

# **stex.sty: $\text{\TeX}$ 2.0\***

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

August 20, 2021

## **Abstract**

TODO

## **1 Introduction**

TODO

---

\*Version v1.9 (last revised 2021/08/01)

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Manual</b>	<b>3</b>
2.1	Notations and Precedences . . . . .	3
2.2	Archives and Imports . . . . .	3
<b>3</b>	<b>Documentation</b>	<b>4</b>
3.1	Utils . . . . .	4
3.2	Files, Paths, URIs . . . . .	6
3.3	MathHub Archives . . . . .	7
3.4	The Module System . . . . .	8
3.5	Symbols and Terms . . . . .	14
<b>4</b>	<b>Implementation</b>	<b>17</b>
4.1	The $\text{\LaTeX}$ document class . . . . .	17
4.2	Preliminaries . . . . .	18
4.3	Files, Paths and URIs . . . . .	23
4.4	MathHub Repositories . . . . .	26
4.5	Module System . . . . .	29
4.6	Symbol Declarations . . . . .	44
4.7	Notations . . . . .	48
4.8	Terms . . . . .	56

## 2 Manual

### 2.1 Notations and 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 + #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.

$\text{\S}$ 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 *smaller* 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  $A$ ’s operator precedence should be larger than  $B$ ’s argument precedences.

For example, we could set

```
\notation[prec=50]{plus}{#1 + #2}
```

and

```
\notation[prec=100]{times}{#1 \cdot #2}
```

then `\plus{a}{\times{b}{c}}` would yield  $a + b \cdot c$ , and `\times{a}{\plus{b}{c}}` would yield  $a \cdot (b + c)$ .

### 2.2 Archives and Imports

#### 2.2.1 Namespaces

Ideally,  $\text{\S}$ 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,  $\text{\T}$ EX 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  $\text{\S}$ 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.

## 2.2.2 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.

## 3 Documentation

### 3.1 Utils

<hr/> <code>\sTeX</code> <code>\stex</code> <hr/>	both print this sTeX logo.
<hr/> <code>\stex_debug:n</code> <hr/>	<code>\stex_debug:n {&lt;message&gt;}</code> Logs <code>&lt;message&gt;</code> , if the package option <code>debug</code> is used.

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

---

<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.
--------------------------------	---

---



---

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

---

### 3.1.1 $\text{\texttt{S}\text{\texttt{C}}\text{\texttt{A}}\text{\texttt{L}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ , $\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}\text{\texttt{M}}\text{\texttt{L}}$ and $\text{\texttt{H}}\text{\texttt{T}}\text{\texttt{M}}\text{\texttt{L}}$ Annotations

---

<code>\if@latexml</code> <code>\latexml_if_p:</code> <code>\latexml_if:T</code> <code>\latexml_if:F</code> <code>\latexml_if:TF</code>	$\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}_{2\text{e}}$ and $\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}_3$ conditionals for $\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}\text{\texttt{M}}\text{\texttt{L}}$ .
--	--

---

We have four macros for annotating generated HTML (via  $\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}\text{\texttt{M}}\text{\texttt{L}}$  or  $\text{\texttt{S}}\text{\texttt{C}}\text{\texttt{A}}\text{\texttt{L}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ ) with attributes:

---

<code>\stex_annotate:nnn</code>	<code>\stex_annotate:nnn {&lt;property&gt;} {&lt;resource&gt;} {&lt;content&gt;}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

---

Annotates the HTML generated by `<content>` 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}{&lt;property&gt;}{&lt;resource&gt;}</code> <code>&lt;content&gt;</code> <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {&lt;property&gt;} {&lt;resource&gt;} {&lt;content&gt;}</code> .
--------------------------------	--

### 3.1.2 Languages

---

<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`.

## 3.2 Files, Paths, URIs

---

`\stex_path_from_string:Nn`      `\stex_path_from_string:Nn <path-variable> {<string>}`  
`\stex_path_from_string:(NV|cn|cV)`

---

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

---

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

---



---

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

---



---

`\stex_path_if_absolute_p:N`       $\star$   
`\stex_path_if_absolute:NTF`       $\star$

---

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

---

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

---



---

`\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|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/./..		

### 3.3 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).

---

`\stex_set_current_repository:n`

---

Sets the current repository to the one with the provided ID. calls `\__stex_mathhub_do_manifest:n`, so works whether this repository's `MANIFEST.MF`-file has already been read or not.

---

`\stex_require_repository:n`

---

Calls `\__stex_mathhub_do_manifest:n` iff the corresponding archive property list does not already exist, and adds a corresponding definition to the `.sms`-file.

#### 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: <http://mathhub.info/tests/Foo/Bar>  
ns: <http://mathhub.info/tests/Foo/Bar>  
deps:

### 3.4 The Module System

---

#### `\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`,

---

<code>\stex_if_in_module_p: *</code>	Conditional for whether we are currently in a module
<code>\stex_if_in_module: <i>TF</i> *</code>	

---



---

<code>\stex_if_module_exists_p:n *</code>	
<code>\stex_if_module_exists:n<i>TF</i> *</code>	

---

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

---

#### `\stex_add_to_current_module:n`

---

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/sty/stex-master/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest
```

### 3.4.1 The module-environment

`module`            `\begin{module}[<options>]{<name>}`  
 Opens a new module with name `<name>`.  
 TODO document options.

---

`\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}[<options>]{<name>}`  
 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:n{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:n{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 3.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

### 3.4.2 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 6

```
\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
```

## 3.4.3 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 7

```
\begin{module}{Foo1}
\symdecl[name=foo, args=3]{bar}
\symdecl[ args=bai]{foobar}
Meaning:-\present\bar\
\end{module}
Meaning:-\present\bar\
\begin{module}{Foo2}
\importmodule{Foo1}
Meaning:-\present\bar\
\end{module}
```

**Module 3.2[Foo1]**

Meaning: »macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo1?foo}«

Meaning: »macro:->\protect \bar «

**Module 3.3[Foo2]**

Meaning: »macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo1?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

---

**\stex\_import\_module\_uri:nn**

---

**\stex\_import\_module\_uri:nn**  $\{\langle archive-ID \rangle\}$   $\{\langle module-path \rangle\}$ 

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

If  $\langle archive-ID \rangle$  is empty, it is automatically set to the ID of the current archive (if one exists).

1. If  $\langle archive-ID \rangle$  is empty:

(a) If  $\langle path \rangle$  is empty, then  $\langle name \rangle$  must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name  $\langle name \rangle.\langle lang \rangle.tex$  must exist in the same folder, containing a module  $\langle name \rangle$ .

That module should have the same namespace as the current one.

(b) If  $\langle path \rangle$  is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

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

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

(b) If  $\langle path \rangle$  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**

---

**\stex\_import\_require\_module:nnnn**  $\{\langle ns \rangle\}$   $\{\langle archive-ID \rangle\}$   $\{\langle path \rangle\}$   $\{\langle name \rangle\}$ 

Checks whether a module with URI  $\langle ns \rangle?\langle name \rangle$  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.

---

**\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).

### 3.5 Symbols and Terms

---

`\symdecl` `\symdecl[⟨args⟩]{⟨macroname⟩}`

---

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  $\S\TeX$ , 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  $\S\TeX$  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`, `\symdef` and `\abbrdef`.

Ultimately stores the symbol `⟨URI⟩` 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 8

```
\begin{module}{Foo3}
\symdecl[name=foo, args=3]{bar}
\symdecl[name=foobar, args=iab]{bari}
\ExplSyntaxOn
Meaning:-\present\bar\
\stex_get_symbol:n { bar }
Result:-\l_stex_get_symbol_uri_str
\ExplSyntaxOff
\end{module}
```

Module 3.4[Foo3]

Meaning:  $\text{\macro:-}\text{\stex\_invoke\_symbol:n}$  (file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo3?foo)<  
Result: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo3?foo

$\text{\stex\_get\_symbol:n}$

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

$\text{\stex\_invoke\_symbol:n}$

TODO

$\text{\notation}$

$\text{\notation}[\langle args \rangle]\{\langle symbol \rangle\}\{\langle notations^+ \rangle\}$

Introduces a new notation for  $\langle symbol \rangle$ , see  $\text{\stex\_notation\_do:nn}$

$\text{\stex\_notation\_do:nn}$

$\text{\stex\_notation\_do:nn}\{\langle URI \rangle\}\{\langle notations^+ \rangle\}$

Implements the core functionality of  $\text{\notation}$ , and is called by  $\text{\notation}$  and  $\text{\symdef}$ .

Ultimately stores the notation in the property list  
 $\text{\g\_stex\_notation\_}\langle URI \rangle \# \langle variant \rangle \# \langle lang \rangle \_prop$  with fields:

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

## Test 9

```
\begin{module}{Foo4}
\importmodule{Foo1}
\notation[foo, prec=500;20x20x20]{bar}{\langle {#1} ^ {#2} _ {#3} \rangle }
\notation[foo, prec=500;20x20x20]{foobar}{\langle #1 \mid [ #2 ] ^ {#3} \rangle }{ {#1}_{:#2} }
\end{module}
```

Module 3.5[ <code>Foo4</code> ]
---------------------------------

.

---

---

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

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

### Test 10

```
\begin{module}{Foo6}
\symdef[args=a, prec=50]{plus}{ #1 }{#1 + #2}
 $\plus{a,b,c}$ 
\end{module}
```

Module 3.6[ <code>Foo6</code> ] $a + b + c$
--

.

---

---

`\stex_term_oms:nnnn`    `⟨URI⟩⟨fragment⟩⟨precedence⟩⟨body⟩`

`\stex_term_oma:nnnn`    Annotates `⟨body⟩` as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol `⟨URI⟩`, generated by the specific notation `⟨fragment⟩` with (upwards) operator precedence `⟨precedence⟩`. Inserts parentheses according to the current downwards precedence and operator precedence.

---

---

`\stex_term_arg:nnn`    `\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨body⟩`

Annotates `⟨body⟩` as the `⟨int⟩`th argument of the current OMA or OMBIND, with (downwards) argument precedence `⟨prec⟩`.

---

---

`\stex_term_assoc_arg:nnnn`    `\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨body⟩`

Annotates `⟨body⟩` as the `⟨int⟩`th (associative) *sequence* argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence `⟨prec⟩` and associative notation `⟨notation⟩`.

---

---

`\infprec`    Maximal and minimal notation precedences.  
`\neginfprec`

---

---

`\STEXdobrackets`    `\STEXdobrackets {⟨body⟩}`

Puts `⟨body⟩` in parentheses; scaled if in display mode unscaled otherwise. Uses the current  $\TeX$  brackets (by default ( and )), which can be changed temporarily using `\STEXwithbrackets`.



---

`\STEXwithbrackets`

---

`\STEXwithbrackets`  $\langle left \rangle$   $\langle right \rangle$   $\{ \langle body \rangle \}$

Temporarily (i.e. within  $\langle body \rangle$ ) sets the brackets used by  $\TeX$  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 `\left` and `\right` in display-mode.

### Test 11

```
\begin{module}{Foo5}
\importmodule{Foo1}
\notation[foo, prec=500;20x20x20]{bar}{\langle #1 ^ {#2} _ {#3} \rangle }
$\bar{abc}$ and $\bar{foo} \ abc$
\end{module}
```

Module 3.7[Foo5]  
 $\langle a^b_c \rangle$  and  $\langle a^b_c \rangle$

### Test 12

```
\begin{module}{Foo7}
\importmodule{Foo1}
\notation[foo, prec=500;20x20x20]{foobar}{\langle #1 \mid [ #2 ]^{#3} \rangle }{ {#1}_{:#2} }
$\foobar a{b,c,d,e,f}g$ and $\foobar[foo] a{b,c}g$ and $\foobar abc$

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

Module 3.8[Foo7]  
 $\langle a \mid [b;c;d:e,f]^g \rangle$  and  $\langle a \mid [b;c]^g \rangle$  and  $\langle a \mid [b]^c \rangle$   
 $a + b \cdot c$  and  $a \cdot (\frac{a}{b} + \frac{a}{c})$   
 $a + b \cdot c$  and  $a \cdot \left( \frac{a}{b} + \frac{a}{c} \right)$   
 $a + b \cdot c$  and  $a \cdot \left[ \frac{a}{b} + \frac{a}{c} \right]$

## 4 Implementation

### 4.1 The $\TeX$ document class

```
1 \*cls)
2 \RequirePackage{expl3,l3keys2e}
3 \ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
4 \LoadClass[border=1px,varwidth]{standalone}
5 \setlength\textwidth{15cm}
```

```

6 \g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \RequirePackage{stex}
12 \</cls>

```

## 4.2 Preliminaries

```

13 \<*package>
14 \RequirePackage{expl3,l3keys2e}
15 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}

Package options:
16 \keys_define:nn { stex } {
17   debug      .bool_set:N = \c_stex_debug_bool ,
18   showmods   .bool_set:N = \c_stex_showmods_bool ,
19   lang       .clist_set:N = \c_stex_languages_clist ,
20   mathhub    .tl_set_x:N = \mathhub ,
21   sms        .bool_set:N = \c_stex_persist_mode_bool
22 }
23 \ProcessKeysOptions { stex }

```

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

```

24 \protected\def\sTeX{%
25   \@ifundefined{texorpdfstring}%
26   {\let\texorpdfstring\@firstoftwo}%
27   }%
28   \texorpdfstring{\raisebox{-.5ex}{S\kern-.5ex\TeX}{sTeX}\xspace%
29 }
30 \def\sTeX{\sTeX}

```

(End definition for \sTeX. This function is documented on page 4.)

Messages

```

31 \msg_new:nnn{stex}{debug}{}
32 \msg_new:nnn{stex}{warning/nomathhub}{
33   MATHHUB~system~variable~not~found~and~no~
34   \detokenize{\mathhub}~value~set!
35 }
36 \msg_new:nnn{stex}{error/norepository}{}
37 \msg_new:nnn{stex}{error/modulemissing}{}

```

**\stex\_debug:n** Debug mode

```

38 \cs_new_protected:Nn \stex_debug:n {
39   \bool_if:nT{\c_stex_debug_bool}{
40     \exp_args:Nnn\msg_set:nnn{stex}{debug}{\Debug:~#1\\}
41     \msg_term:nn{stex}{debug} % should be \msg_note:nn
42   }
43 }
44
45 \stex_debug:n{Debug~mode~on}

```

(End definition for \stex\_debug:n. This function is documented on page 4.)

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

```

46 \iow_new:N \c__stex_sms_iow
47 \AddToHook{begindocument}{
48   \bool_if:NTF \c_stex_persist_mode_bool {
49     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
50   } {
51     \iow_open:Nn \c__stex_sms_iow {\jobname.sms}
52   }
53 }
54 \AddToHook{enddocument}{
55   \bool_if:NF \c_stex_persist_mode_bool {
56     \iow_close:N \c__stex_sms_iow
57   }
58 }

```

*(End definition for `\c__stex_sms_iow`.)*

`\stex_addtosms:n`

```

59 \cs_new_protected:Nn \stex_addtosms:n {
60   \bool_if:NF \c_stex_persist_mode_bool {
61     \iow_now:Nn \c__stex_sms_iow { #1 }
62   }
63 }

```

*(End definition for `\stex_addtosms:n`. This function is documented on page 5.)*

#### 4.2.1 L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L and S<sup>C</sup>A<sub>L</sub>T<sub>E</sub>X

```

64 \RequirePackage{scalatex}

```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to S<sup>C</sup>A<sub>L</sub>T<sub>E</sub>X:

```

65 \scalatex_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
66 \ifcsname if@latexml\endcsname\else
67   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
68 \fi
69
70 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
71   \if@latexml
72     \prg_return_true:
73   \else:
74     \prg_return_false:
75   \fi:
76 }

```

*(End definition for `\if@latexml` and `\latexml_if:TF`. These functions are documented on page 5.)*

## 4.2.2 HTML Annotations

77 `<@=stex_annotate>`

`\l__stex_annotate_arg_tl`  
`\c__stex_annotate_emptyarg_tl`

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

```
78 \tl_new:N \l__stex_annotate_arg_tl
79 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
80   \scalatex_if:TF {
81     \scalatex_direct_HTML:n { \c_ampsand_str lrm; }
82   }{-}
83 }
```

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

`\__stex_annotate_checkempty:n`

```
84 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
85   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
86   \tl_if_empty:NT \l__stex_annotate_arg_tl {
87     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
88   }
89 }
```

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

`\stex_annotate:nnw`

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

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

```
90 \scalatex_if:TF{
91   \cs_new_protected:Nn \stex_annotate:nnn {
92     \__stex_annotate_checkempty:n { #3 }
93     \scalatex_annotate_HTML:nn {
94       property="stex:#1" ~
95       resource="#2"
96     } {
97       \tl_use:N \l__stex_annotate_arg_tl
98     }
99   }
100   \cs_new_protected:Nn \stex_annotate_invisible:n {
101     \__stex_annotate_checkempty:n { #1 }
102     \scalatex_annotate_HTML:nn {
103       stex:visible="false" ~
104       style:display="none"
105     } {
106       \tl_use:N \l__stex_annotate_arg_tl
107     }
108   }
109   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
110     \__stex_annotate_checkempty:n { #3 }
111     \scalatex_annotate_HTML:nn {
112       property="stex:#1" ~
113       resource="#2" ~
114       stex:visible="false" ~
115       style:display="none"
```

```

116     } {
117       \tl_use:N \l__stex_annotate_arg_tl
118     }
119   }
120   \NewDocumentEnvironment{stex_annotate_env} { m m } {
121     \par
122     \scalatex_annotate_HTML_begin:n {
123       property="stex:#1" ~
124       resource="#2"
125     }
126   }{
127     \scalatex_annotate_HTML_end:
128   }
129   }{
130     \latexml_if:TF {
131       \cs_new_protected:Nn \stex_annotate:nnn {
132         \__stex_annotate_checkempty:n { #3 }
133         \mode_if_math:TF {
134           \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
135             \tl_use:N \l__stex_annotate_arg_tl
136           }
137         }{
138           \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
139             \tl_use:N \l__stex_annotate_arg_tl
140           }
141         }
142       }
143       \cs_new_protected:Nn \stex_annotate_invisible:n {
144         \__stex_annotate_checkempty:n { #1 }
145         \mode_if_math:TF {
146           \cs:w latexml@invisible@math\cs_end:{
147             \tl_use:N \l__stex_annotate_arg_tl
148           }
149         } {
150           \cs:w latexml@invisible@text\cs_end:{
151             \tl_use:N \l__stex_annotate_arg_tl
152           }
153         }
154       }
155       \cs_new_protected:Nn \stex_annotate_invisible:nnn {
156         \__stex_annotate_checkempty:n { #3 }
157         \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
158           \tl_use:N \l__stex_annotate_arg_tl
159         }
160       }
161       \NewDocumentEnvironment{stex_annotate_env} { m m } {
162         \par\begin{latexml@annotateenv}{#1}{#2}
163       }{
164         \end{latexml@annotateenv}
165       }
166     }{
167       \cs_new_protected:Nn \stex_annotate:nnn {#3}
168       \cs_new_protected:Nn \stex_annotate_invisible:n {}
169       \cs_new_protected:Nn \stex_annotate_invisible:nnn {}

```

```

170 \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{\}
171 }
172 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 5.)

### 4.2.3 Languages

```

173 <@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```

174 \prop_const_from_keyval:Nn \c_stex_languages_prop {
175   en = english ,
176   de = ngerman ,
177   ar = arabic ,
178   bg = bulgarian ,
179   ru = russian ,
180   fi = finnish ,
181   ro = romanian ,
182   tr = turkish ,
183   fr = french
184 }
185
186 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
187   english   = en ,
188   ngerman   = de ,
189   arabic    = ar ,
190   bulgarian = bg ,
191   russian   = ru ,
192   finnish   = fi ,
193   romanian  = ro ,
194   turkish   = tr ,
195   french    = fr
196 }
197 % todo: chinese simplified (zhs)
198 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 5.)

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

```

199 \clist_if_empty:NF \c_stex_languages_clist {
200   \clist_clear:N \l_tmpa_clist
201   \clist_map_inline:Nn \c_stex_languages_clist {
202     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
203       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
204     } {
205       \msg_set:nnn{stex}{error/unknownlanguage}{
206         Unknown~language~\l_tmpa_str
207       }
208       \msg_error:nn{stex}{error/unknownlanguage}
209     }
210   }
211   \stex_debug:n {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
212   \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
213 }

```

## 4.3 Files, Paths and URIs

214 `<@@=stex_path>`

### 4.3.1 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`

```
215 \cs_new_protected:Nn \stex_path_from_string:Nn {
216   \str_set:Nx \l_tmpa_str { #2 }
217   \str_if_empty:NTF \l_tmpa_str {
218     \seq_clear:N #1
219   }{
220     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
221     \sys_if_platform_windows:T{
222       \seq_clear:N \l_tmpa_tl
223       \seq_map_inline:Nn #1 {
224         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
225         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
226       }
227       \seq_set_eq:NN #1 \l_tmpa_tl
228     }
229     \stex_path_canonicalize:N #1
230   }
231 }
232 \cs_generate_variant:Nn \stex_path_from_string:Nn
233 { NV, cn, cV }
```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 6.)

`\stex_path_to_string:NN`  
`\stex_path_to_string:N`

```
234 \cs_new_protected:Nn \stex_path_to_string:NN {
235   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
236 }
237
238 \cs_new:Nn \stex_path_to_string:N {
239   \seq_use:Nn #1 /
240 }
```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 6.)

`\c__stex_path_dot_str`  
`\c__stex_path_up_str`

```
. and .., respectively.
241 \str_const:Nn \c__stex_path_dot_str {.}
242 \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.

```
243 \cs_new_protected:Nn \stex_path_canonicalize:N {
244   \seq_if_empty:NF #1 {
245     \seq_clear:N \l_tmpa_seq
246     \seq_get_left:NN #1 \l_tmpa_tl
247     \str_if_empty:NT \l_tmpa_tl {
```

```

248     \seq_put_right:Nn \l_tmpa_seq {}
249 }
250 \seq_map_inline:Nn #1 {
251   \str_set:Nn \l_tmpa_tl { ##1 }
252   \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
253     \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
254       \seq_if_empty:NTF \l_tmpa_seq {
255         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
256           \c__stex_path_up_str
257         }
258       }{
259         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
260         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
261           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
262             \c__stex_path_up_str
263           }
264         }{
265           \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
266         }
267       }
268     }{
269       \str_if_empty:NF \l_tmpa_tl {
270         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
271       }
272     }
273   }
274 }
275 \seq_gset_eq:NN #1 \l_tmpa_seq
276 }
277 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 6.)

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

```

278 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
279   \seq_if_empty:NNTF #1 {
280     \prg_return_false:
281   }{
282     \seq_get_left:NN #1 \l_tmpa_tl
283     \str_if_empty:NNTF \l_tmpa_tl {
284       \prg_return_true:
285     }{
286       \prg_return_false:
287     }
288   }
289 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 6.)

### 4.3.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

290 \str_new:N\l_stex_kpsewhich_return_str
291 \cs_new_protected:Nn \stex_kpsewhich:n {

```



```

292 \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
293 \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
294 \tl_trim_spaces:N \l_stex_kpsewhich_return_str
295 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 5.)

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

296 \sys_if_platform_windows:TF{
297   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
298 }{
299   \stex_kpsewhich:n{-var-value~PWD}
300 }
301
302 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
303 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
304 \stex_debug:n {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 6.)

### 4.3.3 File Hooks and Tracking

```

305 <@@=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

```

306 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

```

307 \stex_path_from_string:Nn \c_stex_mainfile_seq {
308   \c_stex_pwd_str/\g_file_curr_name_str.tex
309 }

```

(End definition for `\c_stex_mainfile_seq`. This variable is documented on page 6.)

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

```

310 \seq_gclear_new:N\g_stex_currentfile_seq
311 \AddToHook{file/before}{
312   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
313   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
314     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
315   }{
316     \stex_path_from_string:Nn\g_stex_currentfile_seq{
317       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
318     }
319   }

```

```

320 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
321 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
322 }
323 \AddToHook{file/after}{
324   \seq_if_empty:NF\g__stex_files_stack{
325     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
326   }
327   \seq_if_empty:NTF\g__stex_files_stack{
328     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
329   }{
330     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
331     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
332   }
333 }

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 6.)

## 4.4 MathHub Repositories

```

334 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
335 \str_if_empty:NTF\mathhub{
336   \stex_kpsewhich:n{-var-value~MATHHUB}
337   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
338 }
339 \str_if_empty:NTF\c_stex_mathhub_str{
340   \msg_warning:nn{stex}{warning/nomathhub}
341 }{
342   \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
343   \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
344 }
345 }{
346   \stex_path_from_string:Nn\c_stex_mathhub_seq\mathhub
347   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
348   \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
349 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 7.)

```

\__stex_mathhub_do_manifest:n
350 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
351   \str_set:Nx \l_tmpa_str { #1 }
352   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
353     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
354     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
355     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
356     \__stex_mathhub_find_manifest:N \l_tmpa_seq
357     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
358       \msg_set:nnn{stex}{error/norepository}{
359         No~archive~#1~found~in~
360         \stex_path_to_string:N \c_stex_mathhub_str
361       }
362       \msg_error:nn{stex}{error/norepository}

```

```

363     } {
364     \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
365     }
366   }
367 }

```

(End definition for \\_\_stex\_mathhub\_do\_manifest:n.)

\l\_stex\_mathhub\_manifest\_file\_seq

```

368 \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:

```

369 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
370   \seq_set_eq:NN\l_tmpa_seq #1
371   \bool_set_true:N\l_tmpa_bool
372   \bool_while_do:Nn \l_tmpa_bool {
373     \seq_if_empty:NTF \l_tmpa_seq {
374       \bool_set_false:N\l_tmpa_bool
375     }{
376       \file_if_exist:nTF{
377         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
378       }{
379         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
380         \bool_set_false:N\l_tmpa_bool
381       }{
382         \file_if_exist:nTF{
383           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
384         }{
385           \seq_put_right:Nn\l_tmpa_seq{META-INF}
386           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
387           \bool_set_false:N\l_tmpa_bool
388         }{
389           \file_if_exist:nTF{
390             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
391           }{
392             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
393             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
394             \bool_set_false:N\l_tmpa_bool
395           }{
396             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
397           }
398         }
399       }
400     }
401   }
402   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
403 }

```

(End definition for \\_\_stex\_mathhub\_find\_manifest:N.)

\c\_stex\_mathhub\_manifest\_ior

File variable used for MANIFEST-files

```

404 \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:

```

405 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
406   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
407   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
408   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
409     \str_set:Nn \l_tmpa_str {##1}
410     \exp_args:NNo \seq_set_split:Nnn
411       \l_tmpb_seq \c_colon_str \l_tmpa_str
412     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
413       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
414         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
415       }
416       \exp_args:No \str_case:nnTF \l_tmpa_tl {
417         {id} {
418           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
419             { id } \l_tmpb_tl
420         }
421         {narration-base} {
422           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
423             { narr } \l_tmpb_tl
424         }
425         {source-base} {
426           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
427             { ns } \l_tmpb_tl
428         }
429         {ns} {
430           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
431             { ns } \l_tmpb_tl
432         }
433         {dependencies} {
434           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
435             { deps } \l_tmpb_tl
436         }
437       }{}{}
438     }{}
439   }
440   \ior_close:N \c__stex_mathhub_manifest_ior
441 }
```

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

`\stex_set_current_repository:n`

```

442 \cs_new_protected:Nn \stex_set_current_repository:n {
443   \stex_require_repository:n { #1 }
444   \prop_set_eq:Nc \l_stex_current_repository_prop {
445     c_stex_mathhub_#1_manifest_prop
446   }
447 }
```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 7.)

`\stex_require_repository:n`

```

448 \cs_new_protected:Nn \stex_require_repository:n {
449   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
450     \stex_debug:n{Opening~archive:~#1}
451     \__stex_mathhub_do_manifest:n { #1 }
452     \exp_args:Nx \stex_addtosms:n {
453       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
454         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
455         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
456         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
457         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
458       }
459     }
460   }
461 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 7.)

`\l_stex_current_repository_prop`

Current MathHub repository and a hook for `\begin{document}` to set it initially.

```

462 \prop_new:N \l_stex_current_repository_prop
463 \AddToHook{begindocument}{
464   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
465   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
466     \stex_debug:n{Not~currently~in~a~MathHub~repository}
467   } {
468     \__stex_mathhub_parse_manifest:n { main }
469     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
470     \l_tmpa_str
471     \prop_set_eq:cn { c_stex_mathhub_#1_manifest_prop }
472     \stex_set_current_repository:n { main }
473     \stex_debug:n{Current~repository:~
474       \prop_item:Nn \l_stex_current_repository_map {id}
475     }
476   }
477 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 7.)

## 4.5 Module System

```

478 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

479 \prop_new:N \l_stex_current_module_prop

```

(End definition for `\l_stex_current_module_prop`. This variable is documented on page 8.)

`stex_if_in_module_p:`

`stex_if_in_module:TF`

```

480 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
481   \prop_if_empty:NTF \l_stex_current_module_prop
482     \prg_return_false: \prg_return_true:
483 }

```

(End definition for `stex_if_in_module:TF`. This function is documented on page 8.)

stex\_if\_module\_exists\_p:n  
stex\_if\_module\_exists:nTF

```
484 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
485   \prop_if_exist:cTF { c_stex_module_#1_prop }
486   \prg_return_true: \prg_return_false:
487 }
```

(End definition for stex\_if\_module\_exists:nTF. This function is documented on page 8.)

\stex\_add\_to\_current\_module:n

```
488 \cs_new_protected:Nn \stex_add_to_current_module:n {
489   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
490   \tl_put_right:Nn \l_tmpa_tl { #1 }
491   \prop_put:Nno \l_stex_current_module_prop { content } \l_tmpa_tl
492 }
```

(End definition for \stex\_add\_to\_current\_module:n. This function is documented on page 8.)

\stex\_add\_constant\_to\_current\_module:n

```
493 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
494   \str_set:Nx \l_tmpa_str { #1 }
495   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
496   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
497   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
498 }
```

(End definition for \stex\_add\_constant\_to\_current\_module:n. This function is documented on page 8.)

\stex\_add\_import\_to\_current\_module:n

```
499 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
500   \str_set:Nx \l_tmpa_str { #1 }
501   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
502   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
503   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
504 }
```

(End definition for \stex\_add\_import\_to\_current\_module:n. This function is documented on page 9.)

\stex\_modules\_compute\_namespace:nN stores its return values in:

\l\_stex\_modules\_ns\_str

```
505 \str_new:N \l_stex_modules_ns_str

506 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
507   \str_set:Nx \l_tmpa_str { #1 }
508   \seq_set_eq:NN \l_tmpa_seq #2
509   % split off file extension
510   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
511   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
512   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
513   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
514
515   \bool_set_true:N \l_tmpa_bool
516   \bool_while_do:Nn \l_tmpa_bool {
517     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
518     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
```

```

519     {source} { \bool_set_false:N \l_tmpa_bool }
520   }{}{
521     \seq_if_empty:NT \l_tmpa_seq {
522       \bool_set_false:N \l_tmpa_bool
523     }
524   }
525 }
526
527 \seq_if_empty:NTF \l_tmpa_seq {
528   \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
529 }{
530   \str_set:Nx \l_stex_modules_ns_str {
531     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
532   }
533 }
534 }

```

(End definition for `\stex_modules_compute_namespace:nN` and `\l_stex_modules_ns_str`. These functions are documented on page 9.)

`\stex_modules_current_namespace:`

```

535 \cs_new_protected:Nn \stex_modules_current_namespace: {
536   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
537     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
538   }{
539     % split off file extension
540     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
541     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
542     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
543     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
544     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
545     \str_set:Nx \l_stex_modules_ns_str {
546       file:/\stex_path_to_string:N \l_tmpa_seq
547     }
548   }
549 }

```

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

#### 4.5.1 The module environment

module module arguments:

```

550 \keys_define:nn { stex / module } {
551   title .tl_set_x:N = \l_stex_module_title_str ,
552   ns .tl_set_x:N = \l_stex_module_ns_str ,
553   lang .tl_set_x:N = \l_stex_module_lang_str ,
554   sig .tl_set_x:N = \l_stex_module_sig_str ,
555   meta .tl_set_x:N = \l_stex_module_meta_str
556 }
557
558 % module parameters here? In the body?
559
560 \cs_new_protected:Nn \__stex_module_args:n {
561   \str_clear:N \l_stex_module_title_str

```

```

562 \str_clear:N \l_stex_module_ns_str
563 \str_clear:N \l_stex_module_lang_str
564 \str_clear:N \l_stex_module_sig_str
565 \str_clear:N \l_stex_module_meta_str
566 \keys_set:nn { stex / module } { #1 }
567 \exp_args:NNo \str_set:Nn \l_stex_module_title_str
568   \l_stex_module_title_str
569 \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
570   \l_stex_module_ns_str
571 \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
572   \l_stex_module_lang_str
573 \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
574   \l_stex_module_sig_str
575 \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
576   \l_stex_module_meta_str
577 }

```

\\_stex\_module\_begin\_module: implements \begin{module}

```

578 \cs_new_protected:Nn \_stex_module_begin_module: {
579   % Nested module?
580   \stex_if_in_module:TF {
581     % Nested module
582     \prop_get:NnN \l_stex_current_module_prop
583       { ns } \l_stex_module_ns_str
584     \str_set:Nx \l_stex_module_name_str {
585       \prop_item:Nn \l_stex_current_module_prop
586         { name } / \l_stex_module_name_str
587     }
588   }{
589     % not nested:
590     \str_if_empty:NT \l_stex_module_ns_str {
591       \stex_modules_current_namespace:
592       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
593       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
594         / {\l_stex_module_ns_str}
595       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
596       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
597         \str_set:Nx \l_stex_module_ns_str {
598           \stex_path_to_string:N \l_tmpa_seq
599         }
600       }
601     }
602   }
603
604   % language
605   \str_if_empty:NF \l_stex_module_lang_str {
606     \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
607       \l_tmpa_str {
608       \exp_args:Nx \selectlanguage { \l_tmpa_str }
609     } {
610       \msg_set:nnn{stex}{error/unknownlanguage}{
611         Unknown~language~\l_tmpa_str
612       }
613       \msg_error:nn{stex}{error/unknownlanguage}

```



```

614     }
615 }
616
617 % signature
618 \str_if_empty:NF \l_stex_module_sig_str {
619   \str_if_empty:NT \l_stex_module_lang_str {
620     \msg_set:nnn{stex}{error/siglanguage}{
621       Module~\l_stex_module_ns_str?\l_stex_module_name_str~
622       declares~signature~\l_stex_module_sig_str,~but~does~not~
623       declare~its~language
624     }
625     \msg_error:nn{stex}{error/siglanguage}
626   }
627 }
628
629 % metatheory
630 % \str_if_empty:NTF \l_stex_module_meta_str {
631 %
632 % } {
633 %
634 % }
635
636 \str_clear:N \l_tmpa_str
637 \seq_clear:N \l_tmpa_seq
638 \tl_clear:N \l_tmpa_tl
639 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
640   name      = \l_stex_module_name_str ,
641   ns        = \l_stex_module_ns_str ,
642   imports    = \exp_not:o { \l_tmpa_seq } ,
643   constants  = \exp_not:o { \l_tmpa_seq } ,
644   content    = \exp_not:o { \l_tmpa_tl } ,
645   file       = \exp_not:o { \g_stex_currentfile_seq } ,
646   lang       = \l_stex_module_lang_str ,
647   sig        = \l_stex_module_sig_str ,
648   meta       = \l_stex_module_meta_str
649 }
650
651 \stex_debug:n{
652   New~module:\\
653   Namespace:~\l_stex_module_ns_str\\
654   Name:~\l_stex_module_name_str\\
655   Language:~\l_stex_module_lang_str\\
656   Signature:~\l_stex_module_sig_str\\
657   Metatheory:~\l_stex_module_meta_str\\
658   File:~\stex_path_to_string:N \g_stex_currentfile_seq
659 }
660
661 \seq_gput_right:Nx \g_stex_modules_in_file_seq
662   { \l_stex_module_ns_str ? \l_stex_module_name_str }
663
664 \stex_if_smsmode:TF {
665   \stex_smsmode_set_codes:
666 } {
667   \begin{stex_annotate_env} {theory} {

```

```

668     \l_stex_module_ns_str ? \l_stex_module_name_str
669   }
670
671   \stex_annotate_invisible:nnn{header}{} {
672     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
673     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
674     \str_if_empty:NT \l_stex_module_meta_str {
675       % TODO metatheory
676     }
677   }
678 }
679 }
680 \iffalse \end{stex_annotate_env} \fi % make syntax highlighting work again

```

(End definition for `\_stex_module_begin_module:`)

`\_stex_module_end_module:` implements `\end{module}`

```

681 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
682 \cs_new_protected:Nn \_stex_module_end_module: {
683   \str_set:Nx \l_tmpa_str {
684     c_stex_module_
685     \prop_item:Nn \l_stex_current_module_prop { ns } ?
686     \prop_item:Nn \l_stex_current_module_prop { name }
687     _prop
688   }
689   \prop_new:c { \l_tmpa_str }
690   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
691   \stex_if_smsmode:TF {
692     \exp_args:Nx \stex_addtosms:n {
693       \prop_gset_from_keyval:cn {
694         c_stex_module_
695         \prop_item:Nn \l_stex_current_module_prop { ns } ?
696         \prop_item:Nn \l_stex_current_module_prop { name }
697         _prop
698       } {
699         name      = \prop_item:cn { \l_tmpa_str } { name } ,
700         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
701         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
702         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
703         content   = \prop_item:cn { \l_tmpa_str } { content } ,
704         file      = \prop_item:cn { \l_tmpa_str } { file } ,
705         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
706         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
707         meta      = \prop_item:cn { \l_tmpa_str } { meta }
708       }
709     }
710   }{
711     \end{stex_annotate_env}
712   }
713 }

```

(End definition for `\_stex_module_end_module:`)

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

```

714 \NewDocumentEnvironment { @module } { 0{} m } {
715   \str_set:Nx \l_stex_module_name_str { #2 }
716   \par
717   \__stex_module_args:n { #1 }
718   \__stex_module_begin_module:
719 } {
720   \__stex_module_end_module:
721 }

```

`\stex_modules_heading:` Code for document headers

```

722 \cs_if_exist:NTF \thesection {
723   \newcounter{module}[section]
724 }{
725   \newcounter{module}
726 }
727
728 \bool_if:NT \c_stex_showmods_bool {
729   \latexml_if:F { \RequirePackage{mdframed} }
730 }
731
732 \cs_new_protected:Nn \stex_modules_heading: {
733   \stepcounter{module}
734   \par
735   \bool_if:NT \c_stex_showmods_bool {
736     \noindent{\textbf{Module} ~
737       \cs_if_exist:NT \thesection {\thesection.}
738       \themodule ~ [\l_stex_module_name_str]
739     }
740     % TODO references
741     % \sref@label@id{Module \thesection.\themodule [\module@name]}\%
742     \str_if_empty:NTF \l_stex_module_title_str {
743       }{
744         \quad(\l_stex_module_title_str)\hfill
745       }\par
746     }
747 }

```

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

Finally:

```

748 \NewDocumentEnvironment { module } { 0{} m } {
749   \bool_if:NT \c_stex_showmods_bool {
750     \begin{mdframed}
751   }
752   \begin{@module}[#1]{#2}
753   \stex_modules_heading:
754 }{
755   \end{@module}
756   \bool_if:NT \c_stex_showmods_bool {
757     \end{mdframed}
758   }
759 }

```

## 4.5.2 SMS Mode

```

760 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

761 \tl_new:N \g_stex_smsmode_allowedmacros_tl
762 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
763 \seq_new:N \g_stex_smsmode_allowedenvs_seq
764
765 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
766   \makeatletter
767   \makeatother
768   \ExplSyntaxOn
769   \ExplSyntaxOff
770 }
771
772 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
773   \symdef
774   % \abbrdef
775   % \module@export
776   \importmodule
777   % \mmt@symdecl
778   % \instantiates
779   % \setnotation
780   % \importmhmodule
781   % \gimport
782   % \symvariant
783   % \structural@feature
784   % \symi
785   % \symii
786   % \symiii
787   % \symiv
788   \notation
789   \symdecl
790   % \defi
791   % \defii
792   % \defiii
793   % \defiv
794   % \adefi
795   % \adefii
796   % \adefiii
797   % \adefiv
798   % \defis
799   % \defiis
800   % \defiiis
801   % \defivs
802   % \Defi
803   % \Defii
804   % \Defiii
805   % \Defiv
806   % \Defis
807   % \Defiis
808   % \Defiiis
809   % \Defivs
810 }
```

```

811
812 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
813   \tl_to_str:n {
814     module,
815     @module
816   % modsig,
817   % mhmodsig,
818   % mhmodnl,
819   % modnl,
820   % @structural@feature
821   }
822 }

```

(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 10.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF
823 \bool_new:N \g__stex_smsmode_bool
824 \bool_set_false:N \g__stex_smsmode_bool
825 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
826   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
827 }

```

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

```

\__stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
\__stex_smsmode_if_catcodes:TF
828 \bool_new:N \g__stex_smsmode_catcode_bool
829 \bool_set_false:N \g__stex_smsmode_catcode_bool
830 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
831   \bool_if:NTF \g__stex_smsmode_catcode_bool
832   \prg_return_true: \prg_return_false:
833 }

```

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

```

\stex_smsmode_set_codes:
834 \cs_new_protected:Nn \stex_smsmode_set_codes: {
835   \stex_if_smsmode:T {
836     \__stex_smsmode_if_catcodes:F {
837       \bool_gset_true:N \g__stex_smsmode_catcode_bool
838       \exp_after:wN \char_gset_active_eq:NN
839       \c_backslash_str \__stex_smsmode_cs:
840       \tex_global:D \char_set_catcode_active:N \
841       \tex_global:D \char_set_catcode_other:N $
842       \tex_global:D \char_set_catcode_other:N ^
843       \tex_global:D \char_set_catcode_other:N _
844       \tex_global:D \char_set_catcode_other:N &
845       \tex_global:D \char_set_catcode_other:N ##
846     }
847   }
848 } \iffalse $ \fi % to make syntax highlighting work again

```

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

`\__stex_smsmode_unset_codes:` Sets category code scheme back from the one used in SMS mode.

```

849 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
850   \__stex_smsmode_if_catcodes:T {
851     \bool_gset_false:N \g__stex_smsmode_catcode_bool
852     \exp_after:wN \tex_global:D \exp_after:wN
853     \char_set_catcode_escape:N \c_backslash_str
854     \tex_global:D \char_set_catcode_math_toggle:N $
855     \tex_global:D \char_set_catcode_math_superscript:N ^
856     \tex_global:D \char_set_catcode_math_subscript:N _
857     \tex_global:D \char_set_catcode_alignment:N &
858     \tex_global:D \char_set_catcode_parameter:N ##
859   }
860 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\__stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

861 \cs_new_protected:Nn \stex_in_smsmode:nn {
862   \vbox_set:Nn \l_tmpa_box {
863     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
864     \bool_gset_true:N \g__stex_smsmode_bool
865     \stex_smsmode_set_codes:
866     #2
867     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
868     \stex_if_smsmode:F {
869       \__stex_smsmode_unset_codes:
870     }
871   }
872   \box_clear:N \l_tmpa_box
873 }

```

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

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

```

874 \str_const:Nn \c__stex_smsmode_begin_str { begin }
875 \str_const:Nn \c__stex_smsmode_end_str { end }
876
877 \cs_new_protected:Nn \__stex_smsmode_cs: {
878   \str_clear:N \l_tmpa_str
879   \peek_analysis_map_inline:n {
880     % #1: token (one expansion)
881     % #2: charcode
882     % #3 catcode
883     \token_if_eq_charcode:NNTF ##3 B {
884       % token is a letter
885       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
886     } {
887       \str_if_empty:NTF \l_tmpa_str {
888         % we don't allow (or need) single non-letter CSs
889         % for now
890         \peek_analysis_map_break:
891       }{
892         \str_if_eq:nnTF \l_tmpa_str \c_stex_begin_str {

```

```

893     \peek_analysis_map_break:n {
894       \exp_after:wN \__stex_smsmode_checkbegin:n ##1
895     }
896   } {
897     \str_if_eq:nnTF \l_tmpa_str \c_stex_end_str {
898       \peek_analysis_map_break:n {
899         \exp_after:wN \__stex_smsmode_checkend:n ##1
900       }
901     } {
902       \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
903       \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
904         \g_stex_smsmode_allowedmacros_tl
905         { \use:c{\l_tmpa_str} } {
906         \peek_analysis_map_break:n {
907           \exp_after:wN \l_tmpa_tl ##1
908         }
909       } {
910         \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
911         \g_stex_smsmode_allowedmacros_escape_tl
912         { \use:c{\l_tmpa_str} } {
913         \exp_args:NNNo \exp_args:No
914         \token_if_eq_charcode_p:NNTF \c_backslash_str ##1 {
915         \peek_analysis_map_break:n {
916           \__stex_smsmode_unset_codes:
917           \__stex_smsmode_rescan_cs:
918         }
919       } {
920         \peek_analysis_map_break:n {
921           \__stex_smsmode_unset_codes:
922           \exp_after:wN \l_tmpa_tl ##1
923         }
924       }
925     } {
926       \peek_analysis_map_break:n { ##1 }
927     }
928   }
929 }
930 }
931 }
932 }
933 }
934 }

```

(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:

```

935 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
936   \str_clear:N \l_tmpb_str
937   \peek_analysis_map_inline:n {
938     \token_if_eq_charcode:NNTF ##3 B {
939       % token is a letter
940       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
941     } {

```

```

942     \peek_analysis_map_break:n {
943       \exp_after:wN \use:c \exp_after:wN {
944         \exp_after:wN \l_tmpa_str\exp_after:wN
945       } \use:c { \l_tmpb_str \exp_after:wN } ##1
946     }
947   }
948 }
949 }

```

(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.

```

950 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
951   \str_set:Nn \l_tmpa_str { #1 }
952   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
953     \__stex_smsmode_unset_codes:
954     \begin{#1}
955   }
956 }

```

(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.

```

957 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
958   \str_set:Nn \l_tmpa_str { #1 }
959   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
960     \end{#1}
961   }
962 }

```

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

### 4.5.3 Inheritance

```

963 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

964 \cs_new_protected:Nn \stex_import_module_uri:nn {
965   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
966   \str_set:Nx \l__stex_importmodule_path_str { #2 }
967   \str_if_empty:NT \l__stex_importmodule_archive_str {
968     \prop_if_empty:NF \l_stex_current_repository_prop {
969       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
970     }
971   }
972
973   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
974   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
975   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpa_seq ? }
976
977   \str_if_empty:NTF \l_tmpa_str {
978     \stex_modules_current_namespace:
979     \str_if_empty:NF \l__stex_importmodule_path_str {
980       \str_set:Nx \l_stex_module_ns_str {
981         \l_stex_module_ns_str / \l__stex_importmodule_path_str

```



```

982     }
983   }
984 }{
985   \stex_require_repository:n \l__stex_importmodule_archive_str
986   \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str_manifest_prop } { ns }
987   \l_stex_module_ns_str
988   \str_if_empty:NF \l__stex_importmodule_path_str {
989     \str_set:Nx \l__stex_importmodule_module_ns_str {
990       \l_stex_module_ns_str / \l__stex_importmodule_path_str ? \l__stex_importmodule_name_
991     }
992   }
993 }
994 }

```

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

```

\l__stex_importmodule_name_str Store the return values of \stex_import_module_uri:nn.
\l__stex_importmodule_archive_str 995 \str_new:N \l__stex_importmodule_name_str
\l__stex_importmodule_path_str 996 \str_new:N \l__stex_importmodule_archive_str
997 \str_new:N \l__stex_importmodule_path_str

```

(End definition for `\l__stex_importmodule_name_str`, `\l__stex_importmodule_archive_str`, and `\l__stex_importmodule_path_str`.)

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
998 \cs_new_protected:Nn \stex_import_require_module:nnnn {
999   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1000     % archive
1001     \str_set:Nx \l_tmpa_str { #2 }
1002     \str_if_empty:NTF \l_tmpa_str {
1003       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1004     } {
1005       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1006       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1007       \seq_put_right:Nn \l_tmpa_seq { source }
1008     }
1009
1010     \stex_debug:n{Arguments: #1, #2, #3, #4}
1011
1012     % path
1013     \str_set:Nx \l_tmpb_str { #3 }
1014     \str_if_empty:NT \l_tmpb_str {
1015       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1016
1017       \cs_if_exist:NTF \languagename {
1018         \prop_get:NnN \c_stex_language_abbrevs_prop
1019           { \languagename } \l_tmpb_str
1020       }
1021
1022       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1023       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1024         \str_set:Nx \l_tmpa_str { \l_tmpa_str.\l_tmpb_str.tex }
1025       }{
1026         \stex_debug:n{Checking~\l_tmpa_str.tex}

```

```

1027 \IfFileExists{ \l_tmpa_str.tex }{
1028   \str_set:Nx \l_tmpa_str { \l_tmpa_str.tex }
1029 }{
1030   % try english as default
1031   \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1032   \IfFileExists{ \l_tmpa_str.en.tex }{
1033     \str_set:Nx \l_tmpa_str { \l_tmpa_str.en.tex }
1034   }{
1035     \msg_new:nnn{stex}{error/modulemissing}{
1036       No~file~for~module~#1?#4~found
1037     }
1038     \msg_error:nn{stex}{error/modulemissing}
1039   }
1040 }
1041 }
1042
1043 } {
1044   \stex_path_from_string:NV \l_tmpb_seq \l_tmpb_str
1045   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1046
1047   \cs_if_exist:NTF \language_name {
1048     \prop_get:NnN \c_stex_language_abbrevs_prop
1049       { \language_name } \l_tmpb_str
1050   }
1051
1052   \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq }
1053
1054   \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1055   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1056     \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1057   }{
1058     \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1059     \IfFileExists{ \l_tmpa_str/#4.tex }{
1060       \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.tex }
1061     }{
1062       % try english as default
1063       \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1064       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1065         \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.en.tex }
1066       }{
1067         \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1068         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1069           \str_set:Nx \l_tmpa_str { \l_tmpa_str.\l_tmpb_str.tex }
1070         }{
1071           \stex_debug:n{Checking~\l_tmpa_str.tex}
1072           \IfFileExists{ \l_tmpa_str.tex }{
1073             \str_set:Nx \l_tmpa_str { \l_tmpa_str.tex }
1074           }{
1075             % try english as default
1076             \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1077             \IfFileExists{ \l_tmpa_str.en.tex }{
1078               \str_set:Nx \l_tmpa_str { \l_tmpa_str.en.tex }
1079             }{
1080               \msg_new:nnn{stex}{error/modulemissing}{

```

```

1081         No~file~for~module~#1?#4~found
1082     }
1083     \msg_error:nn{stex}{error/modulemissing}
1084 }
1085 }
1086 }
1087 }
1088 }
1089 }
1090 }
1091
1092 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1093 \seq_clear:N \g_stex_modules_in_file_seq
1094 \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1095     \str_set:Nx \l_tmpb_str { #2 }
1096     \str_if_empty:NF \l_tmpb_str {
1097         \stex_set_current_repository:n { #2 }
1098     }
1099     \input { \l_tmpa_str }
1100 }
1101 \prop_gput:Noo \g_stex_module_files_prop
1102 \l_tmpa_str \g_stex_modules_in_file_seq
1103 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1104
1105 \stex_if_module_exists:nF { #1 ? #4 } {
1106     \msg_new:nnn{stex}{error/modulemissing}{
1107         Module~#1?#4~not~found~in~file~\l_tmpa_str
1108     }
1109     \msg_error:nn{stex}{error/modulemissing}
1110 }
1111 }
1112 % activate
1113 \stex_debug:n{Activating~module~#1?#4}
1114 \prop_item:cn { c_stex_module_#1?#4_prop } { content }
1115 }

```

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

## `\importmodule`

```

1116 \NewDocumentCommand \importmodule { 0{} m } {
1117     \stex_import_module_uri:nn { #1 } { #2 }
1118     \stex_debug:n{Importing~module:~
1119         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1120     }
1121     \stex_if_smsmode:F {
1122         \stex_import_require_module:nnnn
1123         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1124         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1125         \stex_annotate_invisible:nnn
1126         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1127     }
1128     \exp_args:Nx \stex_add_to_current_module:n {
1129         \stex_import_require_module:nnnn
1130         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }

```

```

1131     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1132   }
1133   \exp_args:Nx \stex_add_import_to_current_module:n {
1134     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1135   }
1136   \stex_smsmode_set_codes:
1137 }

```

(End definition for `\importmodule`. This function is documented on page 11.)

#### `\usemodule`

```

1138 \NewDocumentCommand \usemodule { 0{} m } {
1139   \stex_if_smsmode:F {
1140     \stex_import_module_uri:nn { #1 } { #2 }
1141     \stex_import_require_module:nnnn
1142     { \l__stex_importmodule_module_ns_str } { \l__stex_importmodule_archive_str }
1143     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1144     \stex_annotate_invisible:nnn
1145     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1146   }
1147   \stex_smsmode_set_codes:
1148 }

```

(End definition for `\usemodule`. This function is documented on page 12.)

#### `\g_stex_modules_in_file_seq`

#### `\g_stex_module_files_prop`

```

1149 \seq_new:N \g_stex_modules_in_file_seq
1150 \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 13.)

## 4.6 Symbol Declarations

```

1151 <@=stex_symdecl>

symdecl arguments:
1152 \keys_define:nn { stex / symdecl } {
1153   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1154   local .bool_set:N = \l_stex_symdecl_local_bool ,
1155   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1156   type .tl_set:N = \l_stex_symdecl_type_tl
1157 }
1158
1159 \cs_new_protected:Nn \__stex_symdecl_args:n {
1160   \str_clear:N \l_stex_symdecl_name_str
1161   \str_clear:N \l_stex_symdecl_args_str
1162   \bool_set_false:N \l_stex_symdecl_local_bool
1163   \tl_clear:N \l_stex_symdecl_type_tl
1164
1165   \keys_set:nn { stex / symdecl } { #1 }
1166
1167   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1168     \l_stex_symdecl_name_str
1169   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str

```

```

1170     \l_stex_symdecl_args_str
1171 }

```

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

```

1172 \NewDocumentCommand \symdecl { 0{} m } {
1173   \__stex_symdecl_args:n { #1 }
1174   \tl_clear:N \l_stex_symdecl_definiens_tl
1175   \stex_symdecl_do:n { #2 }
1176 }

```

(End definition for `\symdecl`. This function is documented on page 14.)

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

```

1177 \cs_new_protected:Nn \stex_symdecl_do:n {
1178   \stex_if_in_module:F {
1179     % TODO throw error? some default namespace?
1180   }
1181
1182   \str_if_empty:NT \l_stex_symdecl_name_str {
1183     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1184   }
1185
1186   \prop_if_exist:cT { g_stex_symdecl_
1187     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1188     \prop_item:Nn \l_stex_current_module_prop {name} ?
1189     \l_stex_symdecl_name_str
1190     _prop
1191   }{
1192     % TODO throw error (beware of circular dependencies)
1193   }
1194
1195   \prop_clear:N \l_tmpa_prop
1196   \prop_put:Nnx \l_tmpa_prop { module } {
1197     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1198     \prop_item:Nn \l_stex_current_module_prop {name}
1199   }
1200   \seq_clear:N \l_tmpa_seq
1201   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1202   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1203   \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1204   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1205
1206   \exp_args:No \stex_add_constant_to_current_module:n {
1207     \l_stex_symdecl_name_str
1208   }
1209
1210   % arity/args
1211   \int_zero:N \l_tmpb_int
1212
1213   \bool_set_true:N \l_tmpa_bool
1214   \str_map_inline:Nn \l_stex_symdecl_args_str {
1215     \token_case_meaning:NnF ##1 {
1216       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}

```

```

1217     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1218     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1219     {\tl_to_str:n a} {
1220         \bool_set_false:N \l_tmpa_bool
1221         \int_incr:N \l_tmpb_int
1222     }
1223 }{
1224     \msg_set:nnn{stex}{error/wrongargs}{
1225         args~value~in~symbol~declaration~for~
1226         \prop_item:Nn \l_stex_current_module_prop {ns} ?
1227         \prop_item:Nn \l_stex_current_module_prop {name} ?
1228         \l_stex_symdecl_name_str ~
1229         needs~to~be~
1230         i,~a~or~b,~but~##1~given
1231     }
1232     \msg_error:nn{stex}{error/wrongargs}
1233 }
1234 }
1235 \bool_if:NTF \l_tmpa_bool {
1236     % possibly numeric
1237     \str_if_empty:NTF \l_stex_symdecl_args_str {
1238         \prop_put:Nnn \l_tmpa_prop { args } {}
1239         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1240     }{
1241         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1242         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1243         \str_clear:N \l_tmpa_str
1244         \int_step_inline:nn \l_tmpa_int {
1245             \str_put_right:Nn \l_tmpa_str i
1246         }
1247         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1248     }
1249 } {
1250     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1251     \prop_put:Nnx \l_tmpa_prop { arity }
1252     { \str_count:N \l_stex_symdecl_args_str }
1253 }
1254 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1255
1256
1257 % semantic macro
1258
1259 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1260     \prop_item:Nn \l_tmpa_prop { module } ?
1261     \prop_item:Nn \l_tmpa_prop { name }
1262 } }
1263
1264 \bool_if:NF \l_stex_symdecl_local_bool {
1265     \exp_args:Nx \stex_add_to_current_module:n {
1266         \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1267             \prop_item:Nn \l_tmpa_prop { module } ?
1268             \prop_item:Nn \l_tmpa_prop { name }
1269         } }
1270     }

```

```

1271 }
1272
1273
1274 \stex_debug:n{New~symbol:~
1275   \prop_item:Nn \l_tmpa_prop { module } ?
1276   \prop_item:Nn \l_tmpa_prop { name }^^J
1277   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1278   Args:~\prop_item:Nn \l_tmpa_prop { args }
1279 }
1280
1281 \prop_gset_eq:cN {
1282   g_stex_symdecl_
1283   \prop_item:Nn \l_tmpa_prop { module } ?
1284   \prop_item:Nn \l_tmpa_prop { name }
1285   _prop
1286 } \l_tmpa_prop
1287
1288 \stex_if_smsmode:TF {
1289   \bool_if:NF \l_stex_symdecl_local_bool {
1290     \exp_args:Nx \stex_addtosms:n {
1291       \prop_gset_from_keyval:cn {
1292         g_stex_symdecl_
1293         \prop_item:Nn \l_tmpa_prop { module } ?
1294         \prop_item:Nn \l_tmpa_prop { name }
1295         _prop
1296       } {
1297         name      = \prop_item:Nn \l_tmpa_prop { name }
1298         module    = \prop_item:Nn \l_tmpa_prop { module }
1299         notations = \prop_item:Nn \l_tmpa_prop { notations }
1300         local     = \prop_item:Nn \l_tmpa_prop { local }
1301         type      = \prop_item:Nn \l_tmpa_prop { type }
1302         args      = \prop_item:Nn \l_tmpa_prop { args }
1303         arity     = \prop_item:Nn \l_tmpa_prop { arity }
1304         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1305       }
1306     }
1307   }
1308   \stex_smsmode_set_codes:
1309 }{
1310   \stex_annotate_invisible:nnn {symdecl} {
1311     \prop_item:Nn \l_tmpa_prop { module } ?
1312     \prop_item:Nn \l_tmpa_prop { name }
1313   } {
1314     \stex_annotate_invisible:nnn{type}{-}{\l_stex_symdecl_type_tl$}
1315     \stex_annotate_invisible:nnn{args}{-}{
1316       \prop_item:Nn \l_tmpa_prop { args }
1317     }
1318     \stex_annotate_invisible:nnn{macroname}{-}{#1}
1319     \str_if_empty:NF \l_stex_symdecl_definiens_tl {
1320       \stex_annotate_invisible:nnn{definiens}{-}{
1321         {\l_stex_symdecl_definiens_tl$}
1322       }
1323     }
1324   }

```

1325 }

(End definition for `\stex_symdecl_do:n`. This function is documented on page 14.)

`\stex_get_symbol:n`

```

1326 \str_new:N \l_stex_get_symbol_uri_str
1327
1328 \cs_new_protected:Nn \stex_get_symbol:n {
1329   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1330     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1331   }{
1332     % argument is a string
1333     % is it a command name?
1334     \cs_if_exist:cTF { #1 }{
1335       \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1336     }{
1337       % TODO
1338       % argument is not a command name
1339     }
1340   }
1341 }
1342
1343 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1344   \tl_set:Nx \l_tmpa_tl { #1 }
1345   \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
1346     \stex_invoke_symbol:n {
1347       \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1348         { \tl_tail:N \l_tmpa_tl }
1349       \tl_if_single:NTF \l_tmpa_tl {
1350         \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1351           \exp_after:wN \str_set:Nn \exp_after:wN
1352             \l_stex_get_symbol_uri_str \l_tmpa_tl
1353         }{
1354           % TODO
1355           % tail is not a single group
1356         }
1357       }{
1358         % TODO
1359         % tail is not a single group
1360       }
1361     }{
1362       % TODO
1363       % head is not \stex_invoke_symbol:n
1364     }
1365 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 15.)

## 4.7 Notations

```

1366 <@@=stex_notation>

notation arguments:
1367 \keys_define:nn { stex / notation } {
1368   lang      .tl_set_x:N = \l__stex_notation_lang_str ,

```



```

1369   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1370   prec    .tl_set_x:N = \l__stex_notation_prec_str ,
1371   unknown .code:n      = \str_set:Nx
1372         \l__stex_notation_variant_str \l_keys_key_str
1373 }
1374
1375 \cs_new_protected:Nn \__stex_notation_args:n {
1376   \str_clear:N \l__stex_notation_lang_str
1377   \str_clear:N \l__stex_notation_variant_str
1378   \str_clear:N \l__stex_notation_prec_str
1379
1380   \keys_set:nn { stex / notation } { #1 }
1381
1382   \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
1383     \l__stex_notation_lang_str
1384   \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
1385     \l__stex_notation_variant_str
1386   \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
1387     \l__stex_notation_prec_str
1388 }

```

**\notation**

```

1389 \NewDocumentCommand \notation { 0{ } m } {
1390   \__stex_notation_args:n { #1 }
1391   \tl_clear:N \l_stex_symdecl_definiens_tl
1392   \stex_get_symbol:n { #2 }
1393   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1394 }

```

(End definition for \notation. This function is documented on page 15.)

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

```

1395 \cs_new_protected:Nn \stex_notation_do:nn {
1396   \prop_set_eq:Nc \l_tmpa_prop {
1397     g_stex_symdecl_ #1 _prop
1398   }
1399
1400   \prop_clear:N \l_tmpb_prop
1401   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1402   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1403   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1404
1405   % precedences
1406   \seq_clear:N \l_tmpb_seq
1407   \exp_args:NNno
1408   \seq_set_split:Nnn \l_tmpa_seq ; { \l__stex_notation_prec_str }
1409   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1410     \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1411     \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1412       \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1413         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1414       \seq_map_inline:Nn \l_tmpa_seq {
1415         \seq_put_right:Nn \l_tmpb_seq { ##1 }
1416       }
1417     }
1418   }

```

```

1418 \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1419 }{
1420 \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1421 \int_compare:nNnTF \l_tmpa_str = 0 {
1422 \exp_args:NNnx
1423 \prop_put:Nnn \l_tmpb_prop { opprec }
1424 { \int_use:N \infprec }
1425 }{
1426 \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1427 }
1428 }
1429
1430 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1431 \int_step_inline:nn { \l_tmpa_str } {
1432 \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
1433 \exp_args:NNx
1434 \seq_put_right:Nn \l_tmpb_seq {
1435 \prop_item:Nn \l_tmpb_prop { opprec }
1436 }
1437 }
1438 }
1439
1440 \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
1441
1442 \int_compare:nNnTF \l_tmpa_str = 0 {
1443 \cs_set:Npx \l__stex_notation_macrocode_cs { {
1444 \stex_term_oms:nnnn { #1 }
1445 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1446 { \prop_item:Nn \l_tmpb_prop { opprec } }
1447 { #2 }
1448 }
1449 \__stex_notation_final:
1450 }{
1451 \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1452 \str_if_in:NnTF \l_tmpb_str b {
1453 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1454 \cs_set:Npx \l_tmpa_str {
1455 \stex_term_omb:nnnn { #1 }
1456 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1457 { \prop_item:Nn \l_tmpb_prop { opprec } }
1458 { #2 }
1459 }
1460 }{
1461 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1462 \cs_set:Npx \l_tmpa_str {
1463 \stex_term_oma:nnnn { #1 }
1464 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1465 { \prop_item:Nn \l_tmpb_prop { opprec } }
1466 { #2 }
1467 }
1468 }
1469
1470 \int_zero:N \l_tmpa_int
1471 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str

```

```

1472 \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1473 \tl_clear:N \l_tmpa_tl
1474 \__stex_notation_arguments:
1475 }
1476 }

```

(End definition for \stex\_notation\_do:nn. This function is documented on page 15.)

\\_\_stex\_notation\_arguments: Takes care of annotating the arguments in a notation macro

```

1477 \cs_new_protected:Nn \__stex_notation_arguments: {
1478 \int_incr:N \l_tmpa_int
1479 \str_if_empty:NTF \l_tmpa_str {
1480 \__stex_notation_final:
1481 }{
1482 \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1483 \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1484 \str_if_eq:VnTF \l_tmpb_str a {
1485 \__stex_notation_argument_assoc:n
1486 }{
1487 \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1488 \tl_put_right:Nx \l_tmpa_tl {
1489 { \stex_term_arg:nnn
1490 { \int_use:N \l_tmpa_int }
1491 { \l_tmpb_str }
1492 { ####\int_use:N \l_tmpa_int }
1493 }
1494 }
1495 \__stex_notation_arguments:
1496 }
1497 }
1498 }

```

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

\\_stex\_notation\_argument\_assoc:n

```

1499 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
1500 \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1501 \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1502 \tl_put_right:Nx \l_tmpa_tl {
1503 { \stex_term_assoc_arg:nnnn
1504 { \int_use:N \l_tmpa_int }
1505 { \l_tmpb_str }
1506 { \l_tmpa_cs {#####1} {#####2} }
1507 { ####\int_use:N \l_tmpa_int }
1508 }
1509 }
1510 \__stex_notation_arguments:
1511 }

```

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

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

```

1512 \cs_new_protected:Nn \__stex_notation_final: {
1513 \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str

```

```

1514 \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1515 \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1516 \cs_generate_from_arg_count:cNnn {
1517   stex_notation_ \l_tmpa_str \c_hash_str
1518   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1519   _cs
1520 }
1521 \cs_set:Npx \l_tmpb_str {
1522   \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl
1523 }
1524
1525 \stex_debug:n{
1526   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1527   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1528   Operator~precedence:~
1529   \prop_item:Nn \l_tmpb_prop { opprec }^^J
1530   Argument~precedences:~
1531   \seq_use:Nn \l_tmpa_seq { ,~ }^^J
1532   Notation: \cs_meaning:c {
1533     stex_notation_ \l_tmpa_str \c_hash_str
1534     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1535     _cs
1536   }
1537 }
1538
1539 \prop_gset_eq:cN {
1540   g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1541   \c_hash_str \l__stex_notation_lang_str _prop
1542 } \l_tmpb_prop
1543
1544 \exp_args:Nx
1545 \stex_add_to_current_module:n {
1546   \prop_get:cnN {
1547     g_stex_symdecl_
1548     \prop_item:Nn \l_tmpb_prop { symbol }
1549     _prop
1550   } { notations } \exp_not:N \l_tmpa_seq
1551   \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1552     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1553   }
1554   \prop_put:cno {
1555     g_stex_symdecl_
1556     \prop_item:Nn \l_tmpb_prop { symbol }
1557     _prop
1558   } { notations } \exp_not:N \l_tmpa_seq
1559 }
1560
1561 \stex_if_smsmode:TF {
1562   \stex_smsmode_set_codes:
1563   \exp_args:Nx \stex_addtosms:n {
1564     \prop_gset_from_keyval:cn {
1565       g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1566       \c_hash_str \l__stex_notation_lang_str _prop
1567     } {

```

```

1568     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }
1569     language    = \prop_item:Nn \l_tmpb_prop { language }
1570     variant      = \prop_item:Nn \l_tmpb_prop { variant }
1571     opprec       = \prop_item:Nn \l_tmpb_prop { opprec }
1572     argprec      = \prop_item:Nn \l_tmpb_prop { argprec }
1573   }
1574 }
1575 }{
1576   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1577   \seq_put_right:Nx \l_tmpa_seq {
1578     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1579   }
1580   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1581   \prop_set_eq:cN {
1582     g_stex_symdecl_ \l_tmpa_str _prop
1583   } \l_tmpa_prop
1584
1585   % HTML annotations
1586   \stex_annotate_invisible:nnn { notation }
1587   { \prop_item:Nn \l_tmpb_prop { symbol } } {
1588     \stex_annotate_invisible:nnn { notationfragment }
1589     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1590     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1591     \stex_annotate_invisible:nnn { precedence }
1592     { \prop_item:Nn \l_tmpb_prop { opprec };
1593       \seq_use:Nn \l_tmpa_seq { x }
1594     }{}
1595
1596     \int_zero:N \l_tmpa_int
1597     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1598     \tl_clear:N \l_tmpa_tl
1599     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
1600       \int_incr:N \l_tmpa_int
1601       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1602       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1603       \str_if_eq:VnTF \l_tmpb_str a {
1604         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1605           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1606           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1607         } }
1608       }{
1609         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1610           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1611         } }
1612       }
1613     }
1614     \stex_annotate_invisible:nnn { notationcomp }{}{
1615       $ \exp_args:Nno \use:nn { \use:c {
1616         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1617         \c_hash_str \l__stex_notation_variant_str
1618         \c_hash_str \l__stex_notation_lang_str _cs
1619       } } { \l_tmpa_tl } $
1620     }
1621   }

```

```

1622 }
1623 }

```

(End definition for `\_stex_notation_final:.`)

**`\stex_invoke_symbol:n`** Invokes a semantic macro

```

1624 \cs_new_protected:Nn \stex_invoke_symbol:n {
1625   \if_mode_math:
1626     \exp_after:wN \_stex_notation_invoke_math:n
1627   \else:
1628     % TODO
1629     \fi: { #1 }
1630 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 15.)

`\_stex_notation_invoke_math:n`

```

1631 \cs_new_protected:Nn \_stex_notation_invoke_math:n {
1632   \peek_charcode:NTF [ {
1633     \_stex_notation_invoke_math:nw { #1 }
1634   }{
1635     \_stex_notation_invoke_math:nw { #1 } []
1636   }
1637 }

```

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

`\_stex_notation_invoke_math:nw`

```

1638 \cs_new_protected:Npn \_stex_notation_invoke_math:nw #1 [#2] {
1639   \_stex_notation_args:n { #2 }
1640   \prop_set_eq:Nc \l_tmpa_prop {
1641     g_stex_symdecl_ #1 _prop
1642   }
1643   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1644   \seq_if_empty:NTF \l_tmpa_seq {
1645     \msg_set:nnn{stex}{error/nonotations}{
1646       Symbol~#1~used,~but~has~no~notations!
1647     }
1648     \msg_error:nn{stex}{error/nonotations}
1649   } {
1650     \seq_if_in:NxTF \l_tmpa_seq
1651       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
1652         \use:c{
1653           stex_notation_ #1 \c_hash_str
1654           \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1655           _cs
1656         }
1657       }{
1658         \str_if_empty:NTF \l__stex_notation_variant_str {
1659           \str_if_empty:NTF \l__stex_notation_lang_str {
1660             \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
1661             \use:c{
1662               stex_notation_ #1 \c_hash_str \l_tmpa_str
1663               _cs
1664             }

```

```

1665     }{
1666       \msg_set:nnn{stex}{error/wrongnotation}{
1667         Symbol~#1~has~no~notation~
1668         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1669       }
1670       \msg_error:nn{stex}{error/wrongnotation}
1671     }
1672   }{
1673     \msg_set:nnn{stex}{error/wrongnotation}{
1674       Symbol~#1~has~no~notation~
1675       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1676     }
1677     \msg_error:nn{stex}{error/wrongnotation}
1678   }
1679 }
1680 }
1681 }

```

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

**\symdef**

```

1682 \keys_define:nn { stex / symdef } {
1683   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1684   local .bool_set:N = \l_stex_symdecl_local_bool ,
1685   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1686   type .tl_set:N = \l_stex_symdecl_type_tl ,
1687   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1688   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1689   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1690   unknown .code:n = \str_set:Nx
1691     \l__stex_notation_variant_str \l_keys_key_str
1692 }
1693
1694 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
1695   \str_clear:N \l_stex_symdecl_name_str
1696   \str_clear:N \l_stex_symdecl_args_str
1697   \bool_set_false:N \l_stex_symdecl_local_bool
1698   \tl_clear:N \l_stex_symdecl_type_tl
1699   \str_clear:N \l__stex_notation_lang_str
1700   \str_clear:N \l__stex_notation_variant_str
1701   \str_clear:N \l__stex_notation_prec_str
1702
1703   \keys_set:nn { stex / symdef } { #1 }
1704
1705   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1706     \l_stex_symdecl_name_str
1707   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1708     \l_stex_symdecl_args_str
1709   \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
1710     \l__stex_notation_lang_str
1711   \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
1712     \l__stex_notation_variant_str
1713   \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
1714     \l__stex_notation_prec_str

```

```

1715 }
1716
1717 \NewDocumentCommand \symdef { 0{ } m } {
1718   \__stex_notation_symdef_args:n { #1 }
1719   \tl_clear:N \l_stex_symdecl_definiens_tl
1720   \stex_symdecl_do:n { #2 }
1721   \exp_args:Nx \stex_notation_do:nn {
1722     \prop_item:Nn \l_tmpa_prop { module } ?
1723     \prop_item:Nn \l_tmpa_prop { name }
1724   }
1725 }

```

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

## 4.8 Terms

```

1726 <@=stex_term>

```

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
1727 \int_const:Nn \infprec { \c_max_int }
1728 \int_const:Nn \neginfprec { -\c_max_int }
1729 \int_new:N \l__stex_term_downprec
1730 \int_set_eq:NN \l__stex_term_downprec \neginfprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_term_downprec`. These variables are documented on page 16.)

Bracketing:

```

\l_stex_term_left_bracket_str
\l_stex_term_right_bracket_str
1731 \tl_set:Nn \l__stex_term_left_bracket_str (
1732 \tl_set:Nn \l__stex_term_right_bracket_str )
1733 \RequirePackage{scalerel}

```

(End definition for `\l__stex_term_left_bracket_str` and `\l__stex_term_right_bracket_str`.)

`\_stex_term_maybe_brackets:nn` Compares precedences and insert brackets accordingly

```

1734 \cs_new_protected:Nn \_stex_term_maybe_brackets:nn {
1735   \int_compare:nNnTF { #1 } < \l__stex_term_downprec {
1736     \STEXdobrackets { #2 }
1737   }{ #2 }
1738 }

```

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

`\STEXdobrackets`

```

1739 \cs_new_protected:Npn \STEXdobrackets #1 {
1740   \ThisStyle{\if D\m@switch
1741     \exp_args:Nnx \use:nn
1742     { \left\l__stex_term_left_bracket_str #1 }
1743     { \right\l__stex_term_right_bracket_str }
1744   \else
1745     \exp_args:Nnx \use:nn
1746     { \l__stex_term_left_bracket_str #1 }
1747     { \l__stex_term_right_bracket_str }
1748   \fi}
1749 }

```



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

`\STEXwithbrackets`

```

1750 \cs_new_protected:Npn \STEXwithbrackets #1 #2 #3 {
1751   \exp_args:Nnx \use:nn
1752   {
1753     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
1754     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
1755     #3
1756   }
1757   {
1758     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
1759     {\l__stex_term_left_bracket_str}
1760     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
1761     {\l__stex_term_right_bracket_str}
1762   }
1763 }

```

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

OMDOC terms:

`\stex_term_oms:nnnn`

```

1764 \cs_new_protected:Nn \stex_term_oms:nnnn {
1765   \__stex_term_maybe_brackets:nn { #3 }{
1766     \stex_annotate:nnn{OMID}{#1\c_hash_str#2}{#4}
1767   }
1768 }

```

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

`\stex_term_oma:nnnn`

```

1769 \cs_new_protected:Nn \stex_term_oma:nnnn {
1770   \__stex_term_maybe_brackets:nn { #3 }{
1771     \stex_annotate:nnn{OMA}{#1\c_hash_str#2}{#4}
1772   }
1773 }

```

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

`\stex_term_omb:nnnn`

```

1774 \cs_new_protected:Nn \stex_term_omb:nnnn {
1775   \__stex_term_maybe_brackets:nn { #3 }{
1776     \stex_annotate:nnn{OMBIND}{#1\c_hash_str#2}{#4}
1777   }
1778 }

```

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

`\stex_term_arg:nnn`

```

1779 \cs_new_protected:Nn \stex_term_arg:nnn {
1780   \exp_args:Nnx \use:nn
1781   { \int_set:Nn \l__stex_term_downprec { #2 }
1782     \stex_annotate:nnn{arg}{#1}{#3} }
1783   { \int_set:Nn \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
1784 }

```

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

`\stex_term_assoc_arg:nnnn`

```

1785 \cs_new_protected:Nn \stex_term_assoc_arg:nnnn {
1786   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
1787   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
1788     \tl_set:Nn \l_tmpa_tl { #4 }
1789   }{
1790     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
1791     \seq_reverse:N \l_tmpa_seq
1792     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
1793     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
1794     \seq_map_inline:Nn \l_tmpa_seq {
1795       \tl_set:Nx \l_tmpa_tl {
1796         \exp_args:Nno
1797           \l_tmpa_cs { ##1 } { \l_tmpa_tl }
1798       }
1799     }
1800   }
1801   \exp_args:Nnno
1802   \stex_term_arg:nnn{#1}{#2}{ \l_tmpa_tl }
1803 }

```

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