stex.sty: sTeX 2.0[*]

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

2021-08-31

Abstract

TODO

1 Introduction

TODO

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

Contents

2 Manual

2.1 Modules

`{module}`, `{@module}`

2.2 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 1

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

ab

.

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

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

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

Example 2

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

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

.

EdN:1 Not using an explicit option with a semantic macro yields the first declared notation, unless changed[1].

[1]EdNote: TODO

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

Example 3

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

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

.

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

Example 4

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

Multiplying again by $b$ yields...

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

Example 5

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

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

.

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

For a macro with arity > 0, we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point `!` in either text or math mode.

Example 6

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

Multiplication (denoted by $\cdot$) is defined by...

.

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

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

2.2.1 Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have "normal" (or more precisely: `i`-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two `i`-type arguments.

Besides `i`-type arguments, sTeX has two other types, which we will discuss now.

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

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

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

Example 7

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

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

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

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

Example 8

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

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

EdN:2 EdN:3 EdN:4 ·[2] [3] [4]

2.2.2 Precedences

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

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

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

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

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

Consequently, if some operator A should bind stronger than some operator B, then As operator precedence should be larger than Bs argument precedences.

For example:

Example 9

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

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

.

2.3 Archives and Imports

2.3.1 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.

[2]EdNote: what about e.g. \int _x\int _y\int _z f dx dy dz?
[3]EdNote: "decompose" a-type arguments into fixed-arity operators?
[4]EdNote: flexary b-type arguments (e.g. for forall)?

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.

2.3.2 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.
- 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.

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

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:N`*TF* ⋆

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.)

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/../..		

.

3.3 MathHub Archives

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

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

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

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

`\l_stex_current_repository_prop`

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

`\stex_set_current_repository:n`

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

`\stex_require_repository:n`

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

Test 2

```
 \ExplSyntaxOn
\stex_require_repository:n { Foo/Bar }
id:~\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {id}\ \\
narr:~\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {narr}\ \\
ns:~\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {ns}\ \\
deps:~\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {deps}\ \\
\stex_require_repository:n { Bar/Foo }
\ExplSyntaxOff
```

```
id: Foo/Bar
narr:
ns: http://mathhub.info/tests/Foo/Bar
deps:
```

.

3.4 The Module System

`\l_stex_current_module_prop`

All information of a module is stored as a property list. `\l_stex_current_module_prop` always points to the current module (if existent).

Most importantly, the `content`-field stores all the code to execute on activation; i.e. when this module is being included.

Additionally, it stores:

- The *name* in field `name`,
- the *namespace* in field `ns`,
- this module's *language* in field `lang`,
- if a language module that translates some other modules, the *original* module in field `sig` (for signature),
- the *metatheory* in field `meta`,
- the URIs of all *imported modules* in field `imports`,
- the names of all *declarations* in field `constants`,
- the *file* this module was declared in in field `file`,

`\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

Test 3

```
 \ExplSyntaxOn
\stex__modules__current__namespace:
Namespace~1:\\ \l__stex__modules__ns__str \\
Faking~a~repository:\\
\stex__set__current__repository:n{Foo/Bar}
\seq__pop__right:NN \g__stex__currentfile__seq \testtemp
\edef\testtempb{\detokenize{source}}
\exp__args:NNo \seq__put__right:Nn \g__stex__currentfile__seq { \testtempb }
\edef\testtempb{\detokenize{test}}
\exp__args:NNo \seq__put__right:Nn \g__stex__currentfile__seq { \testtempb }
\exp__args:NNo \seq__put__right:Nn \g__stex__currentfile__seq { \testtemp }
\stex__modules__current__namespace:
Namespace~2:\\ \l__stex__modules__ns__str
\ExplSyntaxOff
```

```
Namespace 1:
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest
```

.

3.4.1 The `module`-environment

`module` `\begin{module}[⟨options⟩]{⟨name⟩}`

Opens a new module with name ⟨*name*⟩.
TODO document options.

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

`@module` `\begin{@module}[⟨options⟩]{⟨name⟩}`

Core functionality of the `module`-environment without a header.

Test 4

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

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

.

Test 5

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

Module 3.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:

.

3.4.2 SMS Mode

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

Test 6

```
 \immediate\openout\testfile=./tests/sometest.tex
\immediate\write\testfile{\detokenize{\this is \a test}^^J}
\immediate\write\testfile{\detokenize{this \is a \test}}
\immediate\closeout\testfile
\ExplSyntaxOn
\stex_in_smsmode:nn { foo } {
\input{tests/sometest.tex}
}
\ExplSyntaxOff
```

.

3.4.3 Imports and Inheritance

`\importmodule`

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

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

Test 7

```
 \begin{module}{Foo}
\symdecl[name=foo, args=3]{bar}
\symdecl[args=bai]{foobar}
Meaning:~\present\bar\\
\end{module}
Meaning:~\present\bar\\
\begin{module}{Importtest}
\importmodule{Foo}
Meaning:~\present\bar\\
\end{module}
\begin{module}{Importtest2}
\importmodule{Importtest}
Meaning:~\present\bar\\
\end{module}
```

Module 3.2[Foo]
Meaning: »`macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}`«

Meaning: »`macro:->\protect \bar `«

Module 3.3[Importtest]
Meaning: »`macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}`«

Module 3.4[Importtest2]
Meaning: »`macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}`«

.

`\usemodule`

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

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

Test 8

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

Module 3.5[UseTest1]

Module 3.6[UseTest2]

```
    Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest1?foo}«
```

Module 3.7[UseTest3]

```
    Meaning: »undefined«
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest2?bar}«
```

.

Test 9

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

Circular dependencies:

Module 3.8[CircDep1]

```
    »macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}«
»macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo//circular2?Circular2?fooB}«
```

.

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

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

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

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

     That module should have the same namespace as the current one.
   - (b) If ⟨*path*⟩ is not empty, it must point to the relative path of the containing file as well as the namespace.
2. Otherwise:
   - (a) If ⟨*path*⟩ is empty, then ⟨*name*⟩ must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name ⟨*name*⟩`.`⟨*lang*⟩`.tex` must exist in the top `source` folder of the archive, containing a module ⟨*name*⟩.

     That module should lie directly in the namespace of the archive.
   - (b) If ⟨*path*⟩ is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

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

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

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

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

`\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 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}`.

\abbrdef

`\abbrdef[⟨args⟩]{⟨macroname⟩}{⟨term⟩}`

`\abbrdef` behaves like `\symdecl`, but adds the definiens ⟨*term*⟩ to the symbol. The latter is largely ignored and irrelevant to sTeX, but exported to OMDoc.

\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 string),
- `assocs` (integer string; number of associative arguments),

Test 10

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

```
Module 3.9[SymdeclTest]
      Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?SymdeclTest?foo}«
Result: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?SymdeclTest?foo
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?SymdeclTest?bardef}«
```

.

`\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...

`\stex_invoke_symbol:n`

Executes a semantic macro. Outside of math mode or if followed by `*`, it continues to `\stex_term_custom:nn`. In math mode, it uses the default or optionally provided notation of the associated symbol.

If followed by `!`, it will invoke the symbol *itself* rather than its application (and continue to `\stex_term_custom:nn`), i.e. it allows to refer to `\plus![addition]` as an operation, rather than `\plus[addition of]{some}{terms}`.

`\notation`

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

Introduces a new notation for ⟨*symbol*⟩, see `\stex_notation_do:nn`

`\stex_notation_do:nn`

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

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

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

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

Test 11

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

Module 3.10[NotationTest]

.

`\symdef`

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

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

Test 12

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

Module 3.11[SymdefTest]
$a+b+c$

.

`\_stex_term_math_oms:nnnn`
`\_stex_term_math_oma:nnnn`
`\_stex_term_math_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_math_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_math_assoc_arg:nnnn`

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

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

`\infprec`
`\neginfprec`

Maximal and minimal notation precedences.

`\STEXdobrackets`

`\STEXdobrackets {`⟨*body*⟩`}`

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

`\STEXwithbrackets`

`\STEXwithbrackets` ⟨*left*⟩ ⟨*right*⟩ `{`⟨*body*⟩`}`

Temporarily (i.e. within ⟨*body*⟩) sets the brackets used by sTeX for automated bracketing (by default `(` and `)`) to ⟨*left*⟩ and ⟨*right*⟩.

Note that ⟨*left*⟩ and ⟨*right*⟩ need to be allowed after `\left` and `\right` in display-mode.

Test 13

```
 \begin{module}{MathTest1}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1 ^ {#2}}_{#3} \comp\rangle }
$\bar abc$ and $\bar[foo] abc$.

\end{module}
```

Module 3.12[MathTest1]
$\langle {a^{b}}_{c}\rangle$ and $\langle {a^{b}}_{c}\rangle$.

.

Test 14

```
 \begin{module}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ]^{#3} \comp\rangle }{ {#1}_{\com
$\foobar a{b,c,d,e,f}g$ and $\foobar[foo] a{b,c}g$ and $\foobar abc$

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

Module 3.13[MathTest2]
$\langle a|[b{:}c_{:d_{:e_{:f}}}]^{g}\rangle$ and $\langle a|[b{:}c]^{g}\rangle$ and $\langle a|[b]^{c}\rangle$
$a+b\cdot c$ and $a\cdot(\frac{a}{b}+\frac{a}{c})$

$$a+b\cdot c\text{ and }a\cdot\left(\frac{a}{b}+\frac{a}{c}\right)$$

$\displaystyle a+b\cdot c$ and $\displaystyle a\cdot\left[\frac{a}{b}+\frac{a}{c}\right]$

.

`\stex_term_custom:nn`

`\stex_term_custom:nn{`⟨*URI*⟩`}{`⟨*args*⟩`}`

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

Test 15

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

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

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

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

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

\bar![bar]

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

\end{module}
```

Module 3.14[TextTest]
some a and some b and also some c here.
some *a* and some *b* and also some *c* here.
`bar`
or just some c
bar
or first b, then c, and finally a

.

`\stex_highlight_term:nn`

`\stex_highlight_term:nn{`⟨*URI*⟩`}{`⟨*args*⟩`}`

Establishes a context for `\comp`. Stores the URI in a variable so that `\comp` knows which symbol governs the current notation.

`\comp`
`\@comp`

`\comp{`⟨*args*⟩`}`

Marks ⟨*args*⟩ as a notation component of the current symbol for highlighting, linking, etc.

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

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 8.*)

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}{}
```

\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 8.*)

\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 8.*)

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 8.*)

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
_env
\stex_annotate_invisible:n
\stex_annotate_invisible:nnn

We define four macros for introducing attributes in the HTML output. The definitions depend on the "backend" used (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 8.*)

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 9.*)

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`

```
\cs_new_protected:Nn \stex_path_from_string:Nn {
  \str_set:Nx \l_tmpa_str { #2 }
  \str_if_empty:NTF \l_tmpa_str {
    \seq_clear:N #1
  }{
    \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
    \sys_if_platform_windows:T{
      \seq_clear:N \l_tmpa_tl
      \seq_map_inline:Nn #1 {
        \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
        \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
      }
      \seq_set_eq:NN #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 9.*)

`\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 9.*)

`\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 9.*)

\stex_path_if_absolute_p:N
\stex_path_if_absolute:N*TF*

```
\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 9.*)

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 8.*)

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 9.*)

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 9.*)

`\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 9.*)

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_if_absolute:NF \c_stex_mathhub_seq {
    \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
      \c_stex_pwd_str/\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 10.*)

`\__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 11.*)

\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 11.*)

`\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 }
      \c_stex_mathhub_main_manifest_prop
    \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
    \stex_debug:n{Current~repository:~
      \prop_item:Nn \l_stex_current_repository_prop {id}
    }
  }
}
```

(*End definition for* `\l_stex_current_repository_prop`. *This variable is documented on page 10.*)

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 11.*)

`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 12.*)

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 12.*)

\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 12.*)

\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 12.*)

\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 12.*)

\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 12.*)

\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 12.*)

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 ,
    imports    = \exp_not:o { \l_tmpa_seq } ,
    constants = \exp_not:o { \l_tmpa_seq } ,
    content   = \exp_not:o { \l_tmpa_tl }  ,
    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 `\end{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_debug:n{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
  \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 } ,
        imports    = \prop_item:cn { \l_tmpa_str } { imports } ,
        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:
}
```

`\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
    }\par
  }
}
```

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

Finally:

```
\NewDocumentEnvironment { module } { O{} m } {
  \bool_if:NT \c_stex_showmods_bool {
    \begin{mdframed}
  }
  \begin{@module}[#1]{#2}
  \stex_modules_heading:
}{
  \end{@module}
  \bool_if:NT \c_stex_showmods_bool {
    \end{mdframed}
  }
}
```

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 14.*)

`\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 14.*)

`\__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 14.*)

__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 14.*)

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

```
\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:onTF \l_tmpa_str { begin } {
          \peek_analysis_map_break:n {
            \exp_after:wN \__stex_smsmode_checkbegin:n ##1
          }
        } {
          \str_if_eq:onTF \l_tmpa_str { end } {
            \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} } {
              \stex_debug:n{Executing~1:~\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} } {
                \stex_debug:n{Executing~2:~\l_tmpa_str}
                % TODO \__stex_smsmode_rescan_cs:
%                \exp_after:wN \exp_after:wN \exp_after:wN
%                \token_if_eq_charcode:NNTF \exp_after:wN \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`.)

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__stex_importmodule_path_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_tmpb_seq ? }

  \str_if_empty:NTF \l__stex_importmodule_archive_str {
    \stex_modules_current_namespace:
    \str_if_empty:NF \l__stex_importmodule_path_str {
      \str_set:Nx \l_stex_module_ns_str {
        \l_stex_module_ns_str / \l__stex_importmodule_path_str
      }
    }
  }{
    \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:NF \l__stex_importmodule_path_str {
      \str_set:Nx \l_stex_module_ns_str {
        \l_stex_module_ns_str / \l__stex_importmodule_path_str
      }
    }
  }
}
```

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

\l__stex_importmodule_name_str
\l__stex_importmodule_archive_str
\l__stex_importmodule_path_str
\l__stex_importmodule_file_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
\str_new:N \g__stex_importmodule_file_str
```

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

\stex_import_require_module:nnnn

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

```
\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
    } {
      \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
      \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
      \seq_put_right:Nn \l_tmpa_seq { source }
    }

    \stex_debug:n{Arguments: #1, #2, #3, #4}

    % 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
      }

      \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
      \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
        \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
      }{
        \stex_debug:n{Checking~\l_tmpa_str.tex}
        \IfFileExists{ \l_tmpa_str.tex }{
          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
        }{
          % try english as default
          \stex_debug:n{Checking~\l_tmpa_str.en.tex}
          \IfFileExists{ \l_tmpa_str.en.tex }{
            \str_gset:Nx \g__stex_importmodule_file_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_split:NnV \l_tmpb_seq / \l_tmpb_str
      \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq

      \cs_if_exist:NTF \languagename {
        \exp_args:NNx \prop_get:NnN \c_stex_language_abbrevs_prop
            { \languagename } \l_tmpb_str
      }{
        \str_clear:N \l_tmpb_str
      }

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

      \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
      \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
        \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
      }{
        \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
        \IfFileExists{ \l_tmpa_str/#4.tex }{
          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
        }{
          % try english as default
          \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
          \IfFileExists{ \l_tmpa_str/#4.en.tex }{
            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
          }{
            \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
            \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
            }{
              \stex_debug:n{Checking~\l_tmpa_str.tex}
              \IfFileExists{ \l_tmpa_str.tex }{
                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
              }{
                % try english as default
                \stex_debug:n{Checking~\l_tmpa_str.en.tex}
                \IfFileExists{ \l_tmpa_str.en.tex }{
                  \str_gset:Nx \g__stex_importmodule_file_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:Nnx \use:nn {
        \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
          \prop_clear:N \l_stex_current_module_prop
          \str_set:Nx \l_tmpb_str { #2 }
          \str_if_empty:NF \l_tmpb_str {
            \stex_set_current_repository:n { #2 }
          }
          \stex_debug:n{Loading~\g__stex_importmodule_file_str}
          \input { \g__stex_importmodule_file_str }
        }
%     }{

%     }
      \prop_gput:Noo \g_stex_module_files_prop
      \g__stex_importmodule_file_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~\g__stex_importmodule_file_str
        }
        \msg_error:nn{stex}{error/modulemissing}
      }
    }
    % activate
    \stex_debug:n{Activating~module~#1?#4}
    \prop_item:cn { c_stex_module_#1?#4_prop } { content }
}
```

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

`\importmodule`

```
\NewDocumentCommand \importmodule { O{} m } {
  \stex_import_module_uri:nn { #1 } { #2 }
  \stex_debug:n{Importing~module:~
    \l_stex_module_ns_str ? \l__stex_importmodule_name_str
  }
  \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 15.*)

`\usemodule`

```
\NewDocumentCommand \usemodule { O{} m } {
  \stex_if_smsmode:F {
    \stex_import_module_uri:nn { #1 } { #2 }
    \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
      {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 15.*)

`\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 17.*)

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 18.*)

\abbrdef

```
\NewDocumentCommand \abbrdef { O{} m m } {
  \__stex_symdecl_args:n { #1 }
  \tl_set:Nn \l_stex_symdecl_definiens_tl { #3 }
  \stex_symdecl_do:n { #2 }
}
```

(*End definition for* `\abbrdef`. *This function is documented on page 18.*)

\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:nnn{type}{}{$\l_stex_symdecl_type_tl$}
      \stex_annotate_invisible:nnn{args}{}{
        \prop_item:Nn \l_tmpa_prop { args }
      }
      \stex_annotate_invisible:nnn{macroname}{}{#1}
      \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
        \stex_annotate_invisible:nnn{definiens}{}
          {$\l_stex_symdecl_definiens_tl$}
      }
    }
  }
}
```

(*End definition for* `\stex_symdecl_do:n`*. This function is documented on page 18.*)

\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 {
    \__stex_symdecl_get_symbol_from_cs:n { #1 }
  }{
    % argument is a string
    % is it a command name?
    \cs_if_exist:cTF { #1 }{
      \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
    }{
      % TODO
      % argument is not a command name
    }
  }
}

\cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
  \tl_set:Nx \l_tmpa_tl { #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 19.*)

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 }

  \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
  \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
  \str_set:Nx \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:nn { \l_stex_get_symbol_uri_str }
}
```

(*End definition for* `\notation`. *This function is documented on page 19.*)

\stex_notation_do:nn

```
\cs_new_protected:Nn \stex_notation_do:nn {
  \prop_set_eq:Nc \l_tmpa_prop {
    g_stex_symdecl_ #1 _prop
  }

  \prop_clear:N \l_tmpb_prop
  \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
  \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
```

```
\prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str

% precedences
\seq_clear:N \l_tmpb_seq
\exp_args:NNno
\str_if_empty:NTF \l__stex_notation_prec_str {
  \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
  \int_compare:nNnTF \l_tmpa_str = 0 {
    \exp_args:NNnx
    \prop_put:Nnn \l_tmpb_prop { opprec }
      { \int_use:N \infprec }
  }{
    \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
  }
} {
  \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
  \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
    \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
    \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
      \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
        \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
      \seq_map_inline:Nn \l_tmpa_seq {
        \seq_put_right:Nn \l_tmpb_seq { ##1 }
      }
    }
    \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
  }{
    \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
    \int_compare:nNnTF \l_tmpa_str = 0 {
      \exp_args:NNnx
      \prop_put:Nnn \l_tmpb_prop { opprec }
        { \int_use:N \infprec }
    }{
      \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
    }
  }
}

\seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
\int_step_inline:nn { \l_tmpa_str } {
  \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
    \exp_args:NNx
    \seq_put_right:Nn \l_tmpb_seq {
      \prop_item:Nn \l_tmpb_prop { opprec }
    }
  }
}

\prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
\tl_clear:N \l_tmpa_tl

\int_compare:nNnTF \l_tmpa_str = 0 {
  \cs_set:Npx \l__stex_notation_macrocode_cs {
    \_stex_term_math_oms:nnnn { #1 }
```

```
        { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
        { \prop_item:Nn \l_tmpb_prop { opprec } }
        { #2 }
    }
    \__stex_notation_final:
  }{
    \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
    \str_if_in:NnTF \l_tmpb_str b {
      \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
      \cs_set:Npx \l_tmpa_str {
        \_stex_term_math_omb:nnnn { #1 }
          { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
          { \prop_item:Nn \l_tmpb_prop { opprec } }
          { #2 }
      }
    }{
      \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
      \cs_set:Npx \l_tmpa_str {
        \_stex_term_math_oma:nnnn { #1 }
          { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
          { \prop_item:Nn \l_tmpb_prop { opprec } }
          { #2 }
      }
    }

    \int_zero:N \l_tmpa_int
    \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
    \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
    \__stex_notation_arguments:
  }
}
```

(*End definition for* `\stex_notation_do:nn`*. This function is documented on page 19.*)

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

```
\cs_new_protected:Nn \__stex_notation_arguments: {
  \int_incr:N \l_tmpa_int
  \str_if_empty:NTF \l_tmpa_str {
    \__stex_notation_final:
  }{
    \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
    \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
    \str_if_eq:VnTF \l_tmpb_str a {
      \__stex_notation_argument_assoc:n
    }{
      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
      \tl_put_right:Nx \l_tmpa_tl {
        { \_stex_term_math_arg:nnn
          { \int_use:N \l_tmpa_int }
          { \l_tmpb_str }
          { ####\int_use:N \l_tmpa_int }
        }
      }
      \__stex_notation_arguments:
    }
  }
}
```

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

`\__stex_notation_argument_assoc:n`

```
\cs_new_protected:Nn \__stex_notation_argument_assoc:n {
  \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
  \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
  \tl_put_right:Nx \l_tmpa_tl {
    { \_stex_term_math_assoc_arg:nnnn
      { \int_use:N \l_tmpa_int }
      { \l_tmpb_str }
      { \l_tmpa_cs {########1} {########2} }
      { ####\int_use:N \l_tmpa_int }
    }
  }
  \__stex_notation_arguments:
}
```

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

`\__stex_notation_final:` Called after processing all notation arguments

```
\cs_new_protected:Nn \__stex_notation_final: {
  \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
  \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
  \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
  \cs_generate_from_arg_count:cNnn {
      stex_notation_ \l_tmpa_str \c_hash_str
      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
      _cs
    }
    \cs_set:Npx \l_tmpb_str {
      \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl
  }

  \stex_debug:n{
    Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
    ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
    Operator~precedence:~
      \prop_item:Nn \l_tmpb_prop { opprec }^^J
    Argument~precedences:~
      \seq_use:Nn \l_tmpa_seq {,~}^^J
    Notation: \cs_meaning:c {
      stex_notation_ \l_tmpa_str \c_hash_str
      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
      _cs
    }
  }

  \prop_gset_eq:cN {
    g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
      \c_hash_str \l__stex_notation_lang_str _prop
```

```
} \l_tmpb_prop

\exp_args:Nx
\stex_add_to_current_module:n {
  \prop_get:cnN {
    g_stex_symdecl_
      \prop_item:Nn \l_tmpb_prop { symbol }
    _prop
  } { notations } \exp_not:N \l_tmpa_seq
  \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
    \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
  }
  \prop_put:cno {
    g_stex_symdecl_
      \prop_item:Nn \l_tmpb_prop { symbol }
    _prop
  } { notations } \exp_not:N \l_tmpa_seq
}

\stex_if_smsmode:TF {
  \stex_smsmode_set_codes:
  \exp_args:Nx \stex_addtosms:n {
    \prop_gset_from_keyval:cn {
      g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
        \c_hash_str \l__stex_notation_lang_str _prop
    } {
      symbol    = \prop_item:Nn \l_tmpb_prop { symbol }
      language  = \prop_item:Nn \l_tmpb_prop { language }
      variant   = \prop_item:Nn \l_tmpb_prop { variant }
      opprec    = \prop_item:Nn \l_tmpb_prop { opprec }
      argprecs  = \prop_item:Nn \l_tmpb_prop { argprecs }
    }
  }
}{
  \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
  \seq_put_right:Nx \l_tmpa_seq {
    \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
  }
  \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
  \prop_set_eq:cN {
    g_stex_symdecl_ \l_tmpa_str _prop
  } \l_tmpa_prop

  % HTML annotations
  \stex_annotate_invisible:nnn { notation }
    { \prop_item:Nn \l_tmpb_prop { symbol } } {
      \stex_annotate_invisible:nnn { notationfragment }
        { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
      \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
      \stex_annotate_invisible:nnn { precedence }
        { \prop_item:Nn \l_tmpb_prop { opprec };
          \seq_use:Nn \l_tmpa_seq { x }
        }{}

        \int_zero:N \l_tmpa_int
        \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
        \tl_clear:N \l_tmpa_tl
        \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
          \int_incr:N \l_tmpa_int
          \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
          \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
          \str_if_eq:VnTF \l_tmpb_str a {
            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
              \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
              \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
            } }
          }{
            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
              \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
            } }
          }
        }
        \stex_annotate_invisible:nnn { notationcomp }{}{
          $ \exp_args:Nno \use:nn { \use:c {
            stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
            \c_hash_str \l__stex_notation_variant_str
            \c_hash_str \l__stex_notation_lang_str _cs
          } } { \l_tmpa_tl } $
        }
      }
  }
}
```

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

\symdef

```
\keys_define:nn { stex / symdef } {
  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 ,
  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_symdef_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
  \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 /symdef } { #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
  \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
}

\NewDocumentCommand \symdef { O{} m } {
  \__stex_notation_symdef_args:n { #1 }
  \tl_clear:N \l_stex_symdecl_definiens_tl
  \stex_symdecl_do:n { #2 }
  \exp_args:Nx \stex_notation_do:nn {
    \prop_item:Nn \l_tmpa_prop { module } ?
    \prop_item:Nn \l_tmpa_prop { name }
  }
}
```

(*End definition for* `\symdef`*. This function is documented on page 20.*)

`\stex_invoke_symbol:n` Invokes a semantic macro

```
\cs_new_protected:Nn \stex_invoke_symbol:n {
  \peek_charcode_remove:NTF ! {
    \stex_term_custom:nn { #1 } { }
  } {
    \if_mode_math:
      \exp_after:wN \__stex_notation_invoke_math:n
    \else:
      \exp_after:wN \__stex_notation_invoke_text:n
    \fi: { #1 }
  }
}
```

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

`\__stex_notation_invoke_math:n`

```
\cs_new_protected:Nn \__stex_notation_invoke_math:n {
  \peek_charcode_remove:NTF * {
    \__stex_notation_invoke_text:n { #1 }
  }{
    \peek_charcode:NTF [ {
      \__stex_notation_invoke_math:nw { #1 }
    }{
      \__stex_notation_invoke_math:nw { #1 } []
    }
  }
}
```

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

__stex_notation_invoke_math:nw

```
\cs_new_protected:Npn \__stex_notation_invoke_math:nw  #1 [#2] {
  \__stex_notation_args:n { #2 }
  \prop_set_eq:Nc \l_tmpa_prop {
    g_stex_symdecl_ #1 _prop
  }
  \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
  \seq_if_empty:NTF \l_tmpa_seq {
    \msg_set:nnn{stex}{error/nonotations}{
      Symbol~#1~used,~but~has~no~notations!
    }
    \msg_error:nn{stex}{error/nonotations}
  } {
    \seq_if_in:NxTF \l_tmpa_seq
      { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
      \use:c{
        stex_notation_ #1 \c_hash_str
        \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
        _cs
      }
    }{
      \str_if_empty:NTF \l__stex_notation_variant_str {
        \str_if_empty:NTF \l__stex_notation_lang_str {
          \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
          \use:c{
            stex_notation_ #1 \c_hash_str \l_tmpa_str
            _cs
          }
        }{
          \msg_set:nnn{stex}{error/wrongnotation}{
            Symbol~#1~has~no~notation~
            \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
          }
          \msg_error:nn{stex}{error/wrongnotation}
        }
      }{
        \msg_set:nnn{stex}{error/wrongnotation}{
          Symbol~#1~has~no~notation~
          \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
        }
        \msg_error:nn{stex}{error/wrongnotation}
      }
    }
  }
}
```

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

__stex_notation_invoke_text:n

```
\cs_new_protected:Nn \__stex_notation_invoke_text:n {
  \prop_set_eq:Nc \l_tmpa_prop {
    g_stex_symdecl_ #1 _prop
  }
  \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
  \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
}
```

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

4.8 Terms

```
⟨@@=stex_term⟩
```

Precedences:

\infprec
\neginfprec
\l__stex_term_downprec

```
\int_const:Nn \infprec {\c_max_int}
\int_const:Nn \neginfprec {-\c_max_int}
\int_new:N \l__stex_term_downprec
\int_set_eq:NN \l__stex_term_downprec \neginfprec
```

(*End definition for* `\infprec`*,* `\neginfprec`*, and* `\l__stex_term_downprec`*. These variables are documented on page 20.*)

Bracketing:

\l__stex_term_left_bracket_str
\l__stex_term_right_bracket_str

```
\tl_set:Nn \l__stex_term_left_bracket_str (
\tl_set:Nn \l__stex_term_right_bracket_str )
\RequirePackage{scalerel}
```

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

__stex_term_maybe_brackets:nn

Compares precedences and insert brackets accordingly

```
\cs_new_protected:Nn \__stex_term_maybe_brackets:nn {
  \int_compare:nNnTF { #1 } < \l__stex_term_downprec {
    \STEXdobrackets { #2 }
  }{ #2 }
}
```

(*End definition for* `\__stex_term_maybe_brackets:nn`*.*)

\STEXdobrackets

```
\cs_new_protected:Npn \STEXdobrackets #1 {
  \ThisStyle{\if D\m@switch
      \exp_args:Nnx \use:nn
      { \left\l__stex_term_left_bracket_str #1 }
      { \right\l__stex_term_right_bracket_str }
    \else
      \exp_args:Nnx \use:nn
      { \l__stex_term_left_bracket_str #1 }
      { \l__stex_term_right_bracket_str }
  \fi}
}
```

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

\STEXwithbrackets

```
\cs_new_protected:Npn \STEXwithbrackets #1 #2 #3 {
  \exp_args:Nnx \use:nn
  {
    \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
    \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
    #3
  }
  {
    \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
      {\l__stex_term_left_bracket_str}
    \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
      {\l__stex_term_right_bracket_str}
  }
}
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
\cs_new_protected:Nn \_stex_term_oms:nnn {
  \stex_annotate:nnn{ OMID }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_oms:nnnn {
  \__stex_term_maybe_brackets:nn { #3 }{
    \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_oms:nnnn`. *This function is documented on page 20.*)

_stex_term_math_oma:nnnn

```
\cs_new_protected:Nn \_stex_term_oma:nnn {
  \stex_annotate:nnn{ OMA }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_oma:nnnn {
  \__stex_term_maybe_brackets:nn { #3 }{
    \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_oma:nnnn`. *This function is documented on page 20.*)

_stex_term_math_omb:nnnn

```
\cs_new_protected:Nn \_stex_term_ombind:nnn {
  \stex_annotate:nnn{ OMBIND }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_omb:nnnn {
  \__stex_term_maybe_brackets:nn { #3 }{
    \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_omb:nnnn`. *This function is documented on page 20.*)

`\_stex_term_math_arg:nnn`

```
\cs_new_protected:Nn \_stex_term_arg:nn {
  \stex_unhighlight_term:n {
    \stex_annotate:nnn{ arg }{ #1 }{ #2 }
  }
}
\cs_new_protected:Nn \_stex_term_math_arg:nnn {
  \exp_args:Nnx \use:nn
    { \int_set:Nn \l__stex_term_downprec { #2 }
       \_stex_term_arg:nn { #1 } { #3 }
    }
    { \int_set:Nn \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
}
```

(*End definition for* `\_stex_term_math_arg:nnn`. *This function is documented on page 20.*)

`\_stex_term_math_assoc_arg:nnnn`

```
\cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
  \seq_set_split:Nnn \l_tmpa_seq , { #4 }
  \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
    \tl_set:Nn \l_tmpa_tl { #4 }
  }{
    \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
    \seq_reverse:N \l_tmpa_seq
    \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
    \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
    \seq_map_inline:Nn \l_tmpa_seq {
      \tl_set:Nx \l_tmpa_tl {
        \exp_args:Nno
        \l_tmpa_cs { ##1 } { \l_tmpa_tl }
      }
    }
  }
  \exp_args:Nnno
  \_stex_term_math_arg:nnn{#1}{#2}{ \l_tmpa_tl }
}
```

(*End definition for* `\_stex_term_math_assoc_arg:nnnn`. *This function is documented on page 20.*)

`\stex_term_custom:nn`

```
\cs_new_protected:Nn \stex_term_custom:nn {
  \str_set:Nn \l__stex_term_custom_uri { #1 }
  \str_set:Nn \l_tmpa_str { #2 }
  \tl_clear:N \l_tmpa_tl
  \int_zero:N \l_tmpa_int
  \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
  \__stex_term_custom_loop:
}
```

(*End definition for* `\stex_term_custom:nn`. *This function is documented on page 21.*)

`\__stex_term_custom_loop:`

```
\cs_new_protected:Nn \__stex_term_custom_loop: {
  \bool_set_false:N \l_tmpa_bool
  \bool_while_do:nn {
    \str_if_eq_p:ee X {
      \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
    }
  }{
    \int_incr:N \l_tmpa_int
  }

  \peek_charcode:NTF [ {
    % notation/text component
    \__stex_term_custom_component:w
  } {
    \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
      % all arguments read => finish
      \__stex_term_custom_final:
    } {
      % arguments missing
      \peek_charcode_remove:NTF * {
        % invisible, specific argument position or both
        \peek_charcode:NTF [ {
          % visible specific argument position
          \__stex_term_custom_arg:wn
        } {
          % invisible
          \peek_charcode_remove:NTF * {
            % invisible specific argument position
            \__stex_term_custom_arg_inv:wn
          } {
            % invisible next argument
            \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
          }
        }
      } {
        % next normal argument
        \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
      }
    }
  }
}
```

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

`\__stex_term_custom_arg_inv:wn`

```
\cs_new_protected:Npn \__stex_term_custom_arg_inv:wn [ #1 ] #2 {
  \bool_set_true:N \l_tmpa_bool
  \__stex_term_custom_arg:wn [ #1 ] { #2 }
}
```

(*End definition for* `\__stex_term_custom_arg_inv:wn`.)

`\__stex_term_custom_arg:wn`

```
\cs_new_protected:Npn \__stex_term_custom_arg:wn [ #1 ] #2 {
  \str_set:Nx \l_tmpb_str {
    \str_item:Nn \l_tmpa_str { #1 }
  }
  \str_case:VnTF \l_tmpb_str {
    { X } { } % TODO throw error
    { i } { \__stex_term_custom_set_X:n { #1 } }
    { b } { \__stex_term_custom_set_X:n { #1 } }
    { a } { } % TODO ?
  }{}{
    % TODO throw error
  }

  \bool_if:nTF \l_tmpa_bool {
    \tl_put_right:Nx \l_tmpa_tl {
      \stex_annotate_invisible:n {
        \_stex_term_arg:nn { \int_eval:n { #1 } }
          \exp_not:n { { #2 } }
      }
    }
  } {
    \tl_put_right:Nx \l_tmpa_tl {
      \_stex_term_arg:nn { \int_eval:n { #1 } }
        \exp_not:n { { #2 } }
    }
  }

  \__stex_term_custom_loop:
}
```

(*End definition for* `\__stex_term_custom_arg:wn`.)

`\__stex_term_custom_set_X:n`

```
\cs_new_protected:Nn \__stex_term_custom_set_X:n {
  \str_set:Nx \l_tmpa_str {
    \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
    X
    \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
  }
}
```

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

`\__stex_term_custom_component:`

```
\cs_new_protected:Npn \__stex_term_custom_component:w [ #1 ] {
  \tl_put_right:Nn \l_tmpa_tl { #1 }
  \__stex_term_custom_loop:
}
```

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

__stex_term_custom_final:

```
\cs_new_protected:Nn \__stex_term_custom_final: {
  \int_compare:nNnTF \l_tmpb_int = 0 {
    \exp_args:Nnno \_stex_term_oms:nnn
  }{
    \str_if_in:NnTF \l_tmpa_str {b} {
      \exp_args:Nnno \_stex_term_ombind:nnn
    } {
      \exp_args:Nnno \_stex_term_oma:nnn
    }
  }
  { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
}
```

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

\stex_highlight_term:nn

```
\latexml_if:F {
  \scalatex_if:F{
    \RequirePackage{pdfcomment}
  }
}

\str_new:N \l__stex_term_highlight_uri_str
\cs_new_protected:Nn \stex_highlight_term:nn {
  \latexml_if:TF {
    #2
  } {
    \scalatex_if:TF {
      #2
    } {
      \exp_args:Nnx
      \use:nn {
        \str_set:Nx \l__stex_term_highlight_uri_str { #1 }
        #2
      } {
        \str_set:Nx \exp_not:N \l__stex_term_highlight_uri_str
          { \l__stex_term_highlight_uri_str }
      }
    }
  }
}

\cs_new_protected:Nn \stex_unhighlight_term:n {
%  \latexml_if:TF {
%    #1
%  } {
%    \scalatex_if:TF {
%      #1
%    } {
      #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
%    }
%  }
}
```

(*End definition for* `\stex_highlight_term:nn`. *This function is documented on page 22.*)

\comp
\@comp

```
\cs_new_protected:Npn \comp #1 {
  \str_if_empty:NF \l__stex_term_highlight_uri_str {
    \exp_args:Nnx \@comp { #1 } { \l__stex_term_highlight_uri_str }
  }
}

\cs_new_protected:Npn \@comp #1 #2 {
  \pdftooltip {
    \textcolor{blue}{#1}
  } { #2 }
}
```

(*End definition for* `\comp` *and* `\@comp`. *These functions are documented on page 22.*)