```

4081 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4082 unknown     .code:n      = {
4083   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4084 }
4085 }
4086 \ProcessKeysOptions{ document-structure / pkg }
4087 \str_if_empty:NT \c_document_structure_class_str {
4088   \str_set:Nn \c_document_structure_class_str {article}
4089 }
4090 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4091   {\c_document_structure_class_str}
4092

```

### 31.3 Beefing up the document environment

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

```

4093 \RequirePackage{omdoc}
4094 \bool_if:NF \c_document_structure_minimal_bool {
4095   \RequirePackage{stex-compatibility}

```

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

**document** For the moment we do not use them on the L<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>15</sup>

```

4096 \keys_define:nn { document-structure / document }{
4097   id .str_set_x:N = \c_document_structure_document_id_str
4098 }
4099 \let\__document_structure_orig_document=\document
4100 \renewcommand{\document}[1][]{
4101   \keys_set:nn{ document-structure / document }{ #1 }
4102   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4103   \__document_structure_orig_document
4104 }

```

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

```

4105 }
4106 \</cls>

```

### 31.4 Implementation: OMDoc Package

```

4107 \*package>
4108 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4109 \RequirePackage{expl-keystr-compat,13keys2e}

```

### 31.5 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).

---

<sup>15</sup>EdNOTE: faking documentkeys for now. @HANG, please implement

```

4110
4111 \keys_define:nn{ document-structure / pkg }{
4112   class      .str_set_x:N = \c_document_structure_class_str,
4113   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4114   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4115 }
4116 \ProcessKeysOptions{ document-structure / pkg }
4117 \str_if_empty:NT \c_document_structure_class_str {
4118   \str_set:Nn \c_document_structure_class_str {article}
4119 }
4120 \str_if_empty:NT \c_document_structure_topsect_str {
4121   \str_set:Nn \c_document_structure_topsect_str {section}
4122 }

```

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

```

4123 \RequirePackage{xspace}
4124 \RequirePackage{comment}
4125 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

4126 \@ifpackageloaded{babel}{
4127   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4128   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4129     \input{omdoc-ngerman.ldf}
4130   }
4131 }{}
4132 %\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.

```

4133 \int_new:N \l_document_structure_section_level_int
4134 \str_case:VnF \c_document_structure_topsect_str {
4135   {part}{
4136     \int_set:Nn \l_document_structure_section_level_int {0}
4137   }
4138   {chapter}{
4139     \int_set:Nn \l_document_structure_section_level_int {1}
4140   }
4141 }{
4142   \str_case:VnF \c_document_structure_class_str {
4143     {book}{
4144       \int_set:Nn \l_document_structure_section_level_int {0}
4145     }
4146     {report}{
4147       \int_set:Nn \l_document_structure_section_level_int {0}
4148     }
4149   }{
4150     \int_set:Nn \l_document_structure_section_level_int {2}
4151   }
4152 }

```

## 31.6 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>16</sup>

EdN:16

```
4153 \def\current@section@level{document}%
4154 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4155 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4156 \cs_new_protected:Npn \skipomgroup {
4157   \ifcase\l_document_structure_section_level_int
4158   \or\stepcounter{part}
4159   \or\stepcounter{chapter}
4160   \or\stepcounter{section}
4161   \or\stepcounter{subsection}
4162   \or\stepcounter{subsubsection}
4163   \or\stepcounter{paragraph}
4164   \or\stepcounter{subparagraph}
4165   \fi
4166 }
```

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

`blindomgroup`

```
4167 \newcommand\at@begin@blindomgroup[1]{%
4168 \newenvironment{blindomgroup}
4169 {
4170   \int_incr:N\l_document_structure_section_level_int
4171   \at@begin@blindomgroup\l_document_structure_section_level_int
4172 }{}}
```

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

```
4173 \newcommand\omgroup@nonum[2]{
4174   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4175   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4176 }
```

*(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.

```
4177 \newcommand\omgroup@num[2]{
```

<sup>16</sup>EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.



```

4178 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4179   \@nameuse{#1}{#2}
4180 }{
4181   \cs_if_exist:NTF\rdfmata@sectioning{
4182     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4183   }{
4184     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4185   }
4186 }
4187 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4188 }

```

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

omgroup

```

4189 \keys_define:nn { document-structure / omgroup }{
4190   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4191   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4192   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4193   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4194   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4195   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4196   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4197   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4198   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4199   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4200 }
4201 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
4202   \str_clear:N \l__document_structure_omgroup_id_str
4203   \str_clear:N \l__document_structure_omgroup_date_str
4204   \clist_clear:N \l__document_structure_omgroup_creators_clist
4205   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4206   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4207   \tl_clear:N \l__document_structure_omgroup_type_tl
4208   \tl_clear:N \l__document_structure_omgroup_short_tl
4209   \tl_clear:N \l__document_structure_omgroup_display_tl
4210   \tl_clear:N \l__document_structure_omgroup_intro_tl
4211   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4212   \keys_set:nn { document-structure / omgroup } { #1 }
4213 }

```

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.

```

4214 \newif\if@mainmatter\@mainmattertrue
4215 \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.

```

4216 \keys_define:nn { document-structure / sectioning }{
4217   name .str_set_x:N = \l__document_structure_sect_name_str ,
4218   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4219   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4220   num .bool_set:N = \l__document_structure_sect_num_bool ,
4221 }

```

```

4222 \cs_new_protected:Nn \__document_structure_sect_args:n {
4223   \str_clear:N \l__document_structure_sect_name_str
4224   \str_clear:N \l__document_structure_sect_ref_str
4225   \bool_set_false:N \l__document_structure_sect_clear_bool
4226   \bool_set_false:N \l__document_structure_sect_num_bool
4227   \keys_set:nn { document-structure / sectioning } { #1 }
4228 }
4229 \newcommand\omdoc@sectioning[3][]{
4230   \__document_structure_sect_args:n {#1}
4231   \let\omdoc@sect@name\l__document_structure_sect_name_str
4232   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4233   \if@mainmatter% numbering not overridden by frontmatter, etc.
4234     \bool_if:NTF \l__document_structure_sect_num_bool {
4235       \omgroup@num{#2}{#3}
4236     }{
4237       \omgroup@nonum{#2}{#3}
4238     }
4239     \def\current@section@level{\omdoc@sect@name}
4240   \else
4241     \omgroup@nonum{#2}{#3}
4242   \fi
4243 }% 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.

```

4244 \newcommand\omgroup@redefine@addtocontents[1]{%
4245   %\edef\__document_structureimport{#1}%
4246   %\@for\@I:=\__document_structureimport\do{%
4247     %\edef\@path{\csname module@\@I @path\endcsname}%
4248     %\@ifundefined{tf@toc}\relax%
4249     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4250   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4251   %\def\addcontentsline##1##2##3{%
4252     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4253   %\else% hyperref.sty not loaded
4254   %\def\addcontentsline##1##2##3{%
4255     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4256   %\fi
4257 }% 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.

```

4258 \int_new:N \l_document_structure_omgroup_level_int
4259 \newenvironment{omgroup}[2][]{% keys, title
4260 {
4261   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

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

```

4262 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4263   \omgroup@redefine@addtocontents{
4264     %\@ifundefined{module@id}\used@modules%
4265     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4266     }
4267 }

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

4268 \int_incr:N \l_document_structure_omgroup_level_int
4269 \int_incr:N \l_document_structure_section_level_int
4270 \ifcase\l_document_structure_section_level_int
4271   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4272   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4273   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4274   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4275   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4276   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4277   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4278 \fi
4279 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4280 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4281 }% for customization
4282 {}

```

and finally, we localize the sections

```

4283 \newcommand\omdoc@part@kw{Part}
4284 \newcommand\omdoc@chapter@kw{Chapter}
4285 \newcommand\omdoc@section@kw{Section}
4286 \newcommand\omdoc@subsection@kw{Subsection}
4287 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4288 \newcommand\omdoc@paragraph@kw{paragraph}
4289 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 31.7 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`

```

4290 \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).

```

4291 \cs_if_exist:NTF\frontmatter{
4292   \let\__document_structure_orig_frontmatter\frontmatter
4293   \let\frontmatter\relax
4294 }{
4295   \tl_set:Nn\__document_structure_orig_frontmatter{
4296     \clearpage
4297     \@mainmatterfalse
4298     \pagenumbering{roman}
4299   }
4300 }
4301 \cs_if_exist:NTF\backmatter{

```

```

4302 \let\__document_structure_orig_backmatter\backmatter
4303 \let\backmatter\relax
4304 }{
4305 \tl_set:Nn\__document_structure_orig_backmatter{
4306 \clearpage
4307 \@mainmatterfalse
4308 \pagenumbering{roman}
4309 }
4310 }

```

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.

```

4311 \newenvironment{frontmatter}{
4312 \__document_structure_orig_frontmatter
4313 }{
4314 \cs_if_exist:NTF\mainmatter{
4315 \mainmatter
4316 }{
4317 \clearpage
4318 \@mainmattertrue
4319 \pagenumbering{arabic}
4320 }
4321 }

```

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

```

4322 \newenvironment{backmatter}{
4323 \__document_structure_orig_backmatter
4324 }{
4325 \cs_if_exist:NTF\mainmatter{
4326 \mainmatter
4327 }{
4328 \clearpage
4329 \@mainmattertrue
4330 \pagenumbering{arabic}
4331 }
4332 }

```

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

```

4333 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4334 \newcommand\afterprematurestop{}
4335 \def\prematurestop@endomgroup{
4336 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
4337 \end{omgroup}
4338 \prematurestop@endomgroup
4339 }
4340 }
4341 \providecommand\prematurestop{
4342 \message{Stopping~sTeX~processing~prematurely}

```

```

4343 \prematuarestop@endomgroup
4344 \afterprematuarestop
4345 \end{document}
4346 }

```

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

## 31.8 Global Variables

**\setSGvar** set a global variable

```

4347 \RequirePackage{etoolbox}
4348 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

4349 \newrobustcmd\useSGvar[1]{%
4350 \@ifundefined{sTeX@Gvar@#1}
4351 {\PackageError{omdoc}
4352 {The sTeX Global variable #1 is undefined}
4353 {set it with \protect\setSGvar}}
4354 \@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.

```

4355 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4356 \@ifundefined{sTeX@Gvar@#1}
4357 {\PackageError{omdoc}
4358 {The sTeX Global variable #1 is undefined}
4359 {set it with \protect\setSGvar}}
4360 {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

## Chapter 32

# MiKoSlides – Implementation

### 32.1 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.

```
4361 \*cls)
4362 \@@=mikoslides)
4363 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4364 \RequirePackage{l3keys2e,expl-keystr-compat}
4365
4366 \keys_define:nn{mikoslides / cls}{
4367   class .code:n = {
4368     \PassOptionsToClass{\CurrentOption}{omdoc}
4369     \str_if_eq:nnT{#1}{book}{
4370       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4371     }
4372     \str_if_eq:nnT{#1}{report}{
4373       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4374     }
4375   },
4376   notes .bool_set:N = \c__mikoslides_notes_bool ,
4377   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4378   unknown .code:n = {
4379     \PassOptionsToClass{\CurrentOption}{omdoc}
4380     \PassOptionsToClass{\CurrentOption}{beamer}
4381     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4382   }
4383 }
4384 \ProcessKeysOptions{ mikoslides / cls }
4385 \bool_if:NTF \c__mikoslides_notes_bool {
4386   \PassOptionsToPackage{notes=true}{mikoslides}
4387 }{
4388   \PassOptionsToPackage{notes=false}{mikoslides}
4389 }
4390 \</cls)
```

now we do the same for the mikoslides package.

```

4391 <*package>
4392 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4393 \RequirePackage{l3keys2e,expl-keystr-compat}
4394
4395 \keys_define:nn{mikoslides / pkg}{
4396   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4397   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4398   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4399   slides        .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4400   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4401   frameimages .bool_set:N = \c__mikoslides_frameimages_bool ,
4402   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4403   nopproblems .bool_set:N = \c__mikoslides_nopproblems_bool,
4404   unknown      .code:n      = {
4405     \PassOptionsToClass{\CurrentOption}{stex}
4406     \PassOptionsToClass{\CurrentOption}{tikzinput}
4407   }
4408 }
4409 \ProcessKeysOptions{ mikoslides / pkg }
4410 \newif\ifnotes
4411 \bool_if:NTF \c__mikoslides_notes_bool {
4412   \notesttrue
4413 }{
4414   \notesfalse
4415 }
4416

```

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

```

4417 \str_if_empty:NTF \c__mikoslides_topsect_str {
4418   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4419 }{
4420   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4421 }
4422 </package>

```

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

```

4423 <*cls>
4424 \bool_if:NTF \c__mikoslides_notes_bool {
4425   \LoadClass{omdoc}
4426 }{
4427   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4428   \newcounter{Item}
4429   \newcounter{paragraph}
4430   \newcounter{subparagraph}
4431   \newcounter{Hfootnote}
4432   \RequirePackage{omdoc}
4433 }

```

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

```

4434 \RequirePackage{mikoslides}
4435 </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.

```

4436 \*package>
4437 \bool_if:NT \c__mikoslides_notes_bool {
4438   \RequirePackage{a4wide}
4439   \RequirePackage{marginnote}
4440   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4441   \RequirePackage{mdframed}
4442   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4443   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4444 }
4445 \RequirePackage{stex-compatibility}
4446 \RequirePackage{stex-tikzinput}
4447 \RequirePackage{etoolbox}
4448 \RequirePackage{amssymb}
4449 \RequirePackage{amsmath}
4450 \RequirePackage{comment}
4451 \RequirePackage{textcomp}
4452 \RequirePackage{url}
4453 \RequirePackage{graphicx}
4454 \RequirePackage{pgf}

```

## 32.2 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>17</sup>

```

4455 \bool_if:NT \c__mikoslides_notes_bool {
4456   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4457 }

```

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

```

4458 \newcounter{slide}
4459 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4460 \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.

```

4461 \bool_if:NTF \c__mikoslides_notes_bool {
4462   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4463 }{
4464   \excludecomment{note}
4465 }

```

---

<sup>17</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 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.

```
4466 \bool_if:NT \c__mikoslides_notes_bool {
4467   \newlength{\slideframewidth}
4468   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
4469 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4470   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4471     \bool_set_true:N #1
4472   }{
4473     \bool_set_false:N #1
4474   }
4475 }
4476 \keys_define:nn{mikoslides / frame}{
4477   label .str_set_x:N = \l__mikoslides_frame_label_str,
4478   allowframebreaks .code:n = {
4479     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4480   },
4481   allowdisplaybreaks .code:n = {
4482     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4483   },
4484   fragile .code:n = {
4485     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4486   },
4487   shrink .code:n = {
4488     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4489   },
4490   squeeze .code:n = {
4491     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4492   },
4493   t .code:n = {
4494     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4495   },
4496 }
4497 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4498   \str_clear:N \l__mikoslides_frame_label_str
4499   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4500   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4501   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4502   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4503   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4504   \bool_set_true:N \l__mikoslides_frame_t_bool
4505   \keys_set:nn { mikoslides / frame }{ #1 }
4506 }
```

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

```
4507 \renewenvironment{frame}[1][]{
4508   \__mikoslides_frame_args:n{#1}
4509   \sffamily
4510   \stepcounter{slide}
4511   \def\@currentlabel{\theslide}
4512   \str_if_empty:NF \l__mikoslides_frame_label_str {
4513     \label{\l__mikoslides_frame_label_str}
```

```
4514 }
```

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

```
4515 \def\itemize@level{outer}
4516 \def\itemize@outer{outer}
4517 \def\itemize@inner{inner}
4518 \renewcommand\newpage{\addtocounter{framenum}{1}}
4519 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4520 \renewenvironment{itemize}{
4521   \ifx\itemize@level\itemize@outer
4522     \def\itemize@label{$\rhd$}
4523   \fi
4524   \ifx\itemize@level\itemize@inner
4525     \def\itemize@label{$\scriptstyle\rhd$}
4526   \fi
4527   \begin{list}
4528   {\itemize@label}
4529   {\setlength{\labelsep}{.3em}
4530    \setlength{\labelwidth}{.5em}
4531    \setlength{\leftmargin}{1.5em}
4532   }
4533   \edef\itemize@level{\itemize@inner}
4534 }{
4535   \end{list}
4536 }
```

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

```
4537 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4538 }{
4539   \medskip\miko@slidelabel\end{mdframed}
4540 }
```

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

`\frametitle`

```
4541 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4542 }
```

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

EdN:18

`\pause` 18

```
4543 \bool_if:NT \c__mikoslides_notes_bool {
4544   \newcommand\pause{}
4545 }
```

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

`nomtext`

```
4546 \bool_if:NTF \c__mikoslides_notes_bool {
4547   \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
4548 }{
4549   \excludecomment{nomtext}
4550 }
```

---

<sup>18</sup>EdNOTE: MK: fake it in notes mode for now

nomgroup

```
4551 \bool_if:NTF \c__mikoslides_notes_bool {
4552   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4553 }{
4554   \excludecomment{nomgroup}
4555 }
```

ndefinition

```
4556 \bool_if:NTF \c__mikoslides_notes_bool {
4557   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
4558 }{
4559   \excludecomment{ndefinition}
4560 }
```

nassertion

```
4561 \bool_if:NTF \c__mikoslides_notes_bool {
4562   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
4563 }{
4564   \excludecomment{nassertion}
4565 }
```

nsproof

```
4566 \bool_if:NTF \c__mikoslides_notes_bool {
4567   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4568 }{
4569   \excludecomment{nsproof}
4570 }
```

nexample

```
4571 \bool_if:NTF \c__mikoslides_notes_bool {
4572   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4573 }{
4574   \excludecomment{nexample}
4575 }
```

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

```
4576 \def\inputref@preskip{\smallskip}
4577 \def\inputref@postskip{\medskip}
```

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

\inputref\*

```
4578 \let\orig@inputref\inputref
4579 \def\inputref{\@ifstar\ninputref\orig@inputref}
4580 \newcommand\ninputref[2] [] {
4581   \bool_if:NT \c__mikoslides_notes_bool {
4582     \orig@inputref[#1]{#2}
4583   }
4584 }
```

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

## 32.3 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>}`.

```

4585 \newlength{\slidelogoheight}
4586
4587 \bool_if:NTF \c__mikoslides_notes_bool {
4588   \setlength{\slidelogoheight}{.4cm}
4589 }{
4590   \setlength{\slidelogoheight}{1cm}
4591 }
4592 \newsavebox{\slidelogo}
4593 \sbox{\slidelogo}{\TeX}
4594 \newrobustcmd{\setslidelogo}[1]{
4595   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4596 }
```

(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.

```

4597 \def\source{Michael Kohlhase}% customize locally
4598 \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.

```

4599 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4600 \newsavebox{\cclogo}
4601 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4602 \newif\ifcchref\cchreffalse
4603 \AtBeginDocument{
4604   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4605 }
4606 \def\licensing{
4607   \ifcchref
4608     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4609   \else
4610     {\usebox{\cclogo}}
4611   \fi
4612 }
4613 \newrobustcmd{\setlicensing}[2][]{
4614   \def\@url{#1}
4615   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4616   \ifx\@url\@empty
4617     \def\licensing{{\usebox{\cclogo}}}
4618   \else
4619     \def\licensing{
```

```

4620     \ifcchref
4621     \href{#1}{\usebox{\cclogo}}
4622     \else
4623     {\usebox{\cclogo}}
4624     \fi
4625   }
4626   \fi
4627 }

```

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

EdN:19

`\slidelabel` Now, we set up the slide label for the article mode.<sup>19</sup>

```

4628 \newrobustcmd\miko@slidelabel{
4629   \vbox to \slidelogoheight{
4630     \vss\hbox to \slidewidth
4631     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4632   }
4633 }

```

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

## 32.4 Frame Images

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

```

4634 \def\Gin@mhrepos{}
4635 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4636 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
4637 \newrobustcmd\frameimage[2][{}{
4638   \stepcounter{slide}
4639   \bool_if:NT \c__mikoslides_frameimages_bool {
4640     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4641     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4642     \begin{center}
4643       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4644         \fbox{
4645           \ifx\Gin@ewidth\@empty
4646             \ifx\Gin@mhrepos\@empty
4647               \mhgraphics[width=\slidewidth,#1]{#2}
4648             \else
4649               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4650             \fi
4651           \else% \Gin@ewidth empty
4652             \ifx\Gin@mhrepos\@empty
4653               \mhgraphics[#1]{#2}
4654             \else
4655               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4656             \fi
4657           \fi% \Gin@ewidth empty
4658         }
4659       }{
4660         \ifx\Gin@ewidth\@empty

```

---

<sup>19</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

4661         \ifx\Gin@mhrepos\empty
4662           \mhgraphics[width=\slidewidth,#1]{#2}
4663         \else
4664           \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4665         \fi
4666         \ifx\Gin@mhrepos\empty
4667           \mhgraphics[#1]{#2}
4668         \else
4669           \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4670         \fi
4671       \fi% Gin@ewidth empty
4672     }
4673   \end{center}
4674   \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4675   \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4676 }
4677 } % ifmks@sty@frameimages

```

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

## 32.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

4678 \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

```

4679 \AddToHook{begindocument}{
4680   \definecolor{green}{rgb}{0,.5,0}
4681   \definecolor{purple}{cmyk}{.3,1,0,.17}
4682 }

```

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

```

4683 % \def\STpresent#1{\textcolor{blue}{#1}}
4684 \def\defemph#1{\textcolor{magenta}{#1}}
4685 \def\symrefemph#1{\textcolor{cyan}{#1}}
4686 \def\compemph#1{\textcolor{blue}{#1}}
4687 \def\titleemph#1{\textcolor{blue}{#1}}
4688 \def\__omtext_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.

```

4689 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4690 \def\smalltextwarning{
4691   \pgfuseimage{miko@small@dbend}
4692   \xspace
4693 }
4694 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4695 \newrobustcmd\textwarning{
4696   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4697   \xspace
4698 }
4699 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4700 \newrobustcmd\bigtextwarning{
4701   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4702   \xspace
4703 }

```

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

```

4704 \newrobustcmd\putgraphicsat[3]{
4705   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4706 }
4707 \newrobustcmd\putat[2]{
4708   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4709 }

```

## 32.6 Sectioning

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

```

4710 \bool_if:NT \c__mikoslides_sectocframes_bool {
4711   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4712     \newcounter{chapter}\counterwithin*{section}{chapter}
4713   }{
4714     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4715       \newcounter{chapter}\counterwithin*{section}{chapter}
4716     }
4717   }
4718 }

```

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

```

\section@level
4719 \def\part@prefix{}
4720 \@ifpackageloaded{omdoc}{}{
4721   \str_case:VnF \__mikoslidestopsect {
4722     {part}{
4723       \int_set:Nn \l_document_structure_section_level_int {0}
4724       \def\thesection{\arabic{chapter}.\arabic{section}}
4725       \def\part@prefix{\arabic{chapter}.}
4726     }
4727     {chapter}{
4728       \int_set:Nn \l_document_structure_section_level_int {1}
4729       \def\thesection{\arabic{chapter}.\arabic{section}}
4730       \def\part@prefix{\arabic{chapter}.}
4731     }
4732   }{
4733     \int_set:Nn \l_document_structure_section_level_int {2}
4734     \def\part@prefix{}

```

```

4735 }
4736 }
4737
4738 \bool_if:NF \c__mikoslides_notes_bool { % only in slides

```

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

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

omgroup

```

4739 \renewenvironment{omgroup}[2][]{
4740   \__document_structure_omgroup_args:n { #1 }
4741   \int_incr:N \l_document_structure_omgroup_level_int
4742   \int_incr:N \l_document_structure_section_level_int
4743   \bool_if:NT \c__mikoslides_sectocframes_bool {
4744     \stepcounter{slide}
4745     \begin{frame}[noframenumbering]
4746     \vfill\Large\centering
4747     \red{
4748       \ifcase\l_document_structure_section_level_int\or
4749         \stepcounter{part}
4750         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4751         \def\currentsectionlevel{\omdoc@part@kw}
4752       \or
4753         \stepcounter{chapter}
4754         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4755         \def\currentsectionlevel{\omdoc@chapter@kw}
4756       \or
4757         \stepcounter{section}
4758         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4759         \def\currentsectionlevel{\omdoc@section@kw}
4760       \or
4761         \stepcounter{subsection}
4762         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4763         \def\currentsectionlevel{\omdoc@subsection@kw}
4764       \or
4765         \stepcounter{subsubsection}
4766         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4767         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4768       \or
4769         \stepcounter{paragraph}
4770         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
4771         \def\currentsectionlevel{\omdoc@paragraph@kw}
4772       \else
4773         \def\__mikoslideslabel{}
4774         \def\currentsectionlevel{\omdoc@paragraph@kw}
4775       \fi% end ifcase
4776       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4777       \quad #2%
4778     }%
4779     \vfill%
4780     \end{frame}%
4781   }
4782   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```



```

4783 }{}
4784 }

```

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

```

4785 \def\inserttheorembodyfont{\normalfont}
4786 \bool_if:NF \c__mikoslides_notes_bool {
4787   \defbeamertemplate{theorem begin}{miko}
4788   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4789     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4790     \inserttheorempunctuation\inserttheorembodyfont\space}
4791   \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

4792   \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.

```

4793   \expandafter\def\csname Parent2\endcsname{}
4794 }
4795 \bool_if:NT \c__mikoslides_notes_bool {
4796   \renewenvironment{columns}[1][]{%
4797     \par\noindent%
4798     \begin{minipage}%
4799       \slidewidth\centering\leavevmode%
4800   }{%
4801     \end{minipage}\par\noindent%
4802   }%
4803   \newsavebox\columnbox%
4804   \renewenvironment<>{column}[2][]{%
4805     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4806   }{%
4807     \end{minipage}\end{lrbox}\usebox\columnbox%
4808   }%
4809 }
4810 \bool_if:NTF \c__mikoslides_noproblems_bool {
4811   \newenvironment{problems}{}{}
4812 }{
4813   \excludecomment{problems}
4814 }

```

## 32.7 Excursions

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

```

4815 \gdef\printexcursions{}
4816 \newcommand\excursionref[2]{% label, text
4817   \bool_if:NT \c__mikoslides_notes_bool {
4818     \begin{sparagraph}[title=Excursion]
4819       #2 \sref[fallback=the appendix]{#1}.
4820     \end{sparagraph}
4821   }

```

```

4822 }
4823 \newcommand\activate@excursion[2][]{
4824   \gappto\printexcursions{\inputref{#1}{#2}}
4825 }
4826 \newcommand\excursion[4][]{% repos, label, path, text
4827   \bool_if:NT \c__mikoslides_notes_bool {
4828     \activate@excursion[1]{#3}\excursionref{#2}{#4}
4829   }
4830 }

```

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

\excursiongroup

```

4831 \keys_define:nn{mikoslides / excursiongroup }{
4832   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4833   intro       .tl_set:N    = \l__mikoslides_excursion_intro_tl,
4834   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4835 }
4836 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4837   \tl_clear:N \l__mikoslides_excursion_intro_tl
4838   \str_clear:N \l__mikoslides_excursion_id_str
4839   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4840   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4841 }
4842 \newcommand\excursiongroup[1][]{
4843   \__mikoslides_excursion_args:n{ #1 }
4844   \ifdefempty\printexcursions{}% only if there are excursions
4845   {\begin{note}
4846     \begin{omgroup}[#1]{Excursions}%
4847     \ifdefempty\l__mikoslides_excursion_intro_tl{\{
4848       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4849         \l__mikoslides_excursion_intro_tl
4850       }
4851     }
4852     \printexcursions%
4853     \end{omgroup}
4854   \end{note}}
4855 }
4856 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
4857 \</package>

```

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

## Chapter 33

# The Implementation

### 33.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
4858 <*package>
4859 <@@=problems>
4860 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4861 \RequirePackage{l3keys2e,expl-keystr-compatible}
4862
4863 \keys_define:nn { problem / pkg }{
4864   notes      .default:n    = { true },
4865   notes      .bool_set:N   = \c__problems_notes_bool,
4866   gnotes     .default:n    = { true },
4867   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4868   hints      .default:n    = { true },
4869   hints      .bool_set:N   = \c__problems_hints_bool,
4870   solutions  .default:n    = { true },
4871   solutions  .bool_set:N   = \c__problems_solutions_bool,
4872   pts        .default:n    = { true },
4873   pts        .bool_set:N   = \c__problems_pts_bool,
4874   min        .default:n    = { true },
4875   min        .bool_set:N   = \c__problems_min_bool,
4876   boxed      .default:n    = { true },
4877   boxed      .bool_set:N   = \c__problems_boxed_bool,
4878   unknown    .code:n       = {}
4879 }
4880 \def\solutionstrue{
4881   \bool_set_true:N \c__problems_solutions_bool
4882 }
4883 \def\solutionsfalse{
4884   \bool_set_false:N \c__problems_solutions_bool
4885 }
4886
4887 \ProcessKeysOptions{ problem / pkg }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```

4888 \RequirePackage{stex-compatibility}
4889 \RequirePackage{comment}

```

The next package relies on the L<sup>A</sup>T<sub>E</sub>X3 kernel, which L<sup>A</sup>T<sub>E</sub>XML only partially supports. As it is purely presentational, we only load it when the `boxed` option is given and we run L<sup>A</sup>T<sub>E</sub>XML.

```

4890 \bool_if:NT \c__problems_boxed_bool { \RequirePackage{mdframed} }

```

`\prob@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```

4891 \def\prob@problem@kw{Problem}
4892 \def\prob@solution@kw{Solution}
4893 \def\prob@hint@kw{Hint}
4894 \def\prob@note@kw{Note}
4895 \def\prob@gnote@kw{Grading}
4896 \def\prob@pt@kw{pt}
4897 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4898 \@ifpackageloaded{babel}{
4899   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4900   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4901     \input{problem-ngerman.ldf}
4902   }
4903   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4904     \input{problem-finnish.ldf}
4905   }
4906   \clist_if_in:NnT \l_tmpa_clist {french}{
4907     \input{problem-french.ldf}
4908   }
4909   \clist_if_in:NnT \l_tmpa_clist {russian}{
4910     \input{problem-russian.ldf}
4911   }
4912 }{}

```

## 33.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```

4913 \keys_define:nn{ problem / problem }{
4914   id      .str_set:x:N = \l__problems_prob_id_str,
4915   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4916   min     .tl_set:N    = \l__problems_prob_min_tl,
4917   title   .tl_set:N    = \l__problems_prob_title_tl,
4918   refnum  .int_set:N   = \l__problems_prob_refnum_int
4919 }
4920 \cs_new_protected:Nn \__problems_prob_args:n {
4921   \str_clear:N \l__problems_prob_id_str
4922   \tl_clear:N \l__problems_prob_pts_tl
4923   \tl_clear:N \l__problems_prob_min_tl
4924   \tl_clear:N \l__problems_prob_title_tl

```

```

4925 \int_zero_new:N \l__problems_prob_refnum_int
4926 \keys_set:nn { problem / problem }{ #1 }
4927 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4928   \let\l__problems_inclprob_refnum_int\undefined
4929 }
4930 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4931 \newcounter{problem}
4932 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}

```

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

`\prob@label` We provide the macro `\prob@label` to redefine later to get context involved.

```

4933 \newcommand\prob@label[1]{#1}

```

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

`\prob@number` We consolidate the problem number into a reusable internal macro

```

4934 \newcommand\prob@number{
4935   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4936     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4937   }{
4938     \int_if_exist:NTF \l__problems_prob_refnum_int {
4939       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4940     }{
4941       \prob@label\theproblem
4942     }
4943   }
4944 }

```

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

`\prob@title` We consolidate the problem title into a reusable internal macro as well. `\prob@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```

4945 \newcommand\prob@title[3]{%
4946   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4947     #2 \l__problems_inclprob_title_tl #3
4948   }{
4949     \tl_if_exist:NTF \l__problems_prob_title_tl {
4950       #2 \l__problems_prob_title_tl #3
4951     }{
4952       #1
4953     }
4954   }
4955 }

```

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

With these the problem header is a one-liner

`\prob@heading` We consolidate the problem header line into a separate internal macro that can be reused in various settings.

```

4956 \def\prob@heading{
4957   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4958   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
4959 }

```

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

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`problem`

```

4960 \newenvironment{problem}[1][1]{
4961   \__problems_prob_args:n{#1}%\sref@target%
4962   \@in@omtexttrue% we are in a statement (for inline definitions)
4963   \stepcounter{problem}\record@problem
4964   \def\current@section@level{\prob@problem@kw}
4965   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4966 }%
4967 {\smallskip}
4968 \bool_if:NT \c__problems_boxed_bool {
4969   \surroundwithmdframed{problem}
4970 }

```

`\record@problem` This macro records information about the problems in the `*.aux` file.

```

4971 \def\record@problem{
4972   \protected@write\@auxout{}
4973   {
4974     \string\@problem{\prob@number}
4975     {
4976       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4977         \l__problems_inclprob_pts_tl
4978       }{
4979         \l__problems_prob_pts_tl
4980       }
4981     }%
4982     {
4983       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4984         \l__problems_inclprob_min_tl
4985       }{
4986         \l__problems_prob_min_tl
4987       }
4988     }
4989   }
4990 }

```

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

`\@problem` This macro acts on a problem's record in the `*.aux` file. It does not have any functionality here, but can be redefined elsewhere (e.g. in the `assignment` package).

```

4991 \def\@problem#1#2#3{}

```

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

**solution** The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```

4992 \keys_define:nn { problem / solution }{
4993   id          .str_set_x:N = \l__problems_solution_id_str ,
4994   for         .tl_set:N    = \l__problems_solution_for_tl ,
4995   height      .dim_set:N   = \l__problems_solution_height_dim ,
4996   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4997   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4998   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4999 }
5000 \cs_new_protected:Nn \__problems_solution_args:n {
5001   \str_clear:N \l__problems_solution_id_str
5002   \tl_clear:N \l__problems_solution_for_tl
5003   \tl_clear:N \l__problems_solution_srccite_tl
5004   \clist_clear:N \l__problems_solution_creators_clist
5005   \clist_clear:N \l__problems_solution_contributors_clist
5006   \dim_zero:N \l__problems_solution_height_dim
5007   \keys_set:nn { problem / solution }{ #1 }
5008 }

```

the next step is to define a helper macro that does what is needed to start a solution.

```

5009 \newcommand\@startsolution[1][ ]{
5010   \__problems_solution_args:n { #1 }
5011   \@in@omtexttrue% we are in a statement.
5012   \bool_if:NF \c__problems_boxed_bool { \hrule }
5013   \smallskip\noindent
5014   {\textbf\prob@solution@kw : \enspace}
5015   \begin{small}
5016   \def\current@section@level{\prob@solution@kw}
5017   \ignorespacesandpars
5018 }

```

**\startsolutions** for the `\startsolutions` macro we use the `\specialcomment` macro from the `comment` package. Note that we use the `\@startsolution` macro in the start codes, that parses the optional argument.

```

5019 \newcommand\startsolutions{
5020   \specialcomment{solution}{\@startsolution}{
5021     \bool_if:NF \c__problems_boxed_bool {
5022       \hrule\medskip
5023     }
5024     \end{small}%
5025   }
5026   \bool_if:NT \c__problems_boxed_bool {
5027     \surroundwithmdframed{solution}
5028   }
5029 }

```

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

**\stopsolutions**

```

5030 \newcommand\stopsolutions{\excludecomment{solution}}

```

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

so it only remains to start/stop solutions depending on what option was specified.

```

5031 \bool_if:NTF \c__problems_solutions_bool {
5032   \startsolutions
5033 }{
5034   \stopsolutions
5035 }

```

**exnote**

```

5046 \bool_if:NTF \c__problems_notes_bool {
5047   \newenvironment{exnote}[1][]{
5048     \par\smallskip\hrule\smallskip
5049     \noindent\textbf{\prob@note@kw : }\small
5050   }{
5051     \smallskip\hrule
5052   }
5053 }{
5054   \excludecomment{exnote}
5055 }

```

**hint**

```

5056 \bool_if:NTF \c__problems_notes_bool {
5057   \newenvironment{hint}[1][]{
5058     \par\smallskip\hrule\smallskip
5059     \noindent\textbf{\prob@hint@kw :~ }\small
5060   }{
5061     \smallskip\hrule
5062   }
5063 }{
5064   \newenvironment{exhint}[1][]{
5065     \par\smallskip\hrule\smallskip
5066     \noindent\textbf{\prob@hint@kw :~ }\small
5067   }{
5068     \smallskip\hrule
5069   }
5070 }{
5071   \excludecomment{hint}
5072   \excludecomment{exhint}
5073 }

```

**gnote**

```

5074 \bool_if:NTF \c__problems_notes_bool {
5075   \newenvironment{gnote}[1][]{
5076     \par\smallskip\hrule\smallskip
5077     \noindent\textbf{\prob@gnote@kw : }\small
5078   }{
5079     \smallskip\hrule
5080   }
5081 }{
5082   \excludecomment{gnote}
5083 }

```



### 33.3 Multiple Choice Blocks

```

5073 \newenvironment{mcb}{
5074   \begin{enumerate}
5075 }{
5076   \end{enumerate}
5077 }

```

we define the keys for the mcc macro

```

5078 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5079   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5080     \bool_set_true:N #1
5081   }{
5082     \bool_set_false:N #1
5083   }
5084 }
5085 \keys_define:nn { problem / mcc }{
5086   id          .str_set_x:N = \l__problems_mcc_id_str ,
5087   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5088   T           .default:n   = { true } ,
5089   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5090   F           .default:n   = { true } ,
5091   F           .bool_set:N   = \l__problems_mcc_f_bool ,
5092   Ttext       .code:n      = {
5093     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5094   } ,
5095   Ftext       .code:n      = {
5096     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5097   }
5098 }
5099 \cs_new_protected:Nn \l__problems_mcc_args:n {
5100   \str_clear:N \l__problems_mcc_id_str
5101   \tl_clear:N \l__problems_mcc_feedback_tl
5102   \bool_set_true:N \l__problems_mcc_t_bool
5103   \bool_set_true:N \l__problems_mcc_f_bool
5104   \bool_set_true:N \l__problems_mcc_Ttext_bool
5105   \bool_set_false:N \l__problems_mcc_Ftext_bool
5106   \keys_set:nn { problem / mcc }{ #1 }
5107 }

```

\mcc

```

5108 \newcommand\mcc[2][] {
5109   \l__problems_mcc_args:n{ #1 }
5110   \item #2
5111   \bool_if:NT \c__problems_solutions_bool {
5112     \
5113     \bool_if:NT \l__problems_mcc_t_bool {
5114       % TODO!
5115       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
5116     }
5117     \bool_if:NT \l__problems_mcc_f_bool {

```

---

<sup>20</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

5118         % TODO!
5119         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5120     }
5121     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5122         !
5123     }{
5124         \l__problems_mcc_feedback_tl
5125     }
5126 }
5127 } %solutions

```

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

## 33.4 Including Problems

`\includeproblem` The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

5128
5129 \keys_define:nn{ problem / inclproblem }{
5130 % id      .str_set_x:N = \l__problems_inclprob_id_str,
5131 pts      .tl_set:N     = \l__problems_inclprob_pts_tl,
5132 min      .tl_set:N     = \l__problems_inclprob_min_tl,
5133 title    .tl_set:N     = \l__problems_inclprob_title_tl,
5134 refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5135 mhrepos  .str_set_x:N  = \l__problems_inclprob_mhrepos_str
5136 }
5137 \cs_new_protected:Nn \__problems_inclprob_args:n {
5138 % \str_clear:N \l__problems_prob_id_str
5139 \tl_clear:N \l__problems_inclprob_pts_tl
5140 \tl_clear:N \l__problems_inclprob_min_tl
5141 \tl_clear:N \l__problems_inclprob_title_tl
5142 \int_zero_new:N \l__problems_inclprob_refnum_int
5143 \str_clear:N \l__problems_inclprob_mhrepos_str
5144 \keys_set:nn { problem / inclproblem }{ #1 }
5145 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5146     \let\l__problems_inclprob_pts_tl\undefined
5147 }
5148 \tl_if_empty:NT \l__problems_inclprob_min_tl {
5149     \let\l__problems_inclprob_min_tl\undefined
5150 }
5151 \tl_if_empty:NT \l__problems_inclprob_title_tl {
5152     \let\l__problems_inclprob_title_tl\undefined
5153 }
5154 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5155     \let\l__problems_inclprob_refnum_int\undefined
5156 }
5157 }
5158
5159 \cs_new_protected:Nn \__problems_inclprob_clear: {
5160 % \str_clear:N \l__problems_prob_id_str
5161 \let\l__problems_inclprob_pts_tl\undefined
5162 \let\l__problems_inclprob_min_tl\undefined

```

```

5163 \let\l__problems_inclprob_title_tl\undefined
5164 \let\l__problems_inclprob_refnum_int\undefined
5165 \let\l__problems_inclprob_mhrepos_str\undefined
5166 }
5167
5168 \newcommand\includeproblem[2][]{
5169   \__problems_inclprob_args:n{ #1 }
5170   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5171     \input{#2}
5172   }{
5173     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5174       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5175     }
5176   }
5177   \__problems_inclprob_clear:
5178 }

```

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

## 33.5 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

5179 \AddToHook{enddocument}{
5180   \bool_if:NT \c__problems_pts_bool {
5181     \message{Total:~\arabic{pts}~points}
5182   }
5183   \bool_if:NT \c__problems_min_bool {
5184     \message{Total:~\arabic{min}~minutes}
5185   }
5186 }

```

The margin pars are reader-visible, so we need to translate

```

5187 \def\pts#1{
5188   \bool_if:NT \c__problems_pts_bool {
5189     \marginpar{#1~\prob@pt@kw}
5190   }
5191 }
5192 \def\min#1{
5193   \bool_if:NT \c__problems_min_bool {
5194     \marginpar{#1~\prob@min@kw}
5195   }
5196 }

```

**\show@pts** The **\show@pts** shows the points: if no points are given from the outside and also no points are given locally do nothing, else show and add. If there are outside points then we show them in the margin.

```

5197 \newcounter{pts}
5198 \def\show@pts{
5199   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5200     \bool_if:NT \c__problems_pts_bool {
5201       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5202       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5203     }
5204   }{
5205     \tl_if_exist:NT \l__problems_prob_pts_tl {
5206       \bool_if:NT \c__problems_pts_bool {
5207         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5208         \addtocounter{pts}{\l__problems_prob_pts_tl}
5209       }
5210     }
5211   }
5212 }

```

(End definition for \show@pts. This function is documented on page ??.)  
and now the same for the minutes

\show@min

```

5213 \newcounter{min}
5214 \def\show@min{
5215   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5216     \bool_if:NT \c__problems_min_bool {
5217       \marginpar{\l__problems_inclprob_pts_tl;min}
5218       \addtocounter{min}{\l__problems_inclprob_min_tl}
5219     }
5220   }{
5221     \tl_if_exist:NT \l__problems_prob_min_tl {
5222       \bool_if:NT \c__problems_min_bool {
5223         \marginpar{\l__problems_prob_min_tl;min}
5224         \addtocounter{min}{\l__problems_prob_min_tl}
5225       }
5226     }
5227   }
5228 }
5229 \</package>

```

(End definition for \show@min. This function is documented on page ??.)

## Chapter 34

# Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

### 34.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
5230 <@@=hwexam>
5231 <*cls>
5232 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5233 \RequirePackage{l3keys2e,expl-keystr-compatible}
5234 \DeclareOption*{
5235   \PassOptionsToClass{\CurrentOption}{omdoc}
5236   \PassOptionsToPackage{\CurrentOption}{stex}
5237   \PassOptionsToPackage{\CurrentOption}{hwexam}
5238   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5239 }
5240 \ProcessOptions
```

We load `omdoc.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.

```
5241 \LoadClass{omdoc}
5242 \RequirePackage{stex}
5243 \RequirePackage{hwexam}
5244 \RequirePackage{tikzinput}
5245 \RequirePackage{graphicx}
5246 \RequirePackage{a4wide}
5247 \RequirePackage{amssymb}
5248 \RequirePackage{amstext}
5249 \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```

5250 \newcommand\assig@default@type{\hwexam@assignment@kw}
5251 \def\document@hwexamtype{\assig@default@type}
5252 <@@=document_structure>
5253 \keys_define:nn { document-structure / document }{
5254 id .str_set_x:N = \c_document_structure_document_id_str,
5255 hwexamtype .tl_set:N = \document@hwexamtype
5256 }
5257 <@@=hwexam>
5258 </cls>

```

## Chapter 35

# Implementation: The hwexam Package

### 35.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
5259 \*package>
5260 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5261 \RequirePackage{l3keys2e,expl-keystr-compat}
5262
5263 \newif\iftest\testfalse
5264 \DeclareOption{test}{\testtrue}
5265 \newif\ifmultiple\multiplefalse
5266 \DeclareOption{multiple}{\multipletrue}
5267 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5268 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
5269 \RequirePackage{keyval}[1997/11/10]
5270 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
5271 \newcommand\hwexam@assignment@kw{Assignment}
5272 \newcommand\hwexam@given@kw{Given}
5273 \newcommand\hwexam@due@kw{Due}
5274 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
5275   space}%
5276 \newcommand\correction@probs@kw{prob.}%
5277 \newcommand\correction@pts@kw{total}%
5278 \newcommand\correction@reached@kw{reached}%
5279 \newcommand\correction@sum@kw{Sum}%
5280 \newcommand\correction@grade@kw{grade}%
5281 \newcommand\correction@forgrading@kw{To be used for grading, do not write here}
```

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

For the other languages, we set up triggers

```

5282 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5283
5284 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5285 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5286   \input{hwexam-ngerman.ldf}
5287 }
5288 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5289   \input{hwexam-finnish.ldf}
5290 }
5291 \clist_if_in:NnT \l_tmpa_clist {french}{
5292   \input{hwexam-french.ldf}
5293 }
5294 \clist_if_in:NnT \l_tmpa_clist {russian}{
5295   \input{hwexam-russian.ldf}
5296 }

```

## 35.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

5297 \newcounter{assignment}
5298 \numberproblemsin{assignment}
5299 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

We will prepare the keyval support for the `assignment` environment.

```

5300 \keys_define:nn { hwexam / assignment } {
5301   id .str_set:N = \l__hwexam_assign_id_str,
5302   number .int_set:N = \l__hwexam_assign_number_int,
5303   title .tl_set:N = \l__hwexam_assign_title_tl,
5304   type .tl_set:N = \l__hwexam_assign_type_tl,
5305   given .tl_set:N = \l__hwexam_assign_given_tl,
5306   due .tl_set:N = \l__hwexam_assign_due_tl,
5307   loadmodules .code:n = {
5308     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5309   }
5310 }
5311 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5312   \str_clear:N \l__hwexam_assign_id_str
5313   \int_set:Nn \l__hwexam_assign_number_int {-1}
5314   \tl_clear:N \l__hwexam_assign_title_tl
5315   \tl_clear:N \l__hwexam_assign_type_tl
5316   \tl_clear:N \l__hwexam_assign_given_tl
5317   \tl_clear:N \l__hwexam_assign_due_tl
5318   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5319   \keys_set:nn { hwexam / assignment }{ #1 }
5320 }

```

The next three macros are intermediate functions that handle the case gracefully, where the respective token registers are undefined.



The `\given@due` macro prints information about the given and due status of the assignment. Its arguments specify the brackets.

```

5321 \newcommand\given@due[2]{
5322 \bool_lazy_all:nF {
5323 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5324 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5325 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5326 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5327 }{ #1 }
5328
5329 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5330 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5331 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5332 }
5333 }{
5334 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5335 }
5336
5337 \bool_lazy_or:nnF {
5338 \bool_lazy_and_p:nn {
5339 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5340 }{
5341 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5342 }
5343 }{
5344 \bool_lazy_and_p:nn {
5345 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5346 }{
5347 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5348 }
5349 }{ ,~ }
5350
5351 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5352 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5353 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5354 }
5355 }{
5356 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5357 }
5358
5359 \bool_lazy_all:nF {
5360 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5361 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5362 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5363 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5364 }{ #2 }
5365 }

```

`\assignment@title` This macro prints the title of an assignment, the local title is overwritten, if there is one from the `\inputassignment`. `\assignment@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```

5366 \newcommand\assignment@title[3]{

```

```

5367 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5368 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5369 #1
5370 }{
5371 #2\l__hwexam_assign_title_tl#3
5372 }
5373 }{
5374 #2\l__hwexam_inclassassign_title_tl#3
5375 }
5376 }

```

(End definition for \assignment@title. This function is documented on page ??.)

**\assignment@number** Like \assignment@title only for the number, and no around part.

```

5377 \newcommand\assignment@number{
5378 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5379 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5380 \int_use:N \l__hwexam_assign_number_int
5381 }
5382 }{
5383 \int_use:N \l__hwexam_inclassassign_number_int
5384 }
5385 }

```

(End definition for \assignment@number. This function is documented on page ??.)

With them, we can define the central **assignment** environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

**assignment** For the assignment environment we delegate the work to the @assignment environment that depends on whether multiple option is given.

```

5386 \newenvironment{assignment}[1][]{
5387 \__hwexam_assignment_args:n { #1 }
5388 %\sref@target
5389 \let\__hwexamnum\l__hwexam_assign_number_int
5390 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5391 \stepcounter{assignment}
5392 }{
5393 \setcounter{assignment}{\int_use:N\__hwexamnum}
5394 }
5395 \setcounter{problem}{0}
5396 \def\current@section@level{\document@hwexamtype}
5397 %\sref@label@id{\document@hwexamtype \thesection}
5398 \begin{@assignment}
5399 }{
5400 \end{@assignment}
5401 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

5402 \def\__hwexasstitle{
5403 \protect\document@hwexamtype~\arabic{assignment}
5404 \assignment@title{}\;{} \; -- \given@due{}\}
5405 }

```

```

5406 \ifmultiple
5407 \newenvironment{@assignment}{
5408 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5409 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5410 }{
5411 \begin{omgroup}{\__hwexasstitle}
5412 }
5413 }{
5414 \end{omgroup}
5415 }

```

for the single-page case we make a title block from the same components.

```

5416 \else
5417 \newenvironment{@assignment}{
5418 \begin{center}\bf
5419 \Large\@title\strut\
5420 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}
5421 \large\given@due{--\;}{\;}{--}
5422 \end{center}
5423 }{}
5424 \fi% multiple

```

## 35.3 Including Assignments

**\in\*assignment** This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

5425 \keys_define:nn { hwexam / inclassignment } {
5426 %id .str_set_x:N = \l__hwexam_assign_id_str,
5427 number .int_set:N = \l__hwexam_inclassign_number_int,
5428 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5429 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5430 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5431 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5432 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5433 }
5434 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5435 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5436 \tl_clear:N \l__hwexam_inclassign_title_tl
5437 \tl_clear:N \l__hwexam_inclassign_type_tl
5438 \tl_clear:N \l__hwexam_inclassign_given_tl
5439 \tl_clear:N \l__hwexam_inclassign_due_tl
5440 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5441 \keys_set:nn { hwexam / inclassignment }{ #1 }
5442 }
5443 \__hwexam_inclassignment_args:n {}
5444
5445 \newcommand\inputassignment[2][ ]{
5446 \__hwexam_inclassignment_args:n { #1 }
5447 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5448 \input{#2}
5449 }{
5450 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5451 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5452 }
5453 }
5454 \__hwexam_inclasssign_args:n {}
5455 }
5456 \newcommand\includeassignment[2][ ]{
5457 \newpage
5458 \inputassignment[#1]{#2}
5459 }

```

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

## 35.4 Typesetting Exams

\quizheading

```

5460 \ExplSyntaxOff
5461 \newcommand\quizheading[1]{%
5462 \def\@tas{#1}%
5463 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5464 \ifx\@tas\empty\else%
5465 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5466 \fi%
5467 }
5468 \ExplSyntaxOn

```

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

\testheading

```

5469 \keys_define:nn { hwexam / testheading } {
5470 min .tl_set:N = \l__hwexam_testheading_min_tl,
5471 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5472 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5473 }
5474 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5475 \tl_clear:N \l__hwexam_testheading_min_tl
5476 \tl_clear:N \l__hwexam_testheading_duration_tl
5477 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5478 \keys_set:nn { hwexam / testheading }{ #1 }
5479 }
5480 \newenvironment{testheading}[1][ ]{
5481 \__hwexam_testheading_args:n{ #1 }
5482 \noindent\large{Name:~\hfill
5483 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5484 \begin{center}
5485 \Large\textbf{\@title}\[1ex]
5486 \large\@date\[3ex]
5487 \end{center}
5488 \textbf{You~have~
5489 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5490 \l__hwexam_testheading_min_tl~minutes
5491 }{
5492 \l__hwexam_testheading_duration_tl
5493 }~

```

```

5494 (sharp)~for~the~test
5495 };\
5496 Write~the~solutions~to~the~sheet.
5497 \par\noindent
5498 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5499 \advance\check@time by -\theassignment@totalmin
5500 The~estimated~time~for~solving~this~exam~is~
5501 {\theassignment@totalmin}-minutes,~
5502 leaving~you~{\the\check@time}-minutes~for~revising~
5503 your~exam.
5504
5505 \par\noindent
5506 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5507 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5508 You~can~reach~{\theassignment@totalpts}-points~if~you~
5509 solve~all~problems.~You~will~only~need~
5510 {\l__hwexam_testheading_reqpts_tl}-points~for~a~perfect~score,~
5511 i.e.~\ {\the\bonus@pts}-points~are~bonus~points.
5512 \vfill
5513 \begin{center}
5514 {
5515 \Large\em You~have~ample~time,~so~take~it~slow~
5516 and~avoid~rushing~to~mistakes!\}[2ex]
5517 Different~problems~test~different~skills~and~
5518 knowledge,~so~do~not~get~stuck~on~one~problem.
5519 }
5520 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5521 \end{center}
5522 }{
5523 \newpage
5524 }

```

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

\testspace

```

5525 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5526 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5527 \newcommand\testemptypage[1][ ]{\iftest\begin{center}\hwexam@testemptypage@kw\end{center}\vfi

```

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

\@problem This macro acts on a problem's record in the \*.aux file. Here we redefine it (it was defined to do nothing in problem.sty) to generate the correction table.

```

5528 <@=problems>
5529 \renewcommand\@problem[3]{
5530 \stepcounter{assignment@probs}
5531 \def\__problemspts{#2}

```

```

5532 \ifx\__problemspts\@empty\else
5533 \addtocounter{assignment@totalpts}{#2}
5534 \fi
5535 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5536 \xdef\correction@probs{\correction@probs & #1}%
5537 \xdef\correction@pts{\correction@pts & #2}
5538 \xdef\correction@reached{\correction@reached & }
5539 }
5540 \@@=hwexam

```

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

**\correction@table** This macro generates the correction table

```

5541 \newcounter{assignment@probs}
5542 \newcounter{assignment@totalpts}
5543 \newcounter{assignment@totalmin}
5544 \def\correction@probs{\correction@probs@kw}%
5545 \def\correction@pts{\correction@pts@kw}%
5546 \def\correction@reached{\correction@reached@kw}%
5547 \def\after@correction@table{}%
5548 \stepcounter{assignment@probs}
5549 \newcommand\correction@table{
5550 \resizebox{\textwidth}{!}{%
5551 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5552 &\multicolumn{\theassignment@probs}{c|}|%|
5553 {\footnotesize\correction@forgrading@kw} &\\ \hline
5554 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5555 \correction@pts & \theassignment@totalpts & \\ \hline
5556 \correction@reached & & \[.7cm]\hline
5557 \end{tabular}}
5558 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5559 \end{package}

```

(End definition for \correction@table. This function is documented on page ??.)

## 35.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierrfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\bierrglas{{\bierrfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierrglas}

```