The Mk2 Markup Language

Interstellar Ventures

Saturday, 19 September 2015

Table of Contents

This is a draft standard, this notice will disappear once the specification is final.

Mk2 is a human readable plain text language for expressing Geneva documents.[1] It is designed with both ergonomics and technical pragmatism in mind.

- 1. *Geneva Document Specification* (`geneva-document.html`)

1 Syntax

This formal definition uses the modified *BNF syntax* of *ANSI CL's Notational Conventions*.[1] The following axioms are used throughout the definition:

String—A character sequence. The exact grammar depends on the surrounding context. See *Escape Rules*.

LF—A character sequence denoting a line break. The exact representation is platform dependent.

EOF—The end of input.

SP—A whitespace character. The exact set of characters considered whitespace is platform dependent.

- 1. *ANSI Common Lisp: Notational Conventions* (`http://users-phys.au.dk/harder/Notational-Conventions.html`)

1.1 Document and Section

Symbol	**Expression**
document	`[` element separator `]*` *EOF*
section	`"<"` title separator `[` element separator `]*` `">"` separator
title	rich-text
element	section `|` table `|` plaintext `|` media `|` listing `|` paragraph
separator	double-lf `|` *EOF*
double-lf	*LF* `[` *LF* `]+`

Table 1. *Document* and *section* syntax.

1.2 Paragraph and Listing

Symbol	**Expression**
paragraph	text-token+
listing	item+
item	`"+"` rich-text

Table 2. *Paragraph* and *Listing* syntax.

1.3 Table, Media and Plaintext

Symbol	**Expression**
table	`"#table"` description `"#"` *LF* table-body
description	rich-text
table-body	row`*` last-row
row	column+ *LF*
last-row	column+
column	`"|"` rich-text

Table 3. *Table* syntax.

Symbol	**Expression**
media	`"#media"` description `"#"` *LF String*
description	rich-text

Table 4. *Media* syntax.

Symbol	**Expression**
plaintext	`"#code"` description `"#"` *LF* line+ end
description	rich-text
line	*String LF*
end	*SP*`*` `"#"`

Table 5. *Plaintext* syntax.

1.4 Rich Text

Symbol	Expression
rich-text	text-token∗
text-token	bold | italic | fixed-width | url | plain
bold	`"*"` *String* `"*"`
italic	`"_"` *String* `"_"`
fixed-width	`"{"` *String* `"}"`
url	`"["` *String* `"]" [ "("` *String* `")" ]`
plain	*String*

Table 6. *Rich text* syntax.

1.5 Escape Rules

The "\" (backslash) can be used to *escape* the next character. The grammatical significance of a character following "\" is ignored.

The exact grammar of the *String* axiom is context dependent. A *String* may not contain unescaped *terminating sequences*. A terminating sequence is the set of any token following the *String* axiom in a rule and *double-lf*. In order to escape a terminating sequence its first character must be escaped.

For illustration consider the grammar in *Table 7* which utilizes the *String* axiom. In *rule* the *String* axiom is followed by *terminator*, thus "`foo`" is a *terminating sequence* of *String* in *rule*. Valid and invalid character sequences for *String* in *rule* are shown in *Table 8*.

Symbol	Expression
rule	*String* terminator
terminator	`"foo"`

Table 7. Exemplary grammar rules to illustrate escape rules for the *String* axiom.

Valid	Invalid
`quick brown \foo`	`quick brown foo`

Table 8. Valid and invalid character sequences for *String* in *rule*.

2 Examples

2.1 Document and Section

The Mk2 file in Figure 1 contains a paragraph (*A quick brown fox...*) and a section titled "On Pangrams" which contains another paragraph (*A pangram is...*).

```
A quick brown fox jumps over the lazy dog.

< On Pangrams

 A pangram is a phrase that contains all of the letters of the
 alphabet.

>
```

Figure 1

2.2 Listing and Text Tokens

The listing in Figure 2 contains six items, each being a single text token.

```
+ Plain text token
+ *Bold text token*
+ _Italic text token_
+ {Fixed-width text token}
+ [http://example.org/url/text-token]
+ [Labeled URL](http://example.org)
```

Figure 2

2.3 Table, Media and Plaintext

The Mk2 file in Figure 3 contains table, media and plaintext object, each having a description and their respective bodies.

```
#table Source: Wikipedia.#
| State                   | Area          | Total Population
| Bavaria                 | 70,549.44 km² | 12,604,244
| North Rhine-Westphalia | 34,084.13 km² | 17,571,856

#media Imaginary embedded video.#
http://example.org/video.ogv

#code {SQUARE} function in Common Lisp.#
(defun square (n)
  (expt n 2))
#
```

Figure 3

2.4 Escaping

Mk2 is designed to avoid the need of escaping control tokens as much as possible. Still there are some cases where the user has to use the \ (backslash) character to avoid the semantics of a specific token. Below are examples of the most common cases.

Mk2	**Result**
`In ECMAScript anonymous functions can be expressed using the {function (...) { ... \}} special form.`	In ECMAScript anonymous functions can be expressed using the `function (...) { ... }` special form.

Figure 4 Escaping unintended text token markup.

The Mk2 file in Figure 4 escapes the first `}` (curly bracket) character inside a fixed width text token in order to avoid terminating the fixed width token prematurely. Not that only the closing bracket needs to be escaped because it is the only terminating token of the *String* in a fixed width token.

Mk2	**Result**
`On DOS, {\\} (backslash) is used to separate the components of a pathname.`	On DOS, \ (backslash) is used to separate the components of a pathname.

Figure 5 Including the literal backslash character.

Sometimes the user needs to include the literal backslash character in his prose. The \ (backslash) character can be escaped using itself just like any other character as Figure 5 shows.