`stex.sty`: sTeX 2.0[*]

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

August 17, 2021

Abstract

TODO

1 Introduction

TODO

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

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 - `\input`s are considered the same file).

3.5 Symbols

`\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.
- `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 `i`s, `a`s and `b`s),
- `arity` (integer),
- `assocs` (integer; 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_invoke_symbol:n

4 Implementation

4.1 The sTeX document class

```
⟨*cls⟩
\RequirePackage{expl3,l3keys2e}
\ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
\LoadClass[border=1px,varwidth]{standalone}
\setlength\textwidth{15cm}
\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}

\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
\ProcessOptions

\RequirePackage{stex}
⟨/cls⟩
```

4.2 Preliminaries

```
⟨*package⟩
\RequirePackage{expl3,l3keys2e}
\ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
```

Package options:

```
\keys_define:nn { stex } {
  debug     .bool_set:N   = \c_stex_debug_bool ,
  showmods  .bool_set:N   = \c_stex_showmods_bool ,
  lang      .clist_set:N  = \c_stex_languages_clist ,
  mathhub   .tl_set_x:N   = \mathhub ,
  sms       .bool_set:N   = \c_stex_persist_mode_bool
}
\ProcessKeysOptions { stex }
```

\sTeX The sTeX logo:

```
\protected\def\stex{%
  \@ifundefined{texorpdfstring}%
  {\let\texorpdfstring\@firstoftwo}%
  {}%
  \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace%
}
\def\sTeX{\stex}
```

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

Messages

```
\msg_new:nnn{stex}{debug}{}
\msg_new:nnn{stex}{warning/nomathhub}{
  MATHHUB~system~variable~not~found~and~no~
  \detokenize{\mathhub}-value~set!
}
\msg_new:nnn{stex}{error/norepository}{}
\msg_new:nnn{stex}{error/modulemissing}{}
```

\stex_debug:n Debug mode

```
\cs_new_protected:Nn \stex_debug:n {
  \bool_if:nT{\c_stex_debug_bool}{
    \exp_args:Nnnx\msg_set:nnn{stex}{debug}{\\Debug:~#1\\}
    \msg_term:nn{stex}{debug} % should be \msg_note:nn
  }
}

\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

```
\iow_new:N \c__stex_sms_iow
\AddToHook{begindocument}{
  \bool_if:NTF \c_stex_persist_mode_bool {
    \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
  } {
    \iow_open:Nn \c__stex_sms_iow {\jobname.sms}
  }
}
\AddToHook{enddocument}{
  \bool_if:NF \c_stex_persist_mode_bool {
    \iow_close:N \c__stex_sms_iow
  }
}
```

(*End definition for* \c__stex_sms_iow.)

\stex_addtosms:n

```
\cs_new_protected:Nn \stex_addtosms:n {
  \bool_if:NF \c_stex_persist_mode_bool {
    \iow_now:Nn \c__stex_sms_iow { #1 }
  }
}
```

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

4.2.1 LaTeXML and ScaLaTeX

```
\RequirePackage{scalatex}
```

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

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

\if@latexml
\latexml_if_p:
\latexml_if:*TF*

Conditionals for LaTeXML:

```
\ifcsname if@latexml\endcsname\else
    \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
\fi

\prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
  \if@latexml
    \prg_return_true:
  \else:
    \prg_return_false:
  \fi:
}
```

(*End definition for* `\if@latexml` *and* `\latexml_if:TF`*. These functions are documented on page 4.*)

4.2.2 HTML Annotations

```
⟨@@=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.

```
\tl_new:N \l__stex_annotate_arg_tl
\tl_const:Nx \c__stex_annotate_emptyarg_tl {
  \scalatex_if:TF {
    \scalatex_direct_HTML:n { \c_ampersand_str lrm; }
  }{~}
}
```

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

`\__stex_annotate_checkempty:n`

```
\cs_new_protected:Nn \__stex_annotate_checkempty:n {
  \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
  \tl_if_empty:NT \l__stex_annotate_arg_tl {
    \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
  }
}
```

(*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.

```
\scalatex_if:TF{
  \cs_new_protected:Nn \stex_annotate:nnn {
    \__stex_annotate_checkempty:n { #3 }
    \scalatex_annotate_HTML:nn {
      property="stex:#1" ~
      resource="#2"
    } {
      \tl_use:N \l__stex_annotate_arg_tl
    }
  }
  \cs_new_protected:Nn \stex_annotate_invisible:n {
    \__stex_annotate_checkempty:n { #1 }
    \scalatex_annotate_HTML:nn {
      stex:visible="false" ~
      style:display="none"
    } {
      \tl_use:N \l__stex_annotate_arg_tl
    }
  }
  \cs_new_protected:Nn \stex_annotate_invisible:nnn {
    \__stex_annotate_checkempty:n { #3 }
    \scalatex_annotate_HTML:nn {
      property="stex:#1" ~
      resource="#2" ~
      stex:visible="false" ~
      style:display="none"
    } {
      \tl_use:N \l__stex_annotate_arg_tl
    }
  }
  \NewDocumentEnvironment{stex_annotate_env} { m m } {
    \par
    \scalatex_annotate_HTML_begin:n {
      property="stex:#1" ~
      resource="#2"
    }
  }{
    \scalatex_annotate_HTML_end:
  }
}{
  \latexml_if:TF {
    \cs_new_protected:Nn \stex_annotate:nnn {
      \__stex_annotate_checkempty:n { #3 }
      \mode_if_math:TF {
        \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
          \tl_use:N \l__stex_annotate_arg_tl
        }
      }{
        \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
          \tl_use:N \l__stex_annotate_arg_tl
        }
      }
    }
    \cs_new_protected:Nn \stex_annotate_invisible:n {
      \__stex_annotate_checkempty:n { #1 }
      \mode_if_math:TF {
        \cs:w latexml@invisible@math\cs_end:{
          \tl_use:N \l__stex_annotate_arg_tl
        }
      } {
        \cs:w latexml@invisible@text\cs_end:{
          \tl_use:N \l__stex_annotate_arg_tl
        }
      }
    }
    \cs_new_protected:Nn \stex_annotate_invisible:nnn {
      \__stex_annotate_checkempty:n { #3 }
      \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
    \NewDocumentEnvironment{stex_annotate_env} { m m } {
      \par\begin{latexml@annotateenv}{#1}{#2}
    }{
      \end{latexml@annotateenv}
    }
  }{
    \cs_new_protected:Nn \stex_annotate:nnn {#3}
    \cs_new_protected:Nn \stex_annotate_invisible:n {}
    \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
    \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{}
  }
}
```

(*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

```
⟨@@=stex_language⟩
```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```
\prop_const_from_keyval:Nn \c_stex_languages_prop {
  en = english ,
  de = ngerman ,
  ar = arabic ,
  bg = bulgarian ,
  ru = russian ,
  fi = finnish ,
  ro = romanian ,
  tr = turkish ,
  fr = french
}

\prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
  english   = en ,
  ngerman   = de ,
  arabic    = ar ,
  bulgarian = bg ,
  russian   = ru ,
  finnish   = fi ,
  romanian  = ro ,
  turkish   = tr ,
  french    = fr
}
% todo: chinese simplified (zhs)
%       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:

```
\clist_if_empty:NF \c_stex_languages_clist {
  \clist_clear:N \l_tmpa_clist
  \clist_map_inline:Nn \c_stex_languages_clist {
    \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
      \clist_put_right:No \l_tmpa_clist \l_tmpa_str
    } {
      \msg_set:nnn{stex}{error/unknownlanguage}{
        Unknown~language~\l_tmpa_str
      }
      \msg_error:nn{stex}{error/unknownlanguage}
    }
  }
  \stex_debug:n {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
  \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
}
```

4.3 Files, Paths and URIs

```
⟨@@=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

```
%% TODO Windows paths
\cs_new_protected:Nn \stex_path_from_string:Nn {
  \exp_args:NNe\str_set:Nn \l_tmpa_tl { #2 }
  \tl_trim_spaces:N \l_tmpa_tl
  \str_if_empty:NTF \l_tmpa_tl {
    \seq_set_eq:NN #1 \c_empty_seq
  }{
    \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_tl }
    \stex_path_canonicalize:N #1
  }
}
\cs_generate_variant:Nn \stex_path_from_string:Nn
  { 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

```
\cs_new_protected:Nn \stex_path_to_string:NN {
  \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
}

\cs_new:Nn \stex_path_to_string:N {
  \seq_use:Nn #1 /
}
```

(*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.

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

```
\cs_new_protected:Nn \stex_path_canonicalize:N {
  \seq_if_empty:NF #1 {
    \seq_clear:N \l_tmpa_seq
    \seq_get_left:NN #1 \l_tmpa_tl
    \str_if_empty:NT \l_tmpa_tl {
      \seq_put_right:Nn \l_tmpa_seq {}
    }
    \seq_map_inline:Nn #1 {
      \str_set:Nn \l_tmpa_tl { ##1 }
      \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
        \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
          \seq_if_empty:NTF \l_tmpa_seq {
            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
              \c__stex_path_up_str
            }
          }{
            \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
            \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
              \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
                \c__stex_path_up_str
              }
            }{
              \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
            }
          }
        }{
          \str_if_empty:NF \l_tmpa_tl {
            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
          }
        }
      }
    }
    \seq_gset_eq:NN #1 \l_tmpa_seq
  }
}
```

(*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/../..		

.

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

```
\prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
  \seq_if_empty:NTF #1 {
    \prg_return_false:
  }{
    \seq_get_left:NN #1 \l_tmpa_tl
    \str_if_empty:NTF \l_tmpa_tl {
      \prg_return_true:
    }{
      \prg_return_false:
    }
  }
}
```

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

4.3.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```
\str_new:N\l_stex_kpsewhich_return_str
\cs_new_protected:Nn \stex_kpsewhich:n {
  \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
  \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
  \tl_trim_spaces:N \l_stex_kpsewhich_return_str
}
```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```
\sys_if_platform_windows:TF{
  \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
}{
  \stex_kpsewhich:n{-var-value~PWD}
}

\stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
\stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
\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

```
⟨@@=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.

`\g__stex_files_stack` keeps track of file changes

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

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

`\c_stex_mainfile_seq`

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

(*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`.

```
\seq_gclear_new:N\g_stex_currentfile_seq
\AddToHook{file/before}{
  \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
  \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
    \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
  }{
    \stex_path_from_string:Nn\g_stex_currentfile_seq{
      \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
    }
  }
  \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
  \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
}
\AddToHook{file/after}{
  \seq_if_empty:NF\g__stex_files_stack{
    \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
  }
  \seq_if_empty:NTF\g__stex_files_stack{
    \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
  }{
    \seq_get:NN\g__stex_files_stack\l_tmpa_seq
    \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
  }
}
```

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

4.4 MathHub Repositories

```
⟨@@=stex_mathhub⟩
```

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

```
\str_if_empty:NTF\mathhub{
  \stex_kpsewhich:n{-var-value~MATHHUB}
  \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str

  \str_if_empty:NTF\c_stex_mathhub_str{
    \msg_warning:nn{stex}{warning/nomathhub}
  }{
    \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
    \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
  }
}{
  \stex_path_from_string:Nn\c_stex_mathhub_seq\mathhub
  \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
  \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
}
```

(*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`

```
\cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
  \str_set:Nx \l_tmpa_str { #1 }
  \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
    \prop_new:c { c_stex_mathhub_#1_manifest_prop }
    \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
    \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
    \__stex_mathhub_find_manifest:N \l_tmpa_seq
    \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
      \msg_set:nnn{stex}{error/norepository}{
        No~archive~#1~found~in~
          \stex_path_to_string:N \c_stex_mathhub_str
      }
      \msg_error:nn{stex}{error/norepository}
    } {
      \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
    }
  }
}
```

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

\l__stex_mathhub_manifest_file_seq

```
\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`:

```
\cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
  \seq_set_eq:NN\l_tmpa_seq #1
  \bool_set_true:N\l_tmpa_bool
  \bool_while_do:Nn \l_tmpa_bool {
    \seq_if_empty:NTF \l_tmpa_seq {
      \bool_set_false:N\l_tmpa_bool
    }{
      \file_if_exist:nTF{
        \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
      }{
        \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
        \bool_set_false:N\l_tmpa_bool
      }{
        \file_if_exist:nTF{
          \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
        }{
          \seq_put_right:Nn\l_tmpa_seq{META-INF}
          \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
          \bool_set_false:N\l_tmpa_bool
        }{
          \file_if_exist:nTF{
            \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
          }{
            \seq_put_right:Nn\l_tmpa_seq{meta-inf}
            \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
            \bool_set_false:N\l_tmpa_bool
          }{
            \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
          }
        }
      }
    }
  }
  \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
}
```

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

\c__stex_mathhub_manifest_ior

File variable used for `MANIFEST`-files

```
\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:

```
\cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
  \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
  \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
  \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
    \str_set:Nn \l_tmpa_str {##1}
    \exp_args:NNoo \seq_set_split:Nnn
        \l_tmpb_seq \c_colon_str \l_tmpa_str
    \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
      \exp_args:NNe \str_set:Nn \l_tmpb_tl {
        \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
      }
      \exp_args:No \str_case:nnTF \l_tmpa_tl {
        {id} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { id } \l_tmpb_tl
        }
        {narration-base} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { narr } \l_tmpb_tl
        }
        {source-base} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { ns } \l_tmpb_tl
        }
        {ns} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { ns } \l_tmpb_tl
        }
        {dependencies} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { deps } \l_tmpb_tl
        }
      }{}{}
    }{}
  }
  \ior_close:N \c__stex_mathhub_manifest_ior
}
```

(*End definition for* __stex_mathhub_parse_manifest:n.)

\stex_set_current_repository:n

```
\cs_new_protected:Nn \stex_set_current_repository:n {
  \stex_require_repository:n { #1 }
  \prop_set_eq:Nc \l_stex_current_repository_prop {
    c_stex_mathhub_#1_manifest_prop
  }
}
```

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

\stex_require_repository:n

```
\cs_new_protected:Nn \stex_require_repository:n {
  \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
    \stex_debug:n{Opening~archive:~#1}
    \__stex_mathhub_do_manifest:n { #1 }
    \exp_args:Nx \stex_addtosms:n {
      \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
        id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } {  id  } ,
        ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } {  ns  } ,
        narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
        deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
      }
    }
  }
}
```

(*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.

```
\prop_new:N \l_stex_current_repository_prop
\AddToHook{begindocument}{
  \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
  \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
    \stex_debug:n{Not~currently~in~a~MathHub~repository}
  } {
    \__stex_mathhub_parse_manifest:n { main }
    \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
      \l_tmpa_str
    \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
    \stex_set_current_repository:n { main }
    \stex_debug:n{Current~repository:~
      \prop_item:Nn \l_stex_current_repository_map {id}
    }
  }
}
```

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

4.5 Module System

```
⟨@@=stex_module⟩
```

\l_stex_current_module_prop

```
\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*

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

(*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*

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

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

\stex_add_to_current_module:n

```
\cs_new_protected:Nn \stex_add_to_current_module:n {
  \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
  \tl_put_right:Nn \l_tmpa_tl { #1 }
  \prop_put:Nno \l_stex_current_module_prop { content } \l_tmpa_tl
}
```

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

\stex_add_constant_to_current_module:n

```
\cs_new_protected:Nn \stex_add_constant_to_current_module:n {
  \str_set:Nx \l_tmpa_str { #1 }
  \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
  \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
  \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
}
```

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

\stex_add_import_to_current_module:n

```
\cs_new_protected:Nn \stex_add_import_to_current_module:n {
  \str_set:Nx \l_tmpa_str { #1 }
  \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
  \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
  \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
}
```

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

\stex_modules_compute_namespace:nN stores its return values in:

\l_stex_modules_ns_str

```
\str_new:N \l_stex_modules_ns_str
```

```
\cs_new_protected:Nn \stex_modules_compute_namespace:nN {
  \str_set:Nx \l_tmpa_str { #1 }
  \seq_set_eq:NN \l_tmpa_seq #2
  % split off file extension
  \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
  \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
  \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
  \seq_put_right:No \l_tmpa_seq \l_tmpb_str

  \bool_set_true:N \l_tmpa_bool
  \bool_while_do:Nn \l_tmpa_bool {
    \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
    \exp_args:No \str_case:nnTF { \l_tmpb_str } {
      {source} { \bool_set_false:N \l_tmpa_bool }
    }{}{
      \seq_if_empty:NT \l_tmpa_seq {
        \bool_set_false:N \l_tmpa_bool
      }
    }
  }

  \seq_if_empty:NTF \l_tmpa_seq {
    \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
  }{
    \str_set:Nx \l_stex_modules_ns_str {
      \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
    }
  }
}
```

(*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:

```
\cs_new_protected:Nn \stex_modules_current_namespace: {
  \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
    \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
  }{
    % split off file extension
    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
    \str_set:Nx \l_stex_modules_ns_str {
      file:/\stex_path_to_string:N \l_tmpa_seq
    }
  }
}
```

(*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:

```
\keys_define:nn { stex / module } {
  title .tl_set_x:N  = \l_stex_module_title_str ,
  ns    .tl_set_x:N  = \l_stex_module_ns_str ,
  lang  .tl_set_x:N  = \l_stex_module_lang_str ,
  sig   .tl_set_x:N  = \l_stex_module_sig_str ,
  meta  .tl_set_x:N  = \l_stex_module_meta_str
}

% module parameters here? In the body?

\cs_new_protected:Nn \__stex_module_args:n {
  \str_clear:N \l_stex_module_title_str
  \str_clear:N \l_stex_module_ns_str
  \str_clear:N \l_stex_module_lang_str
  \str_clear:N \l_stex_module_sig_str
  \str_clear:N \l_stex_module_meta_str
  \keys_set:nn { stex / module } { #1 }
  \exp_args:NNo \str_set:Nn \l_stex_module_title_str
    \l_stex_module_title_str
  \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
    \l_stex_module_ns_str
  \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
    \l_stex_module_lang_str
  \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
    \l_stex_module_sig_str
  \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
    \l_stex_module_meta_str
}
```

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

```
\cs_new_protected:Nn \__stex_module_begin_module: {
  % Nested module?
  \stex_if_in_module:TF {
    % Nested module
    \prop_get:NnN \l_stex_current_module_prop
      { ns } \l_stex_module_ns_str
    \str_set:Nx \l_stex_module_name_str {
      \prop_item:Nn \l_stex_current_module_prop
        { name } / \l_stex_module_name_str
    }
  }{
    % not nested:
    \str_if_empty:NT \l_stex_module_ns_str {
      \stex_modules_current_namespace:
      \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
      \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
         / {\l_stex_module_ns_str}
      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
      \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
        \str_set:Nx \l_stex_module_ns_str {
          \stex_path_to_string:N \l_tmpa_seq
        }
      }
    }
  }

  % language
  \str_if_empty:NF \l_stex_module_lang_str {
    \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
      \l_tmpa_str {
        \exp_args:Nx \selectlanguage { \l_tmpa_str }
      } {
        \msg_set:nnn{stex}{error/unknownlanguage}{
          Unknown~language~\l_tmpa_str
        }
        \msg_error:nn{stex}{error/unknownlanguage}
      }
  }

  % signature
  \str_if_empty:NF \l_stex_module_sig_str {
    \str_if_empty:NT \l_stex_module_lang_str {
      \msg_set:nnn{stex}{error/siglanguage}{
        Module~\l_stex_module_ns_str?\l_stex_module_name_str~
        declares~signature~\l_stex_module_sig_str,~but~does~not~
        declare~its~language
      }
      \msg_error:nn{stex}{error/siglanguage}
    }
  }

  % metatheory
%  \str_if_empty:NTF \l_stex_module_meta_str {
%
%  } {
%
%  }

  \str_clear:N \l_tmpa_str
  \seq_clear:N \l_tmpa_seq
  \tl_clear:N \l_tmpa_tl
  \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
    name      = \l_stex_module_name_str ,
    ns        = \l_stex_module_ns_str ,
    import    = \exp_not:o { \l_tmpa_seq } ,
    constants = \exp_not:o { \l_tmpa_seq } ,
    content   = \exp_not:o { \l_tmpa_seq }  ,
    file      = \exp_not:o { \g_stex_currentfile_seq } ,
    lang      = \l_stex_module_lang_str ,
    sig       = \l_stex_module_sig_str ,
    meta      = \l_stex_module_meta_str
  }

  \stex_debug:n{
    New~module:\\
    Namespace:~\l_stex_module_ns_str\\
    Name:~\l_stex_module_name_str\\
    Language:~\l_stex_module_lang_str\\
    Signature:~\l_stex_module_sig_str\\
    Metatheory:~\l_stex_module_meta_str\\
    File:~\stex_path_to_string:N \g_stex_currentfile_seq
  }

  \seq_gput_right:Nx  \g_stex_modules_in_file_seq
      { \l_stex_module_ns_str ? \l_stex_module_name_str }

  \stex_if_smsmode:TF {
    \stex_smsmode_set_codes:
  } {
    \begin{stex_annotate_env} {theory} {
      \l_stex_module_ns_str ? \l_stex_module_name_str
    }

    \stex_annotate_invisible:nnn{header}{} {
      \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
      \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
      \str_if_empty:NT \l_stex_module_meta_str {
        % TODO metatheory
      }
    }
  }
}
\iffalse \end{stex_annotate_env} \fi % make syntax highlighting work again
```

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

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

```
\iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
```

```
\cs_new_protected:Nn \__stex_module_end_module: {
  \str_set:Nx \l_tmpa_str {
    c_stex_module_
    \prop_item:Nn \l_stex_current_module_prop { ns } ?
    \prop_item:Nn \l_stex_current_module_prop { name }
    _prop
  }
  \prop_new:c { \l_tmpa_str }
  \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
  \stex_if_smsmode:TF {
    \exp_args:Nx \stex_addtosms:n {
      \prop_gset_from_keyval:cn {
        c_stex_module_
        \prop_item:Nn \l_stex_current_module_prop { ns } ?
        \prop_item:Nn \l_stex_current_module_prop { name }
        _prop
      } {
        name      = \prop_item:cn { \l_tmpa_str } { name } ,
        ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
        import    = \prop_item:cn { \l_tmpa_str } { import } ,
        constants = \prop_item:cn { \l_tmpa_str } { constants } ,
        content   = \prop_item:cn { \l_tmpa_str } { content } ,
        file      = \prop_item:cn { \l_tmpa_str } { file } ,
        lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
        sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
        meta      = \prop_item:cn { \l_tmpa_str } { meta }
      }
    }
  }{
    \end{stex_annotate_env}
  }
}
```

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

`@module` The core environment, with no header

```
\NewDocumentEnvironment { @module } { O{} m } {
  \str_set:Nx \l_stex_module_name_str { #2 }
  \par
  \__stex_module_args:n { #1 }
  \__stex_module_begin_module:
} {
  \__stex_module_end_module:
}
```

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

```
\cs_if_exist:NTF \thesection {
  \newcounter{module}[section]
}{
  \newcounter{module}
}

\bool_if:NT \c_stex_showmods_bool {
  \latexml_if:F { \RequirePackage{mdframed} }
}

\cs_new_protected:Nn \stex_modules_heading: {
  \stepcounter{module}
  \par
  \bool_if:NT \c_stex_showmods_bool {
    \noindent{\textbf{Module} ~
      \cs_if_exist:NT \thesection {\thesection.}
      \themodule ~ [\l_stex_module_name_str]
    }
    % TODO references
    % \sref@label@id{Module \thesection.\themodule [\module@name]}%
    \str_if_empty:NTF \l_stex_module_title_str {
    }{
      \quad(\l_stex_module_title_str)\hfill
    }
  }
}
```

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

Finally:

```
\NewDocumentEnvironment { module } { O{} m } {
  \begin{@module}[#1]{#2}
```

```
    \stex_modules_heading:
    \bool_if:NT \c_stex_showmods_bool {
      \begin{mdframed}
    }
}{
  \bool_if:NT \c_stex_showmods_bool {
    \end{mdframed}
  }
  \end{@module}
}
```

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

```
⟨@@=stex_smsmode⟩
```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```
\tl_new:N \g_stex_smsmode_allowedmacros_tl
\tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
\seq_new:N \g_stex_smsmode_allowedenvs_seq

\tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
  \makeatletter
  \makeatother
  \ExplSyntaxOn
  \ExplSyntaxOff
}

\tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
  \symdef
%  \abbrdef
%  \module@export
  \importmodule
%  \mmt@symdecl
%  \instantiates
%  \setnotation
%  \importmhmodule
%  \gimport
%  \symvariant
%  \structural@feature
%  \symi
%  \symii
%  \symiii
%  \symiv
  \notation
  \symdecl
%  \defi
%  \defii
%  \defiii
%  \defiv
%  \adefi
%  \adefii
%  \adefiii
%  \adefiv
%  \defis
%  \defiis
%  \defiiis
%  \defivs
%  \Defi
%  \Defii
%  \Defiii
%  \Defiv
%  \Defis
%  \Defiis
%  \Defiiis
%  \Defivs
}

\exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
  \tl_to_str:n {
    module,
    @module
%   modsig,
%   mhmodsig,
%   mhmodnl,
%   modnl,
%   @structural@feature
  }
}
```

(*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`

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

(*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.

```
\bool_new:N \g__stex_smsmode_catcode_bool
\bool_set_false:N \g__stex_smsmode_catcode_bool
\prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
  \bool_if:NTF \g__stex_smsmode_catcode_bool
    \prg_return_true: \prg_return_false:
}
```

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

`\stex_smsmode_set_codes:`

```
\cs_new_protected:Nn \stex_smsmode_set_codes: {
  \stex_if_smsmode:T {
    \__stex_smsmode_if_catcodes:F {
      \bool_gset_true:N \g__stex_smsmode_catcode_bool
      \exp_after:wN \char_gset_active_eq:NN
        \c_backslash_str \__stex_smsmode_cs:
      \tex_global:D \char_set_catcode_active:N \\
      \tex_global:D \char_set_catcode_other:N $
      \tex_global:D \char_set_catcode_other:N ^
      \tex_global:D \char_set_catcode_other:N _
      \tex_global:D \char_set_catcode_other:N &
      \tex_global:D \char_set_catcode_other:N ##
    }
  }
} \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.

```
\cs_new_protected:Nn \__stex_smsmode_unset_codes: {
  \__stex_smsmode_if_catcodes:T {
    \bool_gset_false:N \g__stex_smsmode_catcode_bool
    \exp_after:wN \tex_global:D \exp_after:wN
      \char_set_catcode_escape:N \c_backslash_str
    \tex_global:D \char_set_catcode_math_toggle:N $
    \tex_global:D \char_set_catcode_math_superscript:N ^
    \tex_global:D \char_set_catcode_math_subscript:N _
    \tex_global:D \char_set_catcode_alignment:N &
    \tex_global:D \char_set_catcode_parameter:N ##
  }
} \iffalse $ \fi % to make syntax highlighting work again
```

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

`\stex_in_smsmode:nn`

```
\cs_new_protected:Nn \stex_in_smsmode:nn {
  \vbox_set:Nn \l_tmpa_box {
    \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
    \bool_gset_true:N \g__stex_smsmode_bool
    \stex_smsmode_set_codes:
    #2
    \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
    \stex_if_smsmode:F {
      \__stex_smsmode_unset_codes:
    }
  }
  \box_clear:N \l_tmpa_box
}
```

(*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:

```
\str_const:Nn \c__stex_smsmode_begin_str { begin }
\str_const:Nn \c__stex_smsmode_end_str { end }

\cs_new_protected:Nn \__stex_smsmode_cs: {
  \str_clear:N \l_tmpa_str
  \peek_analysis_map_inline:n {
    % #1: token (one expansion)
    % #2: charcode
    % #3 catcode
    \token_if_eq_charcode:NNTF ##3 B {
      % token is a letter
      \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
    } {
      \str_if_empty:NTF \l_tmpa_str {
        % we don't allow (or need) single non-letter CSs
        % for now
        \peek_analysis_map_break:
      }{
        \str_if_eq:nnTF \l_tmpa_str \c_stex_begin_str {
          \peek_analysis_map_break:n {
            \exp_after:wN \__stex_smsmode_checkbegin:n ##1
          }
        } {
          \str_if_eq:nnTF \l_tmpa_str \c_stex_end_str {
            \peek_analysis_map_break:n {
              \exp_after:wN \__stex_smsmode_checkend:n ##1
            }
          } {
          \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
          \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
            \g_stex_smsmode_allowedmacros_tl
              { \use:c{\l_tmpa_str} } {
              \peek_analysis_map_break:n {
                \exp_after:wN \l_tmpa_tl ##1
              }
            } {
              \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
              \g_stex_smsmode_allowedmacros_escape_tl
                { \use:c{\l_tmpa_str} } {
                \exp_args:NNNo \exp_args:No
                \token_if_eq_charcode_p:NNTF \c_backslash_str ##1 {
                  \peek_analysis_map_break:n {
                    \__stex_smsmode_unset_codes:
                    \__stex_smsmode_rescan_cs:
                  }
                } {
                  \peek_analysis_map_break:n {
                    \__stex_smsmode_unset_codes:
                    \exp_after:wN \l_tmpa_tl ##1
                  }
                }
              } {
                \peek_analysis_map_break:n { ##1 }
              }
            }
          }
        }
      }
    }
  }
}
```

(*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:

```
\cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
  \str_clear:N \l_tmpb_str
  \peek_analysis_map_inline:n {
    \token_if_eq_charcode:NNTF ##3 B {
      % token is a letter
      \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
    } {
      \peek_analysis_map_break:n {
        \exp_after:wN \use:c \exp_after:wN {
          \exp_after:wN \l_tmpa_str\exp_after:wN
        } \use:c { \l_tmpb_str \exp_after:wN } ##1
      }
    }
  }
}
```

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

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

```
\cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
  \str_set:Nn \l_tmpa_str { #1 }
  \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
    \__stex_smsmode_unset_codes:
```

```
    \begin{#1}
  }
}
```

(*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.

```
\cs_new_protected:Nn \__stex_smsmode_checkend:n {
  \str_set:Nn \l_tmpa_str { #1 }
  \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
    \end{#1}
  }
}
```

(*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

```
⟨@@=stex_importmodule⟩
```

`\stex_import_module_uri:nn`

```
\cs_new_protected:Nn \stex_import_module_uri:nn {
  \str_set:Nx \l__stex_importmodule_archive_str { #1 }
  \str_set:Nx \l__stex_importmodule_path_str { #2 }
  \str_if_empty:NT \l__stex_importmodule_archive_str {
    \prop_if_empty:NF \l_stex_current_repository_prop {
      \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
    }
  }

  \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_tmpb_str }
  \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
  \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpa_seq ? }

  \str_if_empty:NTF \l_tmpa_str {
    \stex_modules_current_namespace:
    \str_if_empty:NTF \l__stex_importmodule_path_str {
      \str_set:Nx \l_stex_module_ns_str {
        \l_stex_module_ns_str ? \l__stex_importmodule_name_str
      }
    }{
      \str_set:Nx \l_stex_module_ns_str {
        \l_stex_module_ns_str / \l__stex_importmodule_path_str ? \l__stex_importmodule_name_
      }
    }
  }{
    \stex_require_repository:n \l__stex_importmodule_archive_str
    \prop_get:cnN { c_stex_mathhub_\l__stex_importmodule_archive_str _manifest_prop } { ns }
      \l_stex_module_ns_str
    \str_if_empty:NTF \l__stex_importmodule_path_str {
      \str_set:Nx \l__stex_importmodule_module_ns_str {
        \l_stex_module_ns_str ? \l__stex_importmodule_name_str
      }
    }{
      \str_set:Nx \l__stex_importmodule_module_ns_str {
        \l_stex_module_ns_str / \l__stex_importmodule_path_str ? \l__stex_importmodule_name_
      }
    }
  }
}
```

(*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`.

```
\str_new:N \l__stex_importmodule_name_str
\str_new:N \l__stex_importmodule_archive_str
\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*⟩}

```
\cs_new_protected:Nn \stex_import_require_module:nnnn {
  \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
    % archive
    \str_set:Nx \l_tmpa_str { #2 }
    \str_if_empty:NTF \l_tmpa_str {
      \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
    } {
      \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
      \exp_args:NNo \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
      \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
      \seq_put_right:Nn \l_tmpa_seq { source }
    }

    % path
    \str_set:Nx \l_tmpb_str { #3 }
    \str_if_empty:NT \l_tmpb_str {
      \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }

      \cs_if_exist:NTF \languagename {
        \prop_get:NnN \c_stex_language_abbrevs_prop
            { \languagename } \l_tmpb_str
      }

      \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
        \str_set:Nx \l_tmpa_str { \l_tmpa_str.\l_tmpb_str.tex }
      }{
        \IfFileExists{ \l_tmpa_str.tex }{
          \str_set:Nx \l_tmpa_str { \l_tmpa_str.tex }
        }{
          % try english as default
          \IfFileExists{ \l_tmpa_str.en.tex }{
            \str_set:Nx \l_tmpa_str { \l_tmpa_str.en.tex }
          }{
            \msg_new:nnn{stex}{error/modulemissing}{
              No~file~for~module~#1?#4~found
            }
            \msg_error:nn{stex}{error/modulemissing}
          }
        }
      }

    } {
      \exp_args:NNo \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpb_str }
      \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq

      \cs_if_exist:NTF \languagename {
        \prop_get:NnN \c_stex_language_abbrevs_prop
            { \languagename } \l_tmpb_str
      }

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

      \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
        \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
      }{
        \IfFileExists{ \l_tmpa_str/#4.tex }{
          \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.tex }
        }{
          % try english as default
          \IfFileExists{ \l_tmpa_str/#4.en.tex }{
            \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.en.tex }
          }{
            \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
              \str_set:Nx \l_tmpa_str { \l_tmpa_str.\l_tmpb_str.tex }
            }{
              \IfFileExists{ \l_tmpa_str.tex }{
                \str_set:Nx \l_tmpa_str { \l_tmpa_str.tex }
              }{
                % try english as default
                \IfFileExists{ \l_tmpa_str.en.tex }{
                  \str_set:Nx \l_tmpa_str { \l_tmpa_str.en.tex }
                }{
                  \msg_new:nnn{stex}{error/modulemissing}{
                    No~file~for~module~#1?#4~found
                  }
                  \msg_error:nn{stex}{error/modulemissing}
                }
              }
            }
          }
        }
      }
    }

    \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
    \seq_clear:N \g_stex_modules_in_file_seq
    \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
      \str_set:Nx \l_tmpb_str { #2 }
      \str_if_empty:NF \l_tmpb_str {
        \stex_set_current_repository:n { #2 }
      }
      \input { \l_tmpa_str }
    }
    \prop_gput:Noo \g_stex_module_files_prop
      \l_tmpa_str \g_stex_modules_in_file_seq
    \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq

    \stex_if_module_exists:nF { #1 ? #4 } {
      \msg_new:nnn{stex}{error/modulemissing}{
        Module~#1?#4~not~found~in~file~\l_tmpa_str
      }
      \msg_error:nn{stex}{error/modulemissing}
    }
    % TODO write to sms file
  }
  % activate
  \prop_item:cn { c_stex_module_#1?#4_prop } { content }
}
```

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

\importmodule

```
\NewDocumentCommand \importmodule { O{} m } {
  \stex_import_module_uri:nn { #1 } { #2 }
  \stex_if_smsmode:F {
    \stex_import_require_module:nnnn
    { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
    { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
    \stex_annotate_invisible:nnn
      {import} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
  }
  \exp_args:Nx \stex_add_to_current_module:n {
    \stex_import_require_module:nnnn
    { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
    { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
  }
  \exp_args:Nx \stex_add_import_to_current_module:n {
    \l_stex_module_ns_str ? \l__stex_importmodule_name_str
  }
  \stex_smsmode_set_codes:
}
```

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

`\usemodule`

```
\NewDocumentCommand \usemodule { O{} m } {
  \stex_if_smsmode:F {
    \stex_import_module_uri:nn { #1 } { #2 }
    \stex_import_require_module:nnnn
    { \l__stex_importmodule_module_ns_str } { \l__stex_importmodule_archive_str }
    { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
    \stex_annotate_invisible:nnn
      {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
  }
  \stex_smsmode_set_codes:
}
```

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

`\g_stex_modules_in_file_seq`
`\g_stex_module_files_prop`

```
\seq_new:N \g_stex_modules_in_file_seq
\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

```
⟨@@=stex_symdecl⟩
```

`symdecl` arguments:

```
\keys_define:nn { stex / symdecl } {
  name   .tl_set_x:N  = \l_stex_symdecl_name_str ,
  local  .bool_set:N  = \l_stex_symdecl_local_bool ,
  args   .tl_set_x:N  = \l_stex_symdecl_args_str ,
  type   .tl_set:N    = \l_stex_symdecl_type_tl
}

\cs_new_protected:Nn \__stex_symdecl_args:n {
  \str_clear:N \l_stex_symdecl_name_str
  \str_clear:N \l_stex_symdecl_args_str
  \bool_set_false:N \l_stex_symdecl_local_bool
  \tl_clear:N \l_stex_symdecl_type_tl

  \keys_set:nn { stex /symdecl } { #1 }

  \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
    \l_stex_symdecl_name_str
  \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
    \l_stex_symdecl_args_str
}
```

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

```
\NewDocumentCommand \symdecl { O{} m } {
  \__stex_symdecl_args:n { #1 }
  \tl_clear:N \l_stex_symdecl_definiens_tl
  \stex_symdecl_do:n { #2 }
}
```

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

`\stex_symdecl_do:n`

```
\cs_new_protected:Nn \stex_symdecl_do:n {
  \stex_if_in_module:F {
    % TODO throw error? some default namespace?
  }

  \str_if_empty:NT \l_stex_symdecl_name_str {
    \str_set:Nx \l_stex_symdecl_name_str { #1 }
  }

  \prop_if_exist:cT { g_stex_symdecl_
    \prop_item:Nn \l_stex_current_module_prop {ns} ?
    \prop_item:Nn \l_stex_current_module_prop {name} ?
      \l_stex_symdecl_name_str
    _prop
  }{
    % TODO throw error (beware of circular dependencies)
  }

  \prop_clear:N \l_tmpa_prop
  \prop_put:Nnx \l_tmpa_prop { module } {
    \prop_item:Nn \l_stex_current_module_prop {ns} ?
    \prop_item:Nn \l_stex_current_module_prop {name}
  }
  \seq_clear:N \l_tmpa_seq
  \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
  \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
  \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
  \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl

  \exp_args:No \stex_add_constant_to_current_module:n {
    \l_stex_symdecl_name_str
  }

  % arity/args
  \int_zero:N \l_tmpb_int

  \bool_set_true:N \l_tmpa_bool
  \str_map_inline:Nn \l_stex_symdecl_args_str {
    \token_case_meaning:NnF ##1 {
      0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
      {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
      {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
      {\tl_to_str:n a} {
        \bool_set_false:N \l_tmpa_bool
        \int_incr:N \l_tmpb_int
      }
    }{
      \msg_set:nnn{stex}{error/wrongargs}{
```

```
      args~value~in~symbol~declaration~for~
      \prop_item:Nn \l_stex_current_module_prop {ns} ?
      \prop_item:Nn \l_stex_current_module_prop {name} ?
      \l_stex_symdecl_name_str ~
      needs~to~be~
      i,~a~or~b,~but~##1~given
    }
    \msg_error:nn{stex}{error/wrongargs}
  }
}
\bool_if:NTF \l_tmpa_bool {
  % possibly numeric
  \str_if_empty:NTF \l_stex_symdecl_args_str {
    \prop_put:Nnn \l_tmpa_prop { args } {}
    \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
  }{
    \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
    \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
    \str_clear:N \l_tmpa_str
    \int_step_inline:nn \l_tmpa_int {
      \str_put_right:Nn \l_tmpa_str i
    }
    \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
  }
} {
  \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
  \prop_put:Nnx \l_tmpa_prop { arity }
    { \str_count:N \l_stex_symdecl_args_str }
}
\prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }


% semantic macro

\tl_set:cx { #1 } { \stex_invoke_symbol:n {
  \prop_item:Nn \l_tmpa_prop { module } ?
    \prop_item:Nn \l_tmpa_prop { name }
} }

\bool_if:NF \l_stex_symdecl_local_bool {
  \exp_args:Nx \stex_add_to_current_module:n {
    \tl_set:cx { #1 } { \stex_invoke_symbol:n {
      \prop_item:Nn \l_tmpa_prop { module } ?
        \prop_item:Nn \l_tmpa_prop { name }
    } }
  }
}


\stex_debug:n{New~symbol:~
  \prop_item:Nn \l_tmpa_prop { module } ?
    \prop_item:Nn \l_tmpa_prop { name }^^J
  Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
  Args:~\prop_item:Nn \l_tmpa_prop { args }
  }

  \prop_gset_eq:cN {
    g_stex_symdecl_
    \prop_item:Nn \l_tmpa_prop { module } ?
    \prop_item:Nn \l_tmpa_prop { name }
    _prop
  } \l_tmpa_prop

  \stex_if_smsmode:TF {
    \bool_if:NF \l_stex_symdecl_local_bool {
      \exp_args:Nx \stex_addtosms:n {
        \prop_gset_from_keyval:cn {
          g_stex_symdecl_
          \prop_item:Nn \l_tmpa_prop { module } ?
          \prop_item:Nn \l_tmpa_prop { name }
          _prop
        } {
          name      = \prop_item:Nn \l_tmpa_prop { name }
          module    = \prop_item:Nn \l_tmpa_prop { module }
          notations = \prop_item:Nn \l_tmpa_prop { notations }
          local     = \prop_item:Nn \l_tmpa_prop { local }
          type      = \prop_item:Nn \l_tmpa_prop { type }
          args      = \prop_item:Nn \l_tmpa_prop { args }
          arity     = \prop_item:Nn \l_tmpa_prop { arity }
          assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
        }
      }
    }
    \stex_smsmode_set_codes:
  }{
    \stex_annotate_invisible:nnn {symdecl} {
      \prop_item:Nn \l_tmpa_prop { module } ?
      \prop_item:Nn \l_tmpa_prop { name }
    } {
      \stex_annotate_invisible{type}{}{$\l_stex_symdecl_type_tl$}
      \stex_annotate_invisible{args}{}{
        \prop_item:Nn \l_tmpa_prop { args }
      }
      \stex_annotate_invisible{macroname}{}{#1}
      \str_if_empty:NF \l_stex_symdecl_definiens_tl {
        \stex_annotate_invisible{definiens}{}
          {$\l_stex_symdecl_definiens_tl$}
      }
    }
  }
}
```

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

`\stex_get_symbol:n`

```
\str_new:N \l_stex_get_symbol_uri_str

\cs_new_protected:Nn \stex_get_symbol:n {
  \tl_if_head_eq_catcode:nNTF { #1 } \relax {
    % argument is a command
    % TODO
  }{
    % argument is a string
    % is it a command name?
    \tl_set:Nx \l_tmpa_tl { \use:c { #1 } }

    \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
      \stex_invoke_symbol:n {
      \exp_args:NNx \tl_set:Nn \l_tmpa_tl
        { \tl_tail:N \l_tmpa_tl }
      \tl_if_single:NTF \l_tmpa_tl {
        \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
          \exp_after:wN \str_set:Nn \exp_after:wN
            \l_stex_get_symbol_uri_str \l_tmpa_tl
        }{
          % TODO
          % tail is not a single group
        }
      }{
        % TODO
        % tail is not a single group
      }
    }{
      % TODO
      % head is not \stex_invoke_symbol:n
    }
  }
}
```

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

Test 7

```
 \begin{module}{Foo1}
\symdecl[name=foobar, args=3]{bar}
\symdecl[name=foobar2, args=iab]{bari}
^^A \symdecl[name=foobar3, args=xxx]{barii}
\ExplSyntaxOn
Meaning:~\meaning\bar\\
\stex_get_symbol:n { bar }
Result:~\l_stex_get_symbol_uri_str
^^A TODO: more tests
\ExplSyntaxOff
\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}
Result: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo1?foobar

.

4.7 Notations

```
⟨@@=stex_notation⟩
```

`notation` arguments:

```
\keys_define:nn { stex / notation } {
  lang    .tl_set_x:N = \l__stex_notation_lang_str ,
  variant .tl_set_x:N = \l__stex_notation_variant_str ,
  prec    .tl_set_x:N = \l__stex_notation_prec_str ,
  unknown .code:n     = \str_set:Nx
      \l__stex_notation_variant_str \l_keys_key_str
}

\cs_new_protected:Nn \__stex_notation_args:n {
  \str_clear:N \l__stex_notation_lang_str
  \str_clear:N \l__stex_notation_variant_str
  \str_clear:N \l__stex_notation_prec_str

  \keys_set:nn { stex / notation } { #1 }

  \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
    \l__stex_notation_lang_str
  \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
    \l__stex_notation_variant_str
  \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
    \l__stex_notation_prec_str
}
```

`\notation`

```
\NewDocumentCommand \notation { O{} m } {
  \__stex_notation_args:n { #1 }
  \tl_clear:N \l_stex_symdecl_definiens_tl
  \stex_get_symbol:n { #2 }
  \stex_notation_do:n { \l_stex_get_symbol_uri_str }
}
```

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

`\stex_notation_do:n`

```
\cs_new_protected:Nn \stex_notation_do:n {
  \prop_gset_eq:Nc \l_tmpa_prop {
    g_stex_symdecl_ #1 _prop
  }

  % precedences
  % \notation[prec=500;50x49x51]{foo}{#1 bla #2 bla #3}{arg1}{arg3}





}
```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```
\cs_new_protected:Nn \stex_invoke_symbol:n {
  % TODO
}
```

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