Open Geneva API Documentation

Interstellar Ventures

Friday, 16 September 2016

Table of Contents

1 geneva **1**
- 1.1 content-type (Function) . . . 2
- 1.2 content-values (Function) . . . 2
- 1.3 make-bold (Function) . . . 2
- 1.4 make-document (Function) . . . 3
- 1.5 make-fixed-width (Function) . . . 3
- 1.6 make-italic (Function) . . . 3
- 1.7 make-listing (Function) . . . 4
- 1.8 make-media (Function) . . . 4
- 1.9 make-paragraph (Function) . . . 4
- 1.10 make-plaintext (Function) . . . 5
- 1.11 make-section (Function) . . . 5
- 1.12 make-table (Function) . . . 5
- 1.13 make-url (Function) . . . 6

2 geneva.macros **6**
- 2.1 document (Macro) . . . 6
- 2.2 listing (Macro) . . . 7
- 2.3 media (Macro) . . . 7
- 2.4 paragraph (Macro) . . . 8
- 2.5 plaintext (Macro) . . . 8
- 2.6 section (Macro) . . . 9
- 2.7 syntax (Variable) . . . 9
- 2.8 table (Macro) . . . 10

3 geneva.mk2 **11**
- 3.1 character-position (Generic Function) . . . 11
- 3.2 line-position (Generic Function) . . . 11
- 3.3 malformed-element (Condition Type) . . . 12
- 3.4 open-section (Condition Type) . . . 12
- 3.5 print-mk2 (Function) . . . 12
- 3.6 read-mk2 (Function) . . . 13

3.7 syntax-error (Condition Type) 14
3.8 unrecognized-input (Condition Type) 14

4 geneva.plain-text **14**
4.1 render-plain-text (Function) 15

5 geneva.html **15**
5.1 render-html (Function) 15
5.2 render-html-file (Function) 16

6 geneva.latex **16**
6.1 render-latex (Function) 16

7 geneva.common-lisp **17**
7.1 api-document (Function) 17
7.2 symbol-document (Function) 17

1 geneva

Geneva core package. Exports functions to programatically create and inspect *Geneva documents*.

In Open Geneva a *document* is represented as a *list* of *document elements*. A *document element* can be obtained using the element constructors `make-paragraph`, `make-listing`, `make-table`, `make-media` and `make-section`. In order to ensure integrity, it is recommended to use `make-document` to produce *documents*.

Rich text is represeted as a *list* of *text tokens*. A *text token* may be a *string* or an object obtained using the text token constructors `make-bold`, `make-italic`, `make-fixed-width` and `make-url`.

Document elements and *text tokens* can be inspected using the readers `content-type` and `content-values`.

Documents and *document elements* are printable and readable using the Common Lisp printer and reader.

Exceptional Situations:

All functions external to this *package* validate their parameters and will signal an *error* of *type* `type-error` on mismatch.

See Also:

- *Geneva Document Specification* (`geneva-document.html`)
- *Open Geneva User Manual* (`open-geneva.html`)

1.1 content-type (Function)

Syntax:

— Function: **content-type** *content*

Arguments and Values:

content—an *element* or a *text token*.

Description:

`content-type` returns a *keyword* denoting the type of *content* which may be one of `:paragraph`, `:listing`, `:table`, `:plaintext`, `:media`, `:section`, `:plain`, `:bold`, `:italic`, `:fixed-width` or `:url`.

1.2 content-values (Function)

Syntax:

— Function: **content-values** *content*

Arguments and Values:

content—an *element* or a *text token*.

Description:

`content-values` returns the components of *content*. The returned values are the *normalized* forms of their respective content constructor's arguments and therefore depend on the type of *content*.

1.3 make-bold (Function)

Syntax:

— Function: **make-bold** *string*

Arguments and Values:

string—a *string*.

Description:

`make-bold` returns a *text token* of type `:bold` for *string*.

1.4 make-document (Function)

Syntax:

— Function: **make-document** *elements*

Arguments and Values:

elements—a *list* of *document elements*.

Description:

`make-document` returns a *document* consisting of *elements*.

1.5 make-fixed-width (Function)

Syntax:

— Function: **make-fixed-width** *string*

Arguments and Values:

string—a *string*.

Description:

`make-fixed-width` returns a *text token* of type `:fixed-width` for *string*.

1.6 make-italic (Function)

Syntax:

— Function: **make-italic** *string*

Arguments and Values:

string—a *string*.

Description:

`make-italic` returns a *text token* of type `:bold` for *string*.

1.7 make-listing (Function)

Syntax:

— Function: **make-listing** *items*

Arguments and Values:

items—a *list* of *rich text* sequences.

Description:

`make-listing` returns a *document element* of type `:listing` with *items*.

1.8 make-media (Function)

Syntax:

— Function: **make-media** *description url*

Arguments and Values:

description—a *rich text* sequence.

url—a *string*.

Description:

`make-media` returns a *document element* of type `:media` with *description* and *url*.

1.9 make-paragraph (Function)

Syntax:

— Function: **make-paragraph** *text*

Arguments and Values:

text—a *rich text* sequence.

Description:

`make-paragraph` returns *document element* of type `:paragraph` with *text*.

1.10 make-plaintext (Function)

Syntax:

— Function: **make-plaintext** *description plaintext*

Arguments and Values:

description—a *rich text* sequence.

plaintext—a *string*.

Description:

`make-plaintext` returns a *document element* of type `:plaintext` with *description* and *plaintext*.

1.11 make-section (Function)

Syntax:

— Function: **make-section** *header elements*

Arguments and Values:

header—a *rich text* sequence.

elements—a *list* of *document elements*.

Description:

`make-section` returns a *document element* of type `section` with *header* and *elements*.

1.12 make-table (Function)

Syntax:

— Function: **make-table** *description rows*

Arguments and Values:

description—a *rich text* sequence.

rows—a two dimensional list of *rich text* sequences.

Description:

`make-table` returns a *document element* of type `:table` with *description* and *rows*.

1.13 make-url (Function)

Syntax:

— Function: **make-url** *string* `&optional` *url*

Arguments and Values:

string—a *string*.

url—a *string*.

Description:

`make-url` returns a *text token* of type `:url` for *string*. If *url* is given then *string* is used as the label, otherwise *string* is both label and URL.

2 geneva.macros

Macros and reader macros to help with procedural creation of Geneva documents.

2.1 document (Macro)

Syntax:

— Macro: **document** `&rest` *content*

Arguments and Values:

document—*forms* which evaluate to Geneva *elements*.

Description:

`section` returns a Geneva *docuent* with *content* as if by `geneva:make-document`.

Notes:

`(document {content}*)` ≡ `(make-document (list {content}*))`

2.2 listing (Macro)

Syntax:

— Macro: **listing** `&rest` *items*

Arguments and Values:

items—forms which evaluate to Geneva *rich text*.

Description:

`listing` returns a Geneva listing of *items* as if by `geneva:make-listing`.

Notes:

`(listing {items}*)` ≡ `(make-listing (list {items}*))`

2.3 media (Macro)

Syntax:

— Macro: **media** `(&rest` *description*`)` *url*

Arguments and Values:

description—forms which evaluate to Geneva *text tokens*.

url—a *form* which evaluates to a *string* designating an URL.

Description:

`media` returns a Geneva *media element* for *url* with *description* as if by `geneva:make-media`.

Notes:

```
(media ({description}*) {url})
≡ (make-media (list {description}*) {url})
```

2.4 paragraph (Macro)

Syntax:

— Macro: **paragraph** `&rest` *text*

Arguments and Values:

text—forms which evaluate to Geneva *text tokens*.

Description:

`paragraph` returns a Geneva paragraph made up of *text* as if by `geneva:make-paragraph`.

Notes:

`(paragraph {text}*)` ≡ `(make-paragraph (list {text}*))`

2.5 plaintext (Macro)

Syntax:

— Macro: **plaintext** `(&rest` *description*`)` *plaintext*

Arguments and Values:

description—forms which evaluate to Geneva *text tokens*.

plaintext—a *form* which evaluates to a *string*.

Description:

`plaintext` returns a Geneva *plaintext element* for *plaintext* with *description* as if by `geneva:make-plaintext`.

Notes:

```
(plaintext ({description}*) {plaintext})
≡ (make-plaintext (list {description}*) {plaintext})
```

2.6 section (Macro)

Syntax:

— Macro: **section** `(&rest` *header*`) &rest` *content*

Arguments and Values:

header—forms which evaluate to Geneva *text tokens*.

content—forms which evaluate to Geneva *elements*.

Description:

`section` returns a Geneva *section element* with *header* and *content* as if by `geneva:make-section`.

Notes:

```
(section ({header}*) {body}*)
≡ (make-section (list {header}*) (list {body}*))
```

2.7 syntax (Variable)

Initial Value:

```
#<NAMED-READTABLE GENEVA.MACROS:SYNTAX #x302001D4EF3D>
```

Description:

Readtable containing reader macros for markup literals. Defines `#B`, `#I`, `#F` and `#U` to be expanded to code generating Geneva markup at read-time using *make-bold*, *make-italic*, *make-fixed-width* and *make-url* respectively.

Notes:

This readtable is registered as *geneva.macros:syntax*. In order to use it invoke `named-readtable`'s `in-readtable` like so:

```
(in-readtable geneva.macros:syntax)
```

Examples:

```
#b"bold string" ≡ (geneva:make-bold "bold string")
#i"italic string" ≡ (geneva:make-italic "italic string")
#f"fixed-width string" ≡ (geneva:make-fixed-width "fixed-width string")
#u"url string" ≡ (geneva:make-url "url string")
```

See Also:

- Named-Readtables (`editor-hints.named-readtables`)

2.8 table (Macro)

Syntax:

— Macro: **table** (`&rest` *description*) `&rest` *rows*

Arguments and Values:

description—*forms* which evaluate to Geneva *text tokens*.

rows—a list of column lists containing *forms* which evaluate to Geneva *text tokens*.

Description:

`table` returns a Geneva table with *rows* and *description* as if by `geneva:make-table`.

Examples:

```
(table ("10° Celsius in various units.")
 (("Fahrenheit") ((prin1-to-string (+ (* 1.8 10) 32))))
 (("Kelvin") ((prin1-to-string (+ 10 273.15)))))
≡ (make-table (list "10° Celsius in various units.")
              (list (list "Fahrenheit")
                    (list (prin1-to-string (+ (* 1.8 10) 32))))
              (list (list "Kelvin")
                    (list (prin1-to-string (+ 10 273.15)))))
```

3 geneva.mk2

Implementation of *Mk2*[1], a plain text markup language for the Geneva document preparation system.

- 1. *The Mk2 Markup Language* (`mk2.html`)

3.1 character-position (Generic Function)

Syntax:

— Generic Function: **character-position** *syntax-error*

Arguments and Values:

syntax-error—an *error* of type `syntax-error`.

Description:

`character-position` returns a *positive integer* specifying the character position in the line on which *syntax-error* occured.

See Also:

- syntax-error

3.2 line-position (Generic Function)

Syntax:

— Generic Function: **line-position** *syntax-error*

Arguments and Values:

syntax-error—an *error* of type `syntax-error`.

Description:

`line-position` returns a *positive integer* specifying the line of input on which *syntax-error* occured.

See Also:

- syntax-error

3.3 malformed-element (Condition Type)

Class Precedence List:

`malformed-element`, `syntax-error`, `error`, `serious-condition`, `condition`, `standard-object`, `t`

Description:

The *type* `malformed-element` is an error condition of type `syntax-error`. It occurs during parsing a *table*, *media* or *plaintext* element.

See Also:

- syntax-error

3.4 open-section (Condition Type)

Class Precedence List:

`open-section`, `syntax-error`, `error`, `serious-condition`, `condition`, `standard-object`, `t`

Description:

The *type* `open-section` is an error condition of type `syntax-error`. It denotes an unclosed section.

See Also:

- syntax-error

3.5 print-mk2 (Function)

Syntax:

— Function: **print-mk2** *document* `&optional` *stream* `&key` *columns*

Arguments and Values:

document—a Geneva *document*.

stream—a *character stream*. The default is *standard output*.

columns—an *unsigned integer*. The default is 72.

Description:

`print-mk2` writes the *Mk2* representation of *document* to *stream*. `print-mk2` attempts to produce lines no longer than *comlums* in its output.

Exceptional Situations:

If *document* is not a valid Geneva *document* an *error* of *type* `type-error` is signaled.

See Also:

- *The Mk2 markup language* (`mk2.html`)

3.6 read-mk2 (Function)

Syntax:

— Function: **read-mk2** `&optional` *input*

Arguments and Values:

input—a *string* or *character stream*. The default is *standard input*.

Description:

`read-mk2` reads an *Mk2* file from INPUT and returns a *document*.

Exceptional Situations:

If *input* is not a valid *Mk2* file an *error* of *type* `syntax-error` is signaled.

See Also:

- syntax-error
- *The Mk2 markup language* (`mk2.html`)

3.7 syntax-error (Condition Type)

Class Precedence List:

`syntax-error`, `error`, `serious-condition`, `condition`, `standard-object`, `t`

Description:

The *type* `syntax-error` consists of error conditions that occur during `read-mk2`. It denotes a syntax error in the input to `read-mk2`. The functions `line-position` and `character-position` can be used to retrieve the position where the error occurred.

See Also:

- character-position
- line-position

3.8 unrecognized-input (Condition Type)

Class Precedence List:

`unrecognized-input`, `syntax-error`, `error`, `serious-condition`, `condition`, `standard-object`, `t`

Description:

The *type* `unrecognized-input` is an error condition of type `syntax-error`. It denotes that a portion of the input could not be interpreted as *Mk2*.

See Also:

- syntax-error

4 geneva.plain-text

Render Geneva documents as plain text.

4.1 render-plain-text (Function)

Syntax:

— Function: **render-plain-text** *document* `&key` *stream title author date index-p index-caption index-headers-p*

Description:

`render-plain-text` renders *document* as plain text.

See Also:

- *Common Rendering Interface* (`open-geneva.html#section-3-1`)

5 geneva.html

Render Geneva documents as HTML.

5.1 render-html (Function)

Syntax:

— Function: **render-html** *document* `&key` *stream title author date index-p index-caption index-headers-p header-level id-prefix*

Arguments and Values:

header-level—an *unsigned integer*. The default is `0`.

id-prefix—a *string*. The default is `"section"`.

Description:

`render-html` renders *document* as HTML. *header-level* controls the initial headline level. For instance a *header-level* of `1` will cause the top level headlines to be rendered as `H2` elements and so forth. *Id-prefix* is used as a prefix to `NAME` attribute values of HTML anchor elements.

See Also:

- *Common Rendering Interface* (`open-geneva.html#section-3-1`)

5.2 render-html-file (Function)

Syntax:

— Function: **render-html-file** *document* `&key` *stream title author date index-p index-caption index-headers-p stylesheets encoding*

Arguments and Values:

stylesheets—a *list* of stylesheets applicable to `macro-html.widgets:html-widget-document`.

encoding—a *keyword* designating a valid character encoding (defaults to `:utf-8`).

Description:

`render-html-file` renders *document* as a standalone HTML file. The resulting HTML file will use *stylesheets* and declare its content to be in *encoding*.

See Also:

- *Common Rendering Interface* (`open-geneva.html#section-3-1`)

6 geneva.latex

Render Geneva documents as LaTeX manuscripts.

6.1 render-latex (Function)

Syntax:

— Function: **render-latex** *document* `&key` *stream title author date index-p index-caption index-headers-p preamble appendix*

Arguments and Values:

preamble—a *function* without arguments that prints LaTeX expressions to `*standard-output*`. The produced LaTeX expressions will be inserted at the beginning of the LaTeX manuscript.

appendix—a *function* without arguments that prints LaTeX expressions to `*standard-output*`. The produced LaTeX expressions will be appended to the LaTeX manuscript.

Description:

`render-latex` renders *document* as a LaTeX manuscript. *Preamble* and *appendix* may be supplied to customize the LaTeX layout and functionality. Their output will be inserted at the beginning or appended to the end of the LaTeX manuscript respectively.

See Also:

- *Common Rendering Interface* (`open-geneva.html#section-3-1`)

7 geneva.common-lisp

Compile a Geneva *document* from Common Lisp on-line documentation.

7.1 api-document (Function)

Syntax:

— Function: **api-document** `&rest` *packages*

Arguments and Values:

packages—*packages* or *string designators* naming *packages*.

Description:

`api-document` renders the on-line documentation for the *external symbols* of *packages* as a Geneva document.

7.2 symbol-document (Function)

Syntax:

— Function: **symbol-document** *symbol*

Arguments and Values:

symbol—a *symbol*.

Description:

`symbol-document` renders the on-line documentation for *symbol* as a Geneva document.