The Simple Text Markup Language Specification

This document serves as the formal specification of STML, the Simple Text Markup Language.

Contents

- About STML
- Syntax
  - Tokens
  - Abstract Syntax Tree
  - Structure
- Transport
- Elements
  - Document

About STML

STML is an `s-expression`-based format for encoding static content (note: the actual markup may be dynamically-created, but this does not happen on the client side).

The goals of STML:

- Remain lightweight. No complex layout or styling rules should ever be added to STML. Writing an STML viewer should not be a high-difficulty endeavor.
- STML documents should be accessible (both in terms of being able to consume content regardless of physical abilities, and in terms of data usage).
- STML should primarily be a vehicle for transporting content, STML provides form controls, in order to allow for interacting with servers, but these forms cannot be used automatically, or without the user's approval. Intrusive advertisements and tracking tools that invade user privacy should be considered dark patterns, and actively prevented by both the specification, and STML viewers.

Syntax

STML syntax consists solely of s-expressions, which can be parsed very quickly.

Tokens

The following is a `flex` unit that will lex all valid syntax.

```
%x STRING

NUMBER -?[0-9]+(\.[0-9]+)?
ID [A-Za-z][A-Za-z0-9]*(-([A-Za-z][A-Za-z0-9]*))*
WHITESPACE [ \n\r\t]+
COMMENT ;[^\n]*

HEX \\x[A-Fa-f0-9][A-Fa-f0-9]
UNICODE \\u\{[A-Fa-f0-9]+\}

%%

WHITESPACE ;
COMMENT ;

'(' return LPAREN;
')' return RPAREN;
'?' return QUESTION;
'"' { yy_push_state(STRING); return DOUBLE_QUOTE; }
{NUMBER} return NUMBER;
{ID} return IDENTIFIER;

<STRING>\\" return ESCAPED_QUOTE;
<STRING>{HEX} return ESCAPED_HEX;
<STRING>{UNICODE} return ESCAPED_UNICODE;
<STRING>'"' { yy_pop_state(); return DOUBLE_QUOTE; }
<STRING>.+ return PLAIN_TEXT;
```

Abstract Syntax Tree

STML is first parsed into an abstract syntax tree, and is then "interpreted" by being rendered into some (typically on-screen) display. Documents that are syntactically valid may not be semantically valid. Creators of STML documents are advised to run validation tools before publishing content.

The following is a `bison` grammar that will parse all valid syntax.

```
compilation-unit: expression_list;

expression:
  IDENTIFIER
  | NUMBER
  | DOUBLE_QUOTE string_part_list DOUBLE_QUOTE
  | expression QUESTION
  | LPAREN expression RPAREN
  | LPAREN nonempty_expression_list RPAREN;
```

```
string_part_list:
  %empty
  | string_part_list string_part;

string_part:
  ESCAPED_QUOTE
  | ESCAPED_HEX
  | ESCAPED_UNICODE
  | PLAIN_TEXT;

expression_list:
  %empty
  | expression_list expression;

nonempty_expression_list:
  expression
  | expression_list expression;
```

Structure

In practice, a semantically valid file will look something like the following:

```
; This is a comment!
(version "0.0.0")
(document
  (title "Hello, STML!")
  (heading "This is a simple page.")
  (link "Github Page" "https://github.com/thosakwe/stml"))
```

The root node is the Compilation Unit; start there.