stex.sty: sTeX 2.0[*]

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

August 18, 2021

Abstract

TODO

1 Introduction

TODO

Contents

2 Manual

2.1 Archives and Imports

2.1.1 Namespaces

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

Unfortunately, TeX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that STeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses 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[1].

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.

2.1.2 Paths in Import-Statements

Conversely, here is how namespaces/URIs and file paths are computed in import statements, examplary `\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, is 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.

---

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

- 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

\sTeX
\stex

both print this STEX logo.

\stex_debug:n

\stex_debug:n {⟨*message*⟩}

Logs ⟨*message*⟩, if the package option `debug` is used.

\stex_kpsewhich:n

\stex_kpsewhich:n executes kpsewhich and stores the return in \l_stex_kpsewhich_return_str. This does not require shell escaping.

\stex_addtosms:n

Adds the provided code to the `.sms`-file of the document.

3.1.1 SCALATEX, LATEXML and HTML Annotations

\if@latexml
\latexml_if_p:
\latexml_if:T
\latexml_if:F
\latexml_if:TF

LATEX2e and LATEX3 conditionals for LATEXML.

We have four macros for annotating generated HTML (via LATEXML or SCALATEX) with attributes:

\stex_annotate:nnn
\stex_annotate_invisible:nnn
\stex_annotate_invisible:n

\stex_annotate:nnn {⟨*property*⟩} {⟨*resource*⟩} {⟨*content*⟩}

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.

stex_annotate_env	`\begin{stex_annotate_env}{⟨property⟩}{⟨resource⟩}`
	⟨content⟩
	`\end{stex_annotate_env}`
	behaves like `\stex_annotate:nnn` {⟨property⟩} {⟨resource⟩} {⟨content⟩}.

3.1.2 Languages

`\c_stex_languages_prop` `\c_stex_language_abbrevs_prop`	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:(NV|cn|cV)`	`\stex_path_from_string:Nn` ⟨path-variable⟩ {⟨string⟩} 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` `\stex_path_to_string:N`	The inverse; turns a path into a string and stores it in the second argument variable, or 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` ⋆ `\stex_path_if_absolute:NTF` ⋆	Checks whether the path provided is *absolute*, i.e. starts with an empty segment

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

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

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.

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`,

`\stex_if_in_module_p:` ⋆
`\stex_if_in_module:TF` ⋆

Conditional for whether we are currently in a module

`\stex_if_module_exists_p:n` ⋆
`\stex_if_module_exists:nTF` ⋆

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

`\stex_add_to_current_module:n`

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

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.

3.4.2 SMS Mode

"SMS Mode" is used when loading modules from external tex files. It deactivates any output and ignores all TEX 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.

3.4.3 Imports and Inheritance

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

Imports a module by reading it from a file and "activating" it. sTeX determines the module and its containing file by passing its arguments on to `\stex_import_module_-path:nn`.

`\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 {⟨*archive-ID*⟩} {⟨*module-path*⟩}

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

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

1. If ⟨*archive-ID*⟩ is empty:

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

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

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

2. Otherwise:

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

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

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

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

\stex_import_require_module:nnnn {⟨*ns*⟩} {⟨*archive-ID*⟩} {⟨*path*⟩} {⟨*name*⟩}

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

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

\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 sTeX, but passed on to Mmt for semantic services.

- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.

- **args**: Specifies the "signature" of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:

  - **i** a "normal" argument, e.g. \symdecl[args=ii]{plus} allows for \plus{2}{2}.
  - **a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. \symdecl[args=a]{plus} allows for \plus{2,2,2}.
  - **b** a *variable* argument. Is treated by sTeX like an i-argument, but an application is turned into an **OMBind** in OMDoc, binding the provided variable in the subsequent arguments of the operator; e.g. \symdecl[args=bi]{forall} allows for \forall{x\in\Nat}{x\geq0}.

\stex_symdecl_do:n Implements the core functionality of **\symdecl**, and is called by **\symdecl**, **\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),

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

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

`\stex_notation_do:nn`	`\stex_notation_do:nn{`⟨`URI`⟩`}{`⟨`notations`⟩+`}`

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

Ultimately stores the notation in the property list
`\g_stex_notation_`⟨*URI*⟩`#`⟨*variant*⟩`#`⟨*lang*⟩`_prop` with fields:

- `symbol` (URI string),

- `language` (string),

- `variant` (string),

- `opprec` (integer string),

- `argprecs` (sequence of integer strings)

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

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*⟩.

`\stex_invoke_symbol:n`	

4 Implementation

4.1 The sTEX 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 SТEX 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:Nnnx\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* *4*.)

4.2.1 LaTeXML and SCALATEX

```
64 \RequirePackage{scalatex}
```

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

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

`\if@latexml`   Conditionals for LaTeXML:
`\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* *4*.)

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_ampersand_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:nnn
\stex_annotate_invisible:n
\stex_annotate_invisible:nnn

We define four macros for introducing attributes in the HTML output. The definitions depend on the "backend" used (LaTeXML, SCALATEX, pdflatex).

The pdflatex-macros largely do nothing; the SCALATEX-implementations are pretty clear in what they do, the LaTeXML-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* *4*.)

4.2.3 Languages

```
173 ⟨@@=stex_language⟩
```

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

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

```
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 LaTeX3-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 %% TODO Windows paths
216 \cs_new_protected:Nn \stex_path_from_string:Nn {
217   \exp_args:NNe\str_set:Nn \l_tmpa_tl { #2 }
218   \tl_trim_spaces:N \l_tmpa_tl
219   \str_if_empty:NTF \l_tmpa_tl {
220     \seq_set_eq:NN #1 \c_empty_seq
221   }{
222     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_tl }
223     \stex_path_canonicalize:N #1
224   }
225 }
226 \cs_generate_variant:Nn \stex_path_from_string:Nn
227   { NV, cn, cV }
```

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

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

```
228 \cs_new_protected:Nn \stex_path_to_string:NN {
229   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
230 }
231
232 \cs_new:Nn \stex_path_to_string:N {
233   \seq_use:Nn #1 /
234 }
```

(*End definition for* `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`*. These functions are documented on page 5.*)

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

. and .., respectively.

```
235 \str_const:Nn \c__stex_path_dot_str {.}
236 \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.

```
237 \cs_new_protected:Nn \stex_path_canonicalize:N {
238   \seq_if_empty:NF #1 {
239     \seq_clear:N \l_tmpa_seq
240     \seq_get_left:NN #1 \l_tmpa_tl
241     \str_if_empty:NT \l_tmpa_tl {
242       \seq_put_right:Nn \l_tmpa_seq {}
243     }
244     \seq_map_inline:Nn #1 {
245       \str_set:Nn \l_tmpa_tl { ##1 }
246       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
247         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
```

18

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

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

Test 1

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

.

```
272 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
273   \seq_if_empty:NTF #1 {
274     \prg_return_false:
275   }{
276     \seq_get_left:NN #1 \l_tmpa_tl
277     \str_if_empty:NTF \l_tmpa_tl {
278       \prg_return_true:
279     }{
280       \prg_return_false:
281     }
282   }
283 }
```

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

4.3.2 PWD and kpsewhich

```
284 \str_new:N\l_stex_kpsewhich_return_str
285 \cs_new_protected:Nn \stex_kpsewhich:n {
286   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
287   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
288   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
289 }
```

(*End definition for* `\stex_kpsewhich:n`. *This function is documented on page 4.*)

We determine the PWD

```
290 \sys_if_platform_windows:TF{
291   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
292 }{
293   \stex_kpsewhich:n{-var-value~PWD}
294 }
295
296 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
297 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
298 \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 5.*)

4.3.3 File Hooks and Tracking

```
299 ⟨@@=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 sTEX-purposes.

keeps track of file changes

```
300 \seq_gclear_new:N\g__stex_files_stack
```

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

`\c_stex_mainfile_seq`

```
301 \stex_path_from_string:Nn \c_stex_mainfile_seq {
302   \c_stex_pwd_str/\g_file_curr_name_str.tex
303 }
```

(End definition for `\c_stex_mainfile_seq`. *This variable is documented on page 5.)*

`\g_stex_currentfile_seq`   Hooks for file inputs that push/pop `\g__stex_files_stack` to update `\c_stex_-mainfile_seq`.

```
304 \seq_gclear_new:N\g_stex_currentfile_seq
305 \AddToHook{file/before}{
306   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
307   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
308     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
309   }{
310     \stex_path_from_string:Nn\g_stex_currentfile_seq{
311       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
312     }
313   }
314   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
315   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
316 }
317 \AddToHook{file/after}{
318   \seq_if_empty:NF\g__stex_files_stack{
319     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
320   }
321   \seq_if_empty:NTF\g__stex_files_stack{
322     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
323   }{
324     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
325     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
326   }
327 }
```

(End definition for `\g_stex_currentfile_seq`. *This variable is documented on page 5.)*

4.4   MathHub Repositories

```
328 ⟨@@=stex_mathhub⟩
```

`\mathhub`
`\c_stex_mathhub_seq`
`\c_stex_mathhub_str`

```
329 \str_if_empty:NTF\mathhub{
330   \stex_kpsewhich:n{-var-value~MATHHUB}
331   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
332
333   \str_if_empty:NTF\c_stex_mathhub_str{
334     \msg_warning:nn{stex}{warning/nomathhub}
335   }{
336     \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
337     \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
338   }
339 }{
```

```
340    \stex_path_from_string:Nn\c_stex_mathhub_seq\mathhub
341    \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
342    \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
343  }
```

(*End definition for* \mathhub, \c_stex_mathhub_seq, *and* \c_stex_mathhub_str. *These variables are documented on page* 6.)

__stex_mathhub_do_manifest:n

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

(*End definition for* __stex_mathhub_do_manifest:n.)

\l__stex_mathhub_manifest_file_seq

```
362  \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:

```
363  \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
364    \seq_set_eq:NN\l_tmpa_seq #1
365    \bool_set_true:N\l_tmpa_bool
366    \bool_while_do:Nn \l_tmpa_bool {
367      \seq_if_empty:NTF \l_tmpa_seq {
368        \bool_set_false:N\l_tmpa_bool
369      }{
370        \file_if_exist:nTF{
371          \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
372        }{
373          \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
374          \bool_set_false:N\l_tmpa_bool
375        }{
376          \file_if_exist:nTF{
377            \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
378          }{
379            \seq_put_right:Nn\l_tmpa_seq{META-INF}
380            \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
```

```
381        \bool_set_false:N\l_tmpa_bool
382      }{
383        \file_if_exist:nTF{
384          \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
385        }{
386          \seq_put_right:Nn\l_tmpa_seq{meta-inf}
387          \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
388          \bool_set_false:N\l_tmpa_bool
389        }{
390          \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
391        }
392      }
393    }
394  }
395  }
396  \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
397 }
```

(End definition for __stex_mathhub_find_manifest:N.*)*

\c__stex_mathhub_manifest_ior   File variable used for `MANIFEST`-files

```
398 \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:

```
399 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
400   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
401   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
402   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
403     \str_set:Nn \l_tmpa_str {##1}
404     \exp_args:NNoo \seq_set_split:Nnn
405        \l_tmpb_seq \c_colon_str \l_tmpa_str
406     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
407       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
408         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
409       }
410       \exp_args:No \str_case:nnTF \l_tmpa_tl {
411         {id} {
412           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
413             { id } \l_tmpb_tl
414         }
415         {narration-base} {
416           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
417             { narr } \l_tmpb_tl
418         }
419         {source-base} {
420           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
421             { ns } \l_tmpb_tl
422         }
423         {ns} {
424           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
425             { ns } \l_tmpb_tl
426         }
```

```
427        {dependencies} {
428          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
429            { deps } \l_tmpb_tl
430        }
431      }{}{}
432    }{}
433  }
434  \ior_close:N \c__stex_mathhub_manifest_ior
435 }
```

(*End definition for* __stex_mathhub_parse_manifest:n.)

```
436 \cs_new_protected:Nn \stex_set_current_repository:n {
437   \stex_require_repository:n { #1 }
438   \prop_set_eq:Nc \l_stex_current_repository_prop {
439     c_stex_mathhub_#1_manifest_prop
440   }
441 }
```

(*End definition for* \stex_set_current_repository:n. *This function is documented on page* 6.)

```
442 \cs_new_protected:Nn \stex_require_repository:n {
443   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
444     \stex_debug:n{Opening~archive:~#1}
445     \__stex_mathhub_do_manifest:n { #1 }
446     \exp_args:Nx \stex_addtosms:n {
447       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
448         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } {  id  } ,
449         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } {  ns  } ,
450         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
451         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
452       }
453     }
454   }
455 }
```

(*End definition for* \stex_require_repository:n. *This function is documented on page* 6.)

> **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:
> ```

.

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

```
456 \prop_new:N \l_stex_current_repository_prop
457 \AddToHook{begindocument}{
458   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
459   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
460     \stex_debug:n{Not~currently~in~a~MathHub~repository}
461   } {
462     \__stex_mathhub_parse_manifest:n { main }
463     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
464       \l_tmpa_str
465     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
466     \stex_set_current_repository:n { main }
467     \stex_debug:n{Current~repository:~
468       \prop_item:Nn \l_stex_current_repository_map {id}
469     }
470   }
471 }
```

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

4.5   Module System

```
472 ⟨@@=stex_module⟩
```

\l_stex_current_module_prop

```
473 \prop_new:N \l_stex_current_module_prop
```

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

stex_if_in_module_p:
stex_if_in_module:_TF_

```
474 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
475   \prop_if_empty:NTF \l_stex_current_module_prop
476     \prg_return_false: \prg_return_true:
477 }
```

(*End definition for* `stex_if_in_module:TF`*. This function is documented on page 7.*)

stex_if_module_exists_p:n
stex_if_module_exists:n_TF_

```
478 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
479   \prop_if_exist:cTF { c_stex_module_#1_prop }
480     \prg_return_true: \prg_return_false:
481 }
```

(*End definition for* `stex_if_module_exists:nTF`*. This function is documented on page 7.*)

\stex_add_to_current_module:n

```
482 \cs_new_protected:Nn \stex_add_to_current_module:n {
483   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
484   \tl_put_right:Nn \l_tmpa_tl { #1 }
485   \prop_put:Nno \l_stex_current_module_prop { content } \l_tmpa_tl
486 }
```

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

```
487 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
488    \str_set:Nx \l_tmpa_str { #1 }
489    \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
490    \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
491    \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
492 }
```

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

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

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

stores its return values in:

```
499 \str_new:N \l_stex_modules_ns_str

500 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
501    \str_set:Nx \l_tmpa_str { #1 }
502    \seq_set_eq:NN \l_tmpa_seq #2
503    % split off file extension
504    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
505    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
506    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
507    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
508
509    \bool_set_true:N \l_tmpa_bool
510    \bool_while_do:Nn \l_tmpa_bool {
511      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
512      \exp_args:No \str_case:nnTF { \l_tmpb_str } {
513        {source} { \bool_set_false:N \l_tmpa_bool }
514      }{}{
515        \seq_if_empty:NT \l_tmpa_seq {
516          \bool_set_false:N \l_tmpa_bool
517        }
518      }
519    }
520
521    \seq_if_empty:NTF \l_tmpa_seq {
522      \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
523    }{
524      \str_set:Nx \l_stex_modules_ns_str {
525        \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
526      }
527    }
528 }
```

26

(*End definition for* `\stex_modules_compute_namespace:nN` *and* `\l_stex_modules_ns_str`. *These functions are documented on page* 8.)

`\stex_modules_current_namespace:`

```
529 \cs_new_protected:Nn \stex_modules_current_namespace: {
530   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
531     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
532   }{
533     % split off file extension
534     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
535     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
536     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
537     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
538     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
539     \str_set:Nx \l_stex_modules_ns_str {
540       file:/\stex_path_to_string:N \l_tmpa_seq
541     }
542   }
543 }
```

(*End definition for* `\stex_modules_current_namespace:`. *This function is documented on page* 8.)

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
```

.

4.5.1 The module environment

module   module arguments:

```
544 \keys_define:nn { stex / module } {
545   title .tl_set_x:N  = \l_stex_module_title_str ,
546   ns    .tl_set_x:N  = \l_stex_module_ns_str ,
547   lang  .tl_set_x:N  = \l_stex_module_lang_str ,
548   sig   .tl_set_x:N  = \l_stex_module_sig_str ,
549   meta  .tl_set_x:N  = \l_stex_module_meta_str
550 }
551
552 % module parameters here? In the body?
```

27

```
553
554 \cs_new_protected:Nn \__stex_module_args:n {
555   \str_clear:N \l_stex_module_title_str
556   \str_clear:N \l_stex_module_ns_str
557   \str_clear:N \l_stex_module_lang_str
558   \str_clear:N \l_stex_module_sig_str
559   \str_clear:N \l_stex_module_meta_str
560   \keys_set:nn { stex / module } { #1 }
561   \exp_args:NNo \str_set:Nn \l_stex_module_title_str
562     \l_stex_module_title_str
563   \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
564     \l_stex_module_ns_str
565   \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
566     \l_stex_module_lang_str
567   \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
568     \l_stex_module_sig_str
569   \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
570     \l_stex_module_meta_str
571 }
```

`\__stex_module_begin_module:` implements `\begin{module}`

```
572 \cs_new_protected:Nn \__stex_module_begin_module: {
573   % Nested module?
574   \stex_if_in_module:TF {
575     % Nested module
576     \prop_get:NnN \l_stex_current_module_prop
577       { ns } \l_stex_module_ns_str
578     \str_set:Nx \l_stex_module_name_str {
579       \prop_item:Nn \l_stex_current_module_prop
580         { name } / \l_stex_module_name_str
581     }
582   }{
583     % not nested:
584     \str_if_empty:NT \l_stex_module_ns_str {
585       \stex_modules_current_namespace:
586       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
587       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
588         / {\l_stex_module_ns_str}
589       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
590       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
591         \str_set:Nx \l_stex_module_ns_str {
592           \stex_path_to_string:N \l_tmpa_seq
593         }
594       }
595     }
596   }
597
598   % language
599   \str_if_empty:NF \l_stex_module_lang_str {
600     \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
601       \l_tmpa_str {
602         \exp_args:Nx \selectlanguage { \l_tmpa_str }
603       } {
604         \msg_set:nnn{stex}{error/unknownlanguage}{
```

```
605          Unknown~language~\l_tmpa_str
606        }
607        \msg_error:nn{stex}{error/unknownlanguage}
608      }
609    }
610
611    % signature
612    \str_if_empty:NF \l_stex_module_sig_str {
613      \str_if_empty:NT \l_stex_module_lang_str {
614        \msg_set:nnn{stex}{error/siglanguage}{
615          Module~\l_stex_module_ns_str?\l_stex_module_name_str~
616          declares~signature~\l_stex_module_sig_str,~but~does~not~
617          declare~its~language
618        }
619        \msg_error:nn{stex}{error/siglanguage}
620      }
621    }
622
623    % metatheory
624 %  \str_if_empty:NTF \l_stex_module_meta_str {
625 %
626 %  } {
627 %
628 %  }
629
630    \str_clear:N \l_tmpa_str
631    \seq_clear:N \l_tmpa_seq
632    \tl_clear:N \l_tmpa_tl
633    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
634      name      = \l_stex_module_name_str ,
635      ns        = \l_stex_module_ns_str ,
636      imports   = \exp_not:o { \l_tmpa_seq } ,
637      constants = \exp_not:o { \l_tmpa_seq } ,
638      content   = \exp_not:o { \l_tmpa_tl }  ,
639      file      = \exp_not:o { \g_stex_currentfile_seq } ,
640      lang      = \l_stex_module_lang_str ,
641      sig       = \l_stex_module_sig_str ,
642      meta      = \l_stex_module_meta_str
643    }
644
645    \stex_debug:n{
646      New~module:\\
647      Namespace:~\l_stex_module_ns_str\\
648      Name:~\l_stex_module_name_str\\
649      Language:~\l_stex_module_lang_str\\
650      Signature:~\l_stex_module_sig_str\\
651      Metatheory:~\l_stex_module_meta_str\\
652      File:~\stex_path_to_string:N \g_stex_currentfile_seq
653    }
654
655    \seq_gput_right:Nx  \g_stex_modules_in_file_seq
656        { \l_stex_module_ns_str ? \l_stex_module_name_str }
657
658    \stex_if_smsmode:TF {
```

29

```
659        \stex_smsmode_set_codes:
660     } {
661       \begin{stex_annotate_env} {theory} {
662         \l_stex_module_ns_str ? \l_stex_module_name_str
663       }
664
665       \stex_annotate_invisible:nnn{header}{} {
666         \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
667         \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
668         \str_if_empty:NT \l_stex_module_meta_str {
669           % TODO metatheory
670         }
671       }
672     }
673 }
674 \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}`

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

(*End definition for* `\__stex_module_end_module:`.)

@module The core environment, with no header

```
708 \NewDocumentEnvironment { @module } { O{} m } {
709   \str_set:Nx \l_stex_module_name_str { #2 }
710   \par
711   \__stex_module_args:n { #1 }
712   \__stex_module_begin_module:
713 } {
714   \__stex_module_end_module:
715 }
```

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:
```

.

\stex_modules_heading: Code for document headers

```
716 \cs_if_exist:NTF \thesection {
717   \newcounter{module}[section]
718 }{
719   \newcounter{module}
720 }
721
722 \bool_if:NT \c_stex_showmods_bool {
723   \latexml_if:F { \RequirePackage{mdframed} }
724 }
725
726 \cs_new_protected:Nn \stex_modules_heading: {
727   \stepcounter{module}
728   \par
729   \bool_if:NT \c_stex_showmods_bool {
730     \noindent{\textbf{Module} ~
731       \cs_if_exist:NT \thesection {\thesection.}
732       \themodule ~ [\l_stex_module_name_str]
733     }
734     % TODO references
735     % \sref@label@id{Module \thesection.\themodule [\module@name]}%
736     \str_if_empty:NTF \l_stex_module_title_str {
```

```
737        }{
738            \quad(\l_stex_module_title_str)\hfill
739        }\par
740    }
741 }
```

(End definition for `\stex_modules_heading:`. *This function is documented on page* *8*.)

Finally:

```
742 \NewDocumentEnvironment { module } { O{} m } {
743    \bool_if:NT \c_stex_showmods_bool {
744        \begin{mdframed}
745    }
746    \begin{@module}[#1]{#2}
747    \stex_modules_heading:
748 }{
749    \end{@module}
750    \bool_if:NT \c_stex_showmods_bool {
751        \end{mdframed}
752    }
753 }
```

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

.

4.5.2   SMS Mode

```
754 ⟨@@=stex_smsmode⟩
```

```
755 \tl_new:N \g_stex_smsmode_allowedmacros_tl
756 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
757 \seq_new:N \g_stex_smsmode_allowedenvs_seq
758
```

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

```
813 %    modnl,
814 %      @structural@feature
815   }
816 }
```

(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 8.)*

`\stex_if_smsmode_p:`
`\stex_if_smsmode:TF`

```
817 \bool_new:N \g__stex_smsmode_bool
818 \bool_set_false:N \g__stex_smsmode_bool
819 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
820   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
821 }
```

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

`\__stex_smsmode_if_catcodes_p:`
`\__stex_smsmode_if_catcodes:TF` Checks whether the SMS mode category code scheme is active.

```
822 \bool_new:N \g__stex_smsmode_catcode_bool
823 \bool_set_false:N \g__stex_smsmode_catcode_bool
824 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
825   \bool_if:NTF \g__stex_smsmode_catcode_bool
826     \prg_return_true: \prg_return_false:
827 }
```

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

`\stex_smsmode_set_codes:`

```
828 \cs_new_protected:Nn \stex_smsmode_set_codes: {
829   \stex_if_smsmode:T {
830     \__stex_smsmode_if_catcodes:F {
831       \bool_gset_true:N \g__stex_smsmode_catcode_bool
832       \exp_after:wN \char_gset_active_eq:NN
833         \c_backslash_str \__stex_smsmode_cs:
834       \tex_global:D \char_set_catcode_active:N \\
835       \tex_global:D \char_set_catcode_other:N $
836       \tex_global:D \char_set_catcode_other:N ^
837       \tex_global:D \char_set_catcode_other:N _
838       \tex_global:D \char_set_catcode_other:N &
839       \tex_global:D \char_set_catcode_other:N ##
840     }
841   }
842 } \iffalse $ \fi % to make syntax highlighting work again
```

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

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

```
843 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
844   \__stex_smsmode_if_catcodes:T {
845     \bool_gset_false:N \g__stex_smsmode_catcode_bool
846     \exp_after:wN \tex_global:D \exp_after:wN
847       \char_set_catcode_escape:N \c_backslash_str
848     \tex_global:D \char_set_catcode_math_toggle:N $
849     \tex_global:D \char_set_catcode_math_superscript:N ^
850     \tex_global:D \char_set_catcode_math_subscript:N _
```

```
851      \tex_global:D \char_set_catcode_alignment:N &
852      \tex_global:D \char_set_catcode_parameter:N ##
853    }
854  } \iffalse $ \fi % to make syntax highlighting work again
```

(*End definition for* __stex_smsmode_unset_codes:.)

\stex_in_smsmode:nn

```
855  \cs_new_protected:Nn \stex_in_smsmode:nn {
856    \vbox_set:Nn \l_tmpa_box {
857      \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
858      \bool_gset_true:N \g__stex_smsmode_bool
859      \stex_smsmode_set_codes:
860      #2
861      \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
862      \stex_if_smsmode:F {
863        \__stex_smsmode_unset_codes:
864      }
865    }
866    \box_clear:N \l_tmpa_box
867  }
```

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

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

```
868  \str_const:Nn \c__stex_smsmode_begin_str { begin }
869  \str_const:Nn \c__stex_smsmode_end_str { end }
870
871  \cs_new_protected:Nn \__stex_smsmode_cs: {
872    \str_clear:N \l_tmpa_str
873    \peek_analysis_map_inline:n {
874      % #1: token (one expansion)
875      % #2: charcode
876      % #3 catcode
877      \token_if_eq_charcode:NNTF ##3 B {
878        % token is a letter
879        \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
880      } {
881        \str_if_empty:NTF \l_tmpa_str {
882          % we don't allow (or need) single non-letter CSs
883          % for now
884          \peek_analysis_map_break:
885        }{
886          \str_if_eq:nnTF \l_tmpa_str \c_stex_begin_str {
887            \peek_analysis_map_break:n {
888              \exp_after:wN \__stex_smsmode_checkbegin:n ##1
889            }
890          } {
891            \str_if_eq:nnTF \l_tmpa_str \c_stex_end_str {
892              \peek_analysis_map_break:n {
893                \exp_after:wN \__stex_smsmode_checkend:n ##1
894              }
895            } {
```

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

(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:

```
929 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
930   \str_clear:N \l_tmpb_str
931   \peek_analysis_map_inline:n {
932     \token_if_eq_charcode:NNTF ##3 B {
933       % token is a letter
934       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
935     } {
936       \peek_analysis_map_break:n {
937         \exp_after:wN \use:c \exp_after:wN {
938           \exp_after:wN \l_tmpa_str\exp_after:wN
939         } \use:c { \l_tmpb_str \exp_after:wN } ##1
940       }
941     }
942   }
943 }
```

(End definition for `\__stex_smsmode_rescan_cs:`*.)*

36

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

```
944 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
945   \str_set:Nn \l_tmpa_str { #1 }
946   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
947     \__stex_smsmode_unset_codes:
948     \begin{#1}
949   }
950 }
```

(*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.

```
951 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
952   \str_set:Nn \l_tmpa_str { #1 }
953   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
954     \end{#1}
955   }
956 }
```

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

> **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
> ```
>

.

4.5.3 Inheritance

```
957 ⟨@@=stex_importmodule⟩
```

`\stex_import_module_uri:nn`

```
958 \cs_new_protected:Nn \stex_import_module_uri:nn {
959   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
960   \str_set:Nx \l__stex_importmodule_path_str { #2 }
961   \str_if_empty:NT \l__stex_importmodule_archive_str {
962     \prop_if_empty:NF \l_stex_current_repository_prop {
963       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
964     }
965   }
966
967   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
968   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
969   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpa_seq ? }
970
971   \str_if_empty:NTF \l_tmpa_str {
```

```
972        \stex_modules_current_namespace:
973        \str_if_empty:NF \l__stex_importmodule_path_str {
974          \str_set:Nx \l_stex_module_ns_str {
975            \l_stex_module_ns_str / \l__stex_importmodule_path_str
976          }
977        }
978      }{
979        \stex_require_repository:n \l__stex_importmodule_archive_str
980        \prop_get:cnN { c_stex_mathhub_\l__stex_importmodule_archive_str _manifest_prop } { ns }
981          \l_stex_module_ns_str
982        \str_if_empty:NF \l__stex_importmodule_path_str {
983          \str_set:Nx \l__stex_importmodule_module_ns_str {
984            \l_stex_module_ns_str / \l__stex_importmodule_path_str ? \l__stex_importmodule_name_
985          }
986        }
987      }
988    }
```

(*End definition for* `\stex_import_module_uri:nn`. *This function is documented on page* *10*.)

`\l__stex_importmodule_name_str`
`\l__stex_importmodule_archive_str`
`\l__stex_importmodule_path_str`

Store the return values of `\stex_import_module_uri:nn`.

```
989  \str_new:N \l__stex_importmodule_name_str
990  \str_new:N \l__stex_importmodule_archive_str
991  \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⟩}

```
992  \cs_new_protected:Nn \stex_import_require_module:nnnn {
993    \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
994      % archive
995      \str_set:Nx \l_tmpa_str { #2 }
996      \str_if_empty:NTF \l_tmpa_str {
997        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
998      } {
999        \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
1000        \exp_args:NNo \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1001        \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1002        \seq_put_right:Nn \l_tmpa_seq { source }
1003      }
1004
1005      \stex_debug:n{Arguments: #1, #2, #3, #4}
1006
1007      % path
1008      \str_set:Nx \l_tmpb_str { #3 }
1009      \str_if_empty:NT \l_tmpb_str {
1010        \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1011
1012        \cs_if_exist:NTF \languagename {
1013          \prop_get:NnN \c_stex_language_abbrevs_prop
1014              { \languagename } \l_tmpb_str
1015        }
1016
```

```
1017        \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1018        \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1019          \str_set:Nx \l_tmpa_str { \l_tmpa_str.\l_tmpb_str.tex }
1020        }{
1021          \stex_debug:n{Checking~\l_tmpa_str.tex}
1022          \IfFileExists{ \l_tmpa_str.tex }{
1023            \str_set:Nx \l_tmpa_str { \l_tmpa_str.tex }
1024          }{
1025            % try english as default
1026            \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1027            \IfFileExists{ \l_tmpa_str.en.tex }{
1028              \str_set:Nx \l_tmpa_str { \l_tmpa_str.en.tex }
1029            }{
1030              \msg_new:nnn{stex}{error/modulemissing}{
1031                No~file~for~module~#1?#4~found
1032              }
1033              \msg_error:nn{stex}{error/modulemissing}
1034            }
1035          }
1036        }

1037
1038      } {
1039        \exp_args:NNo \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpb_str }
1040        \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq

1041
1042        \cs_if_exist:NTF \languagename {
1043          \prop_get:NnN \c_stex_language_abbrevs_prop
1044              { \languagename } \l_tmpb_str
1045        }

1046
1047        \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq }

1048
1049        \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1050        \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1051          \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1052        }{
1053          \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1054          \IfFileExists{ \l_tmpa_str/#4.tex }{
1055            \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.tex }
1056          }{
1057            % try english as default
1058            \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1059            \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1060              \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.en.tex }
1061            }{
1062              \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1063              \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1064                \str_set:Nx \l_tmpa_str { \l_tmpa_str.\l_tmpb_str.tex }
1065              }{
1066                \stex_debug:n{Checking~\l_tmpa_str.tex}
1067                \IfFileExists{ \l_tmpa_str.tex }{
1068                  \str_set:Nx \l_tmpa_str { \l_tmpa_str.tex }
1069                }{
1070                  % try english as default
```

```
1071                         \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1072                         \IfFileExists{ \l_tmpa_str.en.tex }{
1073                           \str_set:Nx \l_tmpa_str { \l_tmpa_str.en.tex }
1074                         }{
1075                           \msg_new:nnn{stex}{error/modulemissing}{
1076                             No~file~for~module~#1?#4~found
1077                           }
1078                           \msg_error:nn{stex}{error/modulemissing}
1079                         }
1080                       }
1081                     }
1082                   }
1083                 }
1084               }
1085             }
1086
1087       \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1088       \seq_clear:N \g_stex_modules_in_file_seq
1089       \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1090         \str_set:Nx \l_tmpb_str { #2 }
1091         \str_if_empty:NF \l_tmpb_str {
1092           \stex_set_current_repository:n { #2 }
1093         }
1094         \input { \l_tmpa_str }
1095       }
1096       \prop_gput:Noo \g_stex_module_files_prop
1097         \l_tmpa_str \g_stex_modules_in_file_seq
1098       \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1099
1100       \stex_if_module_exists:nF { #1 ? #4 } {
1101         \msg_new:nnn{stex}{error/modulemissing}{
1102           Module~#1?#4~not~found~in~file~\l_tmpa_str
1103         }
1104         \msg_error:nn{stex}{error/modulemissing}
1105       }
1106       % TODO write to sms file
1107     }
1108     % activate
1109     \stex_debug:n{Activating~module~#1?#4}
1110     \prop_item:cn { c_stex_module_#1?#4_prop } { content }
1111 }
```

(End definition for \stex_import_require_module:nnnn. *This function is documented on page 10.)*

\importmodule

```
1112 \NewDocumentCommand \importmodule { O{} m } {
1113   \stex_import_module_uri:nn { #1 } { #2 }
1114   \stex_debug:n{Importing~module:~
1115     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1116   }
1117   \stex_if_smsmode:F {
1118     \stex_import_require_module:nnnn
1119     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1120     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
```

```
1121        \stex_annotate_invisible:nnn
1122          {import} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1123      }
1124      \exp_args:Nx \stex_add_to_current_module:n {
1125        \stex_import_require_module:nnnn
1126        { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1127        { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1128      }
1129      \exp_args:Nx \stex_add_import_to_current_module:n {
1130        \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1131      }
1132      \stex_smsmode_set_codes:
1133 }
```

(*End definition for* \importmodule. *This function is documented on page* *9.*)

Test 7

```
 \begin{module}{Foo1}
\symdecl[name=foobar, args=3]{bar}
\symdecl[args=bai]{foobar2}
Meaning:~\meaning\bar\\
\end{module}
\begin{module}{Foo2}
\importmodule{Foo1}
Meaning:~\meaning\bar\\
\end{module}
```

> **Module** 4.2[Foo1]
>   Meaning: macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo1?foobar}

> **Module** 4.3[Foo2]
>   Meaning: macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo1?foobar}

.

\usemodule

```
1134 \NewDocumentCommand \usemodule { O{} m } {
1135    \stex_if_smsmode:F {
1136      \stex_import_module_uri:nn { #1 } { #2 }
1137      \stex_import_require_module:nnnn
1138      { \l__stex_importmodule_module_ns_str } { \l__stex_importmodule_archive_str }
1139      { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1140      \stex_annotate_invisible:nnn
1141        {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1142    }
1143    \stex_smsmode_set_codes:
1144 }
```

(*End definition for* \usemodule. *This function is documented on page* *9.*)

\g_stex_modules_in_file_seq
\g_stex_module_files_prop

```
1145 \seq_new:N \g_stex_modules_in_file_seq
1146 \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 [10](#).)*

4.6 Symbol Declarations

```
1147 ⟨@@=stex_symdecl⟩
```

symdecl arguments:
```
1148 \keys_define:nn { stex / symdecl } {
1149   name  .tl_set_x:N  = \l_stex_symdecl_name_str ,
1150   local .bool_set:N  = \l_stex_symdecl_local_bool ,
1151   args  .tl_set_x:N  = \l_stex_symdecl_args_str ,
1152   type  .tl_set:N    = \l_stex_symdecl_type_tl
1153 }
1154
1155 \cs_new_protected:Nn \__stex_symdecl_args:n {
1156   \str_clear:N \l_stex_symdecl_name_str
1157   \str_clear:N \l_stex_symdecl_args_str
1158   \bool_set_false:N \l_stex_symdecl_local_bool
1159   \tl_clear:N \l_stex_symdecl_type_tl
1160
1161   \keys_set:nn { stex /symdecl } { #1 }
1162
1163   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1164     \l_stex_symdecl_name_str
1165   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1166     \l_stex_symdecl_args_str
1167 }
```

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

```
1168 \NewDocumentCommand \symdecl { O{} m } {
1169   \__stex_symdecl_args:n { #1 }
1170   \tl_clear:N \l_stex_symdecl_definiens_tl
1171   \stex_symdecl_do:n { #2 }
1172 }
```

(End definition for `\symdecl`. *This function is documented on page [11](#).)*

`\stex_symdecl_do:n`

```
1173 \cs_new_protected:Nn \stex_symdecl_do:n {
1174   \stex_if_in_module:F {
1175     % TODO throw error? some default namespace?
1176   }
1177
1178   \str_if_empty:NT \l_stex_symdecl_name_str {
1179     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1180   }
1181
1182   \prop_if_exist:cT { g_stex_symdecl_
1183     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1184     \prop_item:Nn \l_stex_current_module_prop {name} ?
1185       \l_stex_symdecl_name_str
1186     _prop
1187   }{
```

```
1188      % TODO throw error (beware of circular dependencies)
1189    }
1190
1191    \prop_clear:N \l_tmpa_prop
1192    \prop_put:Nnx \l_tmpa_prop { module } {
1193      \prop_item:Nn \l_stex_current_module_prop {ns} ?
1194      \prop_item:Nn \l_stex_current_module_prop {name}
1195    }
1196    \seq_clear:N \l_tmpa_seq
1197    \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1198    \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1199    \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1200    \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1201
1202    \exp_args:No \stex_add_constant_to_current_module:n {
1203      \l_stex_symdecl_name_str
1204    }
1205
1206    % arity/args
1207    \int_zero:N \l_tmpb_int
1208
1209    \bool_set_true:N \l_tmpa_bool
1210    \str_map_inline:Nn \l_stex_symdecl_args_str {
1211      \token_case_meaning:NnF ##1 {
1212        0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1213        {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1214        {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1215        {\tl_to_str:n a} {
1216          \bool_set_false:N \l_tmpa_bool
1217          \int_incr:N \l_tmpb_int
1218        }
1219      }{
1220        \msg_set:nnn{stex}{error/wrongargs}{
1221          args~value~in~symbol~declaration~for~
1222          \prop_item:Nn \l_stex_current_module_prop {ns} ?
1223          \prop_item:Nn \l_stex_current_module_prop {name} ?
1224          \l_stex_symdecl_name_str ~
1225          needs~to~be~
1226          i,~a~or~b,~but~##1~given
1227        }
1228        \msg_error:nn{stex}{error/wrongargs}
1229      }
1230    }
1231    \bool_if:NTF \l_tmpa_bool {
1232      % possibly numeric
1233      \str_if_empty:NTF \l_stex_symdecl_args_str {
1234        \prop_put:Nnn \l_tmpa_prop { args } {}
1235        \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1236      }{
1237        \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1238        \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1239        \str_clear:N \l_tmpa_str
1240        \int_step_inline:nn \l_tmpa_int {
1241          \str_put_right:Nn \l_tmpa_str i
```

```
1242        }
1243        \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1244      }
1245    } {
1246      \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1247      \prop_put:Nnx \l_tmpa_prop { arity }
1248        { \str_count:N \l_stex_symdecl_args_str }
1249    }
1250    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }


1252
1253    % semantic macro
1254
1255    \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1256      \prop_item:Nn \l_tmpa_prop { module } ?
1257        \prop_item:Nn \l_tmpa_prop { name }
1258    } }
1259
1260    \bool_if:NF \l_stex_symdecl_local_bool {
1261      \exp_args:Nx \stex_add_to_current_module:n {
1262        \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1263          \prop_item:Nn \l_tmpa_prop { module } ?
1264            \prop_item:Nn \l_tmpa_prop { name }
1265        } }
1266      }
1267    }


1269
1270    \stex_debug:n{New~symbol:~
1271      \prop_item:Nn \l_tmpa_prop { module } ?
1272        \prop_item:Nn \l_tmpa_prop { name }^^J
1273      Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1274      Args:~\prop_item:Nn \l_tmpa_prop { args }
1275    }
1276
1277    \prop_gset_eq:cN {
1278      g_stex_symdecl_
1279      \prop_item:Nn \l_tmpa_prop { module } ?
1280      \prop_item:Nn \l_tmpa_prop { name }
1281      _prop
1282    } \l_tmpa_prop
1283
1284    \stex_if_smsmode:TF {
1285      \bool_if:NF \l_stex_symdecl_local_bool {
1286        \exp_args:Nx \stex_addtosms:n {
1287          \prop_gset_from_keyval:cn {
1288            g_stex_symdecl_
1289            \prop_item:Nn \l_tmpa_prop { module } ?
1290            \prop_item:Nn \l_tmpa_prop { name }
1291            _prop
1292          } {
1293            name      = \prop_item:Nn \l_tmpa_prop { name }
1294            module    = \prop_item:Nn \l_tmpa_prop { module }
1295            notations = \prop_item:Nn \l_tmpa_prop { notations }
```

```
1296              local    = \prop_item:Nn \l_tmpa_prop { local }
1297              type     = \prop_item:Nn \l_tmpa_prop { type }
1298              args     = \prop_item:Nn \l_tmpa_prop { args }
1299              arity    = \prop_item:Nn \l_tmpa_prop { arity }
1300              assocs   = \prop_item:Nn \l_tmpa_prop { assocs }
1301            }
1302          }
1303        }
1304      \stex_smsmode_set_codes:
1305    }{
1306      \stex_annotate_invisible:nnn {symdecl} {
1307        \prop_item:Nn \l_tmpa_prop { module } ?
1308        \prop_item:Nn \l_tmpa_prop { name }
1309      } {
1310        \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
1311        \stex_annotate_invisible:nnn{args}{}{
1312          \prop_item:Nn \l_tmpa_prop { args }
1313        }
1314        \stex_annotate_invisible:nnn{macroname}{}{#1}
1315        \str_if_empty:NF \l_stex_symdecl_definiens_tl {
1316          \stex_annotate_invisible:nnn{definiens}{}
1317            {$\l_stex_symdecl_definiens_tl$}
1318        }
1319      }
1320    }
1321  }
```

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

`\stex_get_symbol:n`

```
1322  \str_new:N \l_stex_get_symbol_uri_str
1323
1324  \cs_new_protected:Nn \stex_get_symbol:n {
1325    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1326      % argument is a command
1327      % TODO
1328    }{
1329      % argument is a string
1330      % is it a command name?
1331      \tl_set:Nx \l_tmpa_tl { \use:c { #1 } }
1332
1333      \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
1334        \stex_invoke_symbol:n {
1335        \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1336          { \tl_tail:N \l_tmpa_tl }
1337        \tl_if_single:NTF \l_tmpa_tl {
1338          \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1339            \exp_after:wN \str_set:Nn \exp_after:wN
1340              \l_stex_get_symbol_uri_str \l_tmpa_tl
1341          }{
1342            % TODO
1343            % tail is not a single group
1344          }
1345        }{
```

```
1346        % TODO
1347        % tail is not a single group
1348      }
1349    }{
1350      % TODO
1351      % head is not \stex_invoke_symbol:n
1352    }
1353  }
1354 }
```

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

Test 8

```
 \begin{module}{Foo3}
\symdecl[name=foobar, args=3]{bar}
\symdecl[name=foobar2, args=iab]{bari}
\ExplSyntaxOn
Meaning:~\meaning\bar\\
\stex_get_symbol:n { bar }
Result:~\l__stex_get_symbol_uri_str
\ExplSyntaxOff
\end{module}
```

> **Module** 4.4[Foo3]
>   Meaning: macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-
> master/stextest?Foo3?foobar}
> Result: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo3?foobar

.

4.7 Notations

```
1355 ⟨@@=stex_notation⟩
```

notation arguments:
```
1356 \keys_define:nn { stex / notation } {
1357   lang    .tl_set_x:N = \l__stex_notation_lang_str ,
1358   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1359   prec    .tl_set_x:N = \l__stex_notation_prec_str ,
1360   unknown .code:n     = \str_set:Nx
1361       \l__stex_notation_variant_str \l_keys_key_str
1362 }
1363
1364 \cs_new_protected:Nn \__stex_notation_args:n {
1365   \str_clear:N \l__stex_notation_lang_str
1366   \str_clear:N \l__stex_notation_variant_str
1367   \str_clear:N \l__stex_notation_prec_str
1368
1369   \keys_set:nn { stex / notation } { #1 }
1370
1371   \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
1372     \l__stex_notation_lang_str
1373   \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
1374     \l__stex_notation_variant_str
1375   \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
```

```
1376        \l__stex_notation_prec_str
1377    }
```

\notation

```
1378    \NewDocumentCommand \notation { O{} m } {
1379      \__stex_notation_args:n { #1 }
1380      \tl_clear:N \l_stex_symdecl_definiens_tl
1381      \stex_get_symbol:n { #2 }
1382      \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1383    }
```

(*End definition for* \notation. *This function is documented on page* *12*.)

\stex_notation_do:nn

```
1384    \cs_new_protected:Nn \stex_notation_do:nn {
1385      \prop_set_eq:Nc \l_tmpa_prop {
1386        g_stex_symdecl_ #1 _prop
1387      }
1388
1389      \prop_clear:N \l_tmpb_prop
1390      \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1391      \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1392      \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1393
1394      % precedences
1395      \seq_clear:N \l_tmpb_seq
1396      \exp_args:NNno
1397      \seq_set_split:Nnn \l_tmpa_seq ; { \l__stex_notation_prec_str }
1398      \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1399        \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1400        \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1401          \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1402            \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
1403          \seq_map_inline:Nn \l_tmpa_seq {
1404            \seq_put_right:Nn \l_tmpb_seq { ##1 }
1405          }
1406        }
1407      }{
1408        \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1409        % TODO set to (+/-)infprec if arity==0
1410      }
1411
1412      \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1413      \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1414      \int_step_inline:nn { \l_tmpa_str } {
1415        \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
1416          \seq_put_right:Nn \l_tmpb_seq { 0 }
1417        }
1418      }
1419
1420      \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
1421
1422      \int_compare:nNnTF \l_tmpa_str = 0 {
1423        \cs_set:Npx \l__stex_notation_macrocode_cs {} {
1424          \stex_term_oms:nnnn { #1 }
```

```
1425            { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1426            { \prop_item:Nn \l_tmpb_prop { opprec } }
1427            { #2 }
1428        }
1429        \__stex_notation_final:
1430    }{
1431      \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1432      \str_if_in:NnTF \l_tmpb_str b {
1433        \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1434        \cs_set:Npx \l_tmpa_str {
1435          \stex_term_omb:nnnn { #1 }
1436            { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1437            { \prop_item:Nn \l_tmpb_prop { opprec } }
1438            { #2 }
1439        }
1440      }{
1441        \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1442        \cs_set:Npx \l_tmpa_str {
1443          \stex_term_oma:nnnn { #1 }
1444            { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1445            { \prop_item:Nn \l_tmpb_prop { opprec } }
1446            { #2 }
1447        }
1448      }
1449
1450      \int_zero:N \l_tmpa_int
1451      \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1452      \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1453      \tl_clear:N \l_tmpa_tl
1454      \__stex_notation_arguments:
1455    }
1456 }
```

(*End definition for* \stex_notation_do:nn. *This function is documented on page* *12.*)

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

```
1457 \cs_new_protected:Nn \__stex_notation_arguments: {
1458    \int_incr:N \l_tmpa_int
1459    \str_if_empty:NTF \l_tmpa_str {
1460      \__stex_notation_final:
1461    }{
1462      \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1463      \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1464 %    \exp_args:NNx
1465      \str_if_eq:VnTF \l_tmpb_str a { %{\tl_to_str:n{a}} {
1466        \__stex_notation_argument_assoc:n
1467      }{
1468        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1469        \tl_put_right:Nx \l_tmpa_tl {
1470          { \stex_term_arg:nnn
1471            { \int_use:N \l_tmpa_int }
1472            { \l_tmpb_str }
1473            { ####\int_use:N \l_tmpa_int }
1474          }
```

```
1475        }
1476        \__stex_notation_arguments:
1477      }
1478    }
1479 }
```

(*End definition for __stex_notation_arguments:.*)

__stex_notation_argument_assoc:n

```
1480 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1481   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1482   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1483   \tl_put_right:Nx \l_tmpa_tl {
1484     { \stex_term_assoc_arg:nnnn
1485       { \int_use:N \l_tmpa_int }
1486       { \l_tmpb_str }
1487       { \l_tmpa_cs {########1} {########2} }
1488       { ####\int_use:N \l_tmpa_int }
1489     }
1490   }
1491   \__stex_notation_arguments:
1492 }
```

(*End definition for __stex_notation_argument_assoc:n.*)

__stex_notation_final: Called after processing all notation arguments

```
1493 \cs_new_protected:Nn \__stex_notation_final: {
1494   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1495   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1496   \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1497   \cs_generate_from_arg_count:cNnn {
1498     stex_notation_ \l_tmpa_str \c_hash_str
1499     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1500     _cs
1501   }
1502   \cs_set:Npx \l_tmpb_str {
1503     \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl
1504   }
1505
1506   \stex_debug:n{
1507     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1508     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1509     Operator~precedence:~
1510       \prop_item:Nn \l_tmpb_prop { opprec }^^J
1511     Argument~precedences:~
1512       \seq_use:Nn \l_tmpa_seq {,~}^^J
1513     Notation: \cs_meaning:c {
1514       stex_notation_ \l_tmpa_str \c_hash_str
1515       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1516       _cs
1517     }
1518   }
1519
1520   \prop_gset_eq:cN {
```

49

```
1521      g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1522         \c_hash_str \l__stex_notation_lang_str _prop
1523    } \l_tmpb_prop
1524
1525    \stex_if_smsmode:TF {
1526      \stex_smsmode_set_codes:
1527      % TODO: sms file, module content, HTML annotations
1528    }{
1529      \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1530      \seq_put_right:Nx \l_tmpa_seq {
1531        \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1532      }
1533      \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1534      \prop_set_eq:cN {
1535        g_stex_symdecl_ \l_tmpa_str _prop
1536      } \l_tmpa_prop
1537    }
1538
1539 }
```

(*End definition for* `\__stex_notation_final:`.)

Test 9

```
\begin{module}{Foo4}
\importmodule{Foo1}
\notation[foo, prec=500;20x20x20]{bar}{\langle {#1 ^ {#2}}_{#3} \rangle }
\notation[foo, prec=500;20x20x20]{foobar2}{\langle #1 \mid [ #2 ]^{#3} \rangle }{ {#1}_{:#2} }
\end{module}
```

> **Module** 4.5[Foo4]

.

`\stex_invoke_symbol:n` Invokes a semantic macro

```
1540 \prg_new_conditional:Nnn \if_mathmode: {p, T, F, TF} {
1541    \if_mode_math:
1542      \prg_return_true:
1543    \else:
1544      \prg_return_false:
1545    \fi:
1546 }
1547
1548 \cs_new_protected:Nn \stex_invoke_symbol:n {
1549    \if_mode_math:
1550      \exp_after:wN \__stex_notation_invoke_math:n
1551    \else:
1552      % TODO
1553    \fi: { #1 }
1554 }
```

(*End definition for* `\stex_invoke_symbol:n`. *This function is documented on page* *12*.)

```
1555 \cs_new_protected:Nn \__stex_notation_invoke_math:n {
1556   \peek_charcode:NTF [ {
1557     \__stex_notation_invoke_math:nw { #1 }
1558   }{
1559     \__stex_notation_invoke_math:nw { #1 } []
1560   }
1561 }
```

(*End definition for* __stex_notation_invoke_math:n.)

```
1562 \cs_new_protected:Npn \__stex_notation_invoke_math:nw  #1 [#2] {
1563   \__stex_notation_args:n { #2 }
1564   \prop_set_eq:Nc \l_tmpa_prop {
1565     g_stex_symdecl_ #1 _prop
1566   }
1567   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1568   \seq_if_empty:NTF \l_tmpa_seq {
1569     % TODO throw error
1570   } {
1571     \seq_if_in:NxTF \l_tmpa_seq
1572       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
1573       \use:c{
1574         stex_notation_ #1 \c_hash_str
1575         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1576         _cs
1577       }
1578     }{
1579       \str_if_empty:NTF \l__stex_notation_variant_str {
1580         \str_if_empty:NTF \l__stex_notation_lang_str {
1581           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
1582           \use:c{
1583             stex_notation_ #1 \c_hash_str \l_tmpa_str
1584             _cs
1585           }
1586         }{
1587           % TODO throw error
1588         }
1589       }{
1590         % TODO throw error
1591       }
1592     }
1593   }
1594 }
```

(*End definition for* __stex_notation_invoke_math:nw.)

Test 10

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

<div style="border:1px solid black; padding:10px;">

Module 4.6[Foo5]

$\langle a^b{}_c \rangle$ and $\langle a^b{}_c \rangle$

</div>

.

4.8 Terms

1595 ⟨@@=stex_term⟩

```
1596 \cs_new_protected:Nn \stex_term_oms:nnnn {
1597   % TODO: precedence/bracketing
1598   \stex_annotate:nnn{OMID}{#1\c_hash_str#2}{#4}
1599 }
```

(*End definition for* \stex_term_oms:nnnn*. This function is documented on page* *12.*)

```
1600 \cs_new_protected:Nn \stex_term_oma:nnnn {
1601   % TODO: precedence/bracketing
1602   \stex_annotate:nnn{OMA}{#1\c_hash_str#2}{#4}
1603 }
```

(*End definition for* \stex_term_oma:nnnn*. This function is documented on page* *12.*)

```
1604 \cs_new_protected:Nn \stex_term_omb:nnnn {
1605   % TODO: precedence/bracketing
1606   \stex_annotate:nnn{OMBIND}{#1\c_hash_str#2}{#4}
1607 }
```

(*End definition for* \stex_term_omb:nnnn*. This function is documented on page* *12.*)

```
1608 \cs_new_protected:Nn \stex_term_arg:nnn {
1609   % TODO: precedence/bracketing
1610   \stex_annotate:nnn{arg}{#1}{#3}
1611 }
```

(*End definition for* \stex_term_arg:nnn*. This function is documented on page* *12.*)

```
1612 \cs_new_protected:Nn \stex_term_assoc_arg:nnnn {
1613   \stex_term_arg:nnn{#1}{#2}{
1614     % TODO something with #3 applied to #4
1615   }
1616 }
```

(*End definition for* \stex_term_assoc_arg:nnnn*. This function is documented on page* *12.*)