Package 'highlight'

December 6, 2010

Type Package

Title Syntax highlighter

Version 0.2-5

Date 2010-12-06

Author Romain Francois

Maintainer Romain Francois <romain@r-enthusiasts.com>

Description Syntax highlighter for R code based
on the results of the R parser. Rendering in HTML and latex
markup. Custom sweave driver performing syntax highlighting of R code chunks

License GPL (>= 2)

Depends R (>= 2.12.0), tools, codetools, utils, parser (>= 0.0-10), Rcpp (>= 0.8.9)

LinkingTo Rcpp

URL http://r-forge.r-project.org/projects/highlight/, http://romainfrancois.blog.free.fr/index.php?category/R-package/highlight

R topics documented:

`highlight-package` *Syntax highlighter for R*

Description

Syntax highlighter for R based on output from the R parser

Details

Package:	highlight
Type:	Package
Version:	0.2-5
Date:	2010-12-06
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Romain Francois

Maintainer: Romain Francois <romain@r-enthusiasts.com>

See Also

The main function of the package is `highlight`. The `parser` is a modified R parser using a very similar grammar to the standard `parse` function but presenting the information differently.

`highlight` delegates rendering the document to `renderer`s, such as the `renderer_latex` or `renderer_html` and is helped by a detective to make sense of the results from the parser. The package ships a `dummy_detective` and a `simple_detective`.

The package also defines a custom sweave driver (`HighlightWeaveLatex`) for latex based on the standard sweave latex driver (`RweaveLatex`) using `highlight` to perform syntax highlighting of R code chunks.

Examples

```
## Not run:
tf <- tempfile()
dump( "glm" , file = tf )

# modified R parser
p <- parser( tf )
```

```
# rendering in html
highlight( tf, output = stdout(),
renderer = renderer_html() )

# rendering in latex
highlight( tf, output = stdout(),
renderer = renderer_latex() )

# Sweave driver using syntax highlighting
if( require( grid ) ){
v <- vignette( "grid", package = "grid" )$file
file.copy( v, "grid.Snw" )
Sweave( "grid.Snw", driver= HighlightWeaveLatex() )
system( "pdflatex grid.tex" )
if (.Platform$OS.type == "windows"){
shell.exec( "grid.pdf" )
} else {
system(paste(shQuote(getOption("pdfviewer")), "grid.pdf" ),
wait = FALSE)
}
}

unlink( tf )

## End(Not run)
```

`boxes_latex` *Creates the set of latex boxes*

Description

This function returns the set of latex boxes definitions that should be included in the document preamble. The latex renderer includes these definitions automatically when the document argument is TRUE, but not otherwise.

Usage

```
boxes_latex()
```

Value

A character vector containing latex definitions for boxes used by the latex renderer

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

`translator_latex` translates text into markup that makes use of these boxes.

Examples

```
## Not run:
boxes_latex()

## End(Not run)
```

`css.parser` *Minimal CSS parser*

Description

Minimal CSS (Cascading Style Sheets) parser.

Usage

```
css.parser(file, lines = readLines(file))
```

Arguments

`file`	file to parse
`lines`	lines of text to parse, read from '`file`' by default

Value

A list with one element per style class declaration. Each element is a list which has one element per CSS setting ('`color`', '`background`', ...)

Note

The parser is very minimal and will only identify CSS declarations like the following :

```
.classname{
setting1 : value ;
setting2 : value ;
}
```

The line where a declaration occurs must start with a dot, followed by the name of the class and a left brace. The declaration ends with the first line that starts with a right brace. The function will warn about class names containing numbers as this is likely to cause trouble when the parsed style is translated into another language (e.g. latex commands).

Within the css declaration, the parser identifies setting/value pairs separated by '`:`' on a single line. Each setting must be on a seperate line.

If the setting is '`color`' or '`background`', the parser then tries to map the value to a hex color specification by trying the following options: the value is already a hex color, the name of the color is one of the 16 w3c standard colors (see `http://www.w3schools.com/css/css_colors.asp`), the name is an R color (see `colors`), the color is specified as '`rgb(r,g,b)`'. If all fails, the color used is black for the '`color`' setting and '`white`' for the '`background`' setting.

Other settings are not further parsed at present.

Author(s)

Romain Francois <romain@r-enthusiasts.com>

References

http://www.w3schools.com/css/ http://www.w3schools.com/css/css_colors.asp

Examples

```
## Not run:
cssfile <- system.file( "stylesheet", "default.css", package="highlight")
css.parser( cssfile )

## End(Not run)
```

---

dummy_detective *Incompetent detective*

Description

The simplest possible detective. It simply returns an empty character vector of the appropriate length.

Usage

```
dummy_detective(x, ...)
```

Arguments

x	The result of the parser. The only information this detective uses is the number of tokens.
...	Additional arguments. Ignored.

Value

An empty character vector of the appropriate length, corresponding to the number of tokens to render.

Note

This is mostly here to illustrate the simplest possible detective implementation.

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

Actual detective: simple_detective

Examples

```
## Not run:
tf <- tempfile()
dump( "jitter", file = tf )
highlight( file = tf, detective = dummy_detective,
renderer = renderer_html( document = TRUE ) )

## End(Not run)
```

`formatter_html` *html formatter*

Description

Wraps tokens into span tags with the class corresponding to the style

Usage

```
formatter_html(tokens, styles, ...)
```

Arguments

`tokens`	tokens to wrap
`styles`	styles to give to the tokens
`...`	ignored

Value

A vector of span tags

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

`renderer_html`

Examples

```
## Not run:
f <- formatter_html( )
f( "hello world", "blue")

## End(Not run)
```

formatter_latex *Latex formatter*

Description

Combines tokens and styles into a latex command

Usage

```
formatter_latex(tokens, styles, ...)
```

Arguments

tokens	vector of tokens
styles	vector of styles
...	ignored

Value

A vector of latex commands

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

renderer_latex

Examples

```
## Not run:
formatter_latex( "hello world", "blue" )

## End(Not run)
```

getStyleFile *get a style file*

Description

This function retrieves the path of a style file by looking in various places of the system

Usage

```
getStyleFile(name = "default", extension = "css")
```

Arguments

name	name of the stylesheet, without extension
extension	extension of the stylesheet (e.g. css)

Details

The function looks for a file called 'name.extension' in the following places: the current working directory (see getwd), the user directory used by highlight ('~/.R/highlight'), the 'stylesheet' directory of the installed package.

Value

The first file found is returned. If no file is found, the function returns NULL.

Author(s)

Romain Francois <romain@r-enthusiasts.com>

Examples

```
## Not run:
getwd()
file.path(Sys.getenv("HOME"), ".R", "highlight" )
system.file("stylesheet", package = "highlight")

## End(Not run)
```

---

header_html *html renderer header and footer*

Description

these functions build the header function and the footer function used by the html renderer

Usage

```
header_html(document, stylesheet)
footer_html(document)
```

Arguments

document	logical. If TRUE the built header and footer functions will return the beginning and end of a full html document. If FALSE, the built functions will only return the opening and closing '<pre>' tags.
stylesheet	stylesheet to use. See getStyleFile for details on where the stylesheet can be located.

Value

header and footer functions.

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

renderer_html uses these functions to create a renderer suitable for the 'renderer' argument of highlight

Examples

```
h <- header_html( document = FALSE )
h()
h <- header_html( document = TRUE, stylesheet = "default")
h()
f <- footer_html( document = TRUE )
f()
f <- footer_html( document = FALSE )
f()
```

---

header_latex *latex header and footer*

Description

These functions return appropriate header and footer functions for the latex renderer

Usage

```
header_latex(document, styles, boxes, minipage = FALSE)
footer_latex(document, minipage = FALSE)
```

Arguments

document	logical. If TRUE the header and footer functions will create the full document (including preamble with boxes and styles)
styles	a vector of style definitions to include in the preamble if document is TRUE
boxes	a vector of boxes definitions to include in the preamble if document is TRUE
minipage	if TRUE, the highlighted latex is included in a minipage environment

Value

A function is returned, suitable for the header or footer argument of the latex renderer

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

renderer_latex

Examples

```
## Not run:
h <- header_latex( document = FALSE )
h()
f <- footer_latex( document = FALSE )
f()

## End(Not run)
```

`highlight` *syntax highlighting based on the R parser*

Description

The `highlight` function performs syntax highlighting based on the results of the `parser` and the investigation of a detective.

Usage

```
highlight(file, output = stdout(), detective = simple_detective, renderer,
encoding = "unknown", parser.output = parser(file, encoding = encoding),
styles = detective(parser.output), expr = NULL, final.newline = FALSE,
showPrompts = FALSE, prompt = getOption("prompt"), continue = getOption("continu
initial.spaces = TRUE,
size = c("normalsize", "tiny", "scriptsize",
    "footnotesize", "small", "large", "Large", "LARGE", "huge",
    "Huge"),
    ...)
```

Arguments

`file`	code file to parse. This is only used if the `parser.output` is given
`output`	where to write the rendered text. If this is anything else than the default (standard output), the `sink` function is used to redirect the standard output to the output.
`detective`	the detective chooses the style to apply to each token, basing its investigation on the results of the `parser`
`renderer`	highlight delegates rendering the information to the renderer. This package includes html and latex renderers. See `renderer_html` and `renderer_latex`
`encoding`	encoding to assume for the file. the argument is directly passed to the `parser`.
`parser.output`	output from the `parser`. If this is given, the arguments `file` and `encoding` are not used
`styles`	result of the detective investigation. A character vector with as many elements as there are tokens in the parser output
`expr`	In case we want to render only one expression and not the full parse tree, this argument can be used to specify which expression to render. The default (NULL) means render all expressions. This feature is used by the sweave driver shipped with this package. See `HighlightWeaveLatex`

`final.newline`
: logical. Indicates if a newline character is added after all tokens.

`showPrompts`
: if TRUE, the highlighted text will show standard and continue prompt

`prompt`
: standard prompt

`continue`
: continue prompt

`initial.spaces`
: should initial spaces be displayed or skipped.

`size`
: font size. only respected by the latex renderer so far.

`...`
: additional arguments, currently ignored.

Value

The resulting formatted text is returned invisibly. It is also written to the output if the output is not `NULL`

Note

At the moment, the result is sent to the output by means of a call to `sink`. This is mainly due to the fact that the writing is performed by a C function and the R API does not allow packages to use connections in the C code.

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

`renderer_html` and `renderer_latex` are the two implementation of renderers currently available in this package. The 'xterm256' package defines an additional renderer that uses xterm escape sequences.

`simple_detective` is an example detective which does a very simple investigation.

Examples

```
## Not run:
tf <- tempfile()
dump( "jitter", file = tf )
highlight( file = tf, detective = simple_detective,
renderer = renderer_html( document = TRUE ) )
highlight( file = tf, detective = simple_detective,
renderer = renderer_latex( document = TRUE ) )

# verbatim renderer that upper cases function names
uppercase_formatter <- function( tokens, styles, ...){
out <- tokens
fcalls <- styles == "functioncall"
out[ fcalls ] <- casefold( out[ fcalls ], upper = TRUE )
out
}
uppercase_renderer <- renderer_verbatim(
formatter = uppercase_formatter )
# to debug the formatter, use this :
# debug( uppercase_renderer$formatter )
```

```
# because this will not work:
# debug( uppercase_formatter )
highlight( file = tf, detective = simple_detective,
renderer = uppercase_renderer )


## End(Not run)
```

`highlight.httpd.handler` *httpd handler for highlight*

Description

httpd handler for highlight

Usage

```
highlight.httpd.handler(path, query, body)
```

Arguments

`path`	path of the http request
`query`	ignored
`body`	request body, currently ignored

Details

Since R version 2.11.0, highlight installs a custom handler to handle http request of the prefix "/custom/highlight".

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

highlight is used with the html renderer renderer_html

`HighlightWeaveLatex` *Sweave driver performing syntax highlighting*

Description

Sweave driver using the highlight latex renderer to perform syntax highlighting of input R code in sweave chunks.

Usage

```
HighlightWeaveLatex(
boxes=FALSE, bg = rgb( 0.95,0.95,0.95, max = 1 ), border = "black",
highlight.options = list( boxes = boxes, bg = bg, border = border ) )
```

Arguments

`boxes`	if `TRUE`, code blocks are wrapped in boxes.
`bg`	background color for code boxes.
`border`	color to use for the border of code boxes.
`highlight.options`	Can be used instead of the other arguments to set the `boxes`, `bg` and `border` settings.

Details

This sweave driver is very similar to standard driver that is included in '`utils`'. The difference is that input R code and verbatim output is rendered using `highlight` enabling syntax highlighting of R code.

Instead of using '`Sinput`' and '`Soutput`' commands, this driver uses '`Hinput`' and '`Houtput`' and defines these commands at the very beginning of the document, letting the user the option to overwrite them as necessary.

Latex boxes defined by the latex renderer (`renderer_latex`) and style definitions needed are also written at the beginning of the document.

Because highlight does not use verbatim environments, the user of this driver can freely redefine the '`Hinput`', '`Houtput`' and '`Hchunk`' environments to achieve greater control of the output latex document than with the standard driver.

Value

A sweave driver, suitable for the '`driver`' argument of `Sweave`

Author(s)

Romain Francois <romain@r-enthusiasts.com>

References

Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, Compstat 2002 - Proceedings in Computational Statistics, pages 575-580. Physica Verlag, Heidelberg, 2002. ISBN 3-7908-1517-9

See Also

the standard sweave latex driver: RweaveLatex

Examples

```
## Not run:
# using the driver on the grid vignette
require( grid )
v <- vignette( "grid", package = "grid" )$file
file.copy( v, "grid.Snw" )
Sweave( "grid.Snw", driver= HighlightWeaveLatex() )

## End(Not run)
```

`renderer` *highlight renderer*

Description

The function builds a renderer, suitable for the renderer argument of the highlight function. In the highlight process, renderers are responsible to render the information in the target markup language.

Usage

```
renderer(translator, formatter, space, newline, header, footer, ...)
```

Arguments

`translator`	This argument should be a function with one argument. The translator needs to work token characters so that they display nicely in the target markup language.
`formatter`	The formatter should be a function with at least two arguments: the tokens and the styles. These two arguments are supplied to the formatter by the highlight function. The formatter should wrap tokens and styles into the target markup language. For example, the formatter used by the html renderer makes a '`<span>`' tag of '`class`' given by the '`styles`' and content given by the '`token`'.
`space`	This should be a function with no argument. The output of this function should be a character vector of length one giving the representation of a space character in the target language. For example, in the latex renderer, the function returns '`"{\ }"`'.
`newline`	This should be a function with no argument. The output of the function is a character vector of length one giving the representation of a newline character in the target language.
`header`	This should be a function with no argument. The output of this function is a character vector of arbitrary length. The elements of the output are written before the highlighted content. headers and footers are used to embed the highlighted tokens into some markup. For example, the header used in the html renderer starts a '`<pre>`' tag that is closed by the footer. headers and footer might also be used to write style definitions such as CSS, STY, ...

footer	This should be a function with no argument. The output of this function is written after all tokens.
...	Additional arguments. This might be used to store additional renderer specific objects.

Value

A 'renderer' object. Renderer objects define the interface expected by the highlight function. At the moment, a renderer object is a list of class 'renderer' containing elements: 'translator', 'formatter', 'space', 'newline', 'header' and 'footer'.

Note

Implementations of renderers should call this function to ensure that a proper renderer is created. At the moment, no checking is performed to ensure that the built object complies with the expected interface, but this is very likely to change.

Author(s)

Romain Francois <fr ancoisromain@free.fr>

See Also

The renderer_html implements a renderer using html markup, '' tags and CSS.

The renderer_latex implements a latex renderer.

The renderer_verbatim implements a renderer that does not do anything.

The renderer_xterm function in the xterm256 package implements a renderer based on the 256 color mode of xterm suitable for rendering the information in an xterm console.

Examples

```
## Not run:
# html renderer which will render the content as a document
# (starting with <html> tags
r <- renderer_html( document = T )

# html renderer suitable to render the information within
# an already built document
r <- renderer_html( document = F )

## End(Not run)
```

---

renderer_html *html renderer using span tags and CSS*

Description

implementation of the renderer that renders the information as a series of '' html tags

Usage

```
renderer_html(document = TRUE, translator = translator_html,
formatter = formatter_html, space = space_html,
newline = newline_html, header = header_html(document, stylesheet),
footer = footer_html(document),
stylesheet = "default" , ...)
translator_html( x, size )
space_html()
newline_html()
```

Arguments

document	logical. Indicates if the renderer should render a full document or simply a '<pre>' section containing the highlighted tokens. This argument is used by the header_html and footer_html to build appropriate header and footer.
translator	Since the highlighted tokens are wrapped in a '<pre>' tag, no further translation is needed.
formatter	html formatter. creates '' tags for all tokens. See formatter_html
space	returns a space character
newline	returns a newline character
header	html header. Depending on the 'document' argument, this will be a function building a the beginning of a complete html document (starting with '<html>') including css definitions or simply a function returning '<pre>' enabling the renderer to be used to just render the syntax as part of a bigger document.
footer	html footer. Depending on the 'document' argument, this will either close the full document (close the '</html>' tag) or simply close the '</pre>' tag.
stylesheet	stylesheet to use. This is used by the header when document is TRUE. The content of the stylesheet is copied verbatim into a '<style>' tag in that case. See getStyleFile for details on where the stylesheet can be located
x	argument to the translator. Returned as is.
size	font size. ignored
...	Additional arguments. unused.

Value

A renderer capable suitable for the 'renderer' argument of highlight

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

renderer for a description of the interface this renderer is implementing.

highlight takes a renderer argument to which it delegates rendering.

Examples

```
## Not run:
r <- renderer_html()

## End(Not run)
```

`renderer_latex` *LaTeX renderer*

Description

renderer implementation targetting latex markup. The result markup uses the latex 'alltt' package to achieve true type renderering and therefore does not depend on verbatim-like environments.

Usage

```
renderer_latex(document = TRUE, boxes = boxes_latex(),
translator = translator_latex, formatter = formatter_latex,
space = space_latex, newline = newline_latex,
stylesheet = "default",
styles = styler(stylesheet, "sty", styler_assistant_latex),
header = header_latex(document, styles = styles, boxes = boxes, minipage = minip
footer = footer_latex(document, minipage = minipage),
minipage = FALSE, ...)
newline_latex()
space_latex()
```

Arguments

`document`	logical. Should the renderer create the full document or only the code section, assuming the document is already created. Using FALSE is used by the sweave driver shipped with this package.
`boxes`	a function that returns definitions of latex boxes used for non standard characters. The reason for using boxes is that some character need to be escaped to be rendered, and unfortunately, escaping turns alltt off, which does not produce satisfying rendering. This argument is used by the header function when the document argument is TRUE. It is also used in the sweave driver at the very beginning of the document
`translator`	translation of characters into latex markup. See `translator_latex` for details
`formatter`	latex formatter. Tokens are wrapped into a latex command related to the style they should honor.
`space`	returns a space character that does not get reduced by latex
`newline`	returns a newline character
`stylesheet`	stylesheet to use.
`styles`	style definitions inferred from the parsing of the stylesheet. See `styler` and `styler_assistant_latex`.

header	returns the header. If the document argument is TRUE, the header contains the style definitions and the boxes definitions. If it is FALSE, a minimal header is produced to turn alltt on. In the latter case, boxes and style definitions are assumed to have been inserted already, latex will not compile the document otherwise.
footer	returns the footer. Depending on the document argument, either a minimal footer is produced (turning off alltt) or the full latex document is closed.
minipage	if TRUE, the highlighted latex is included in a minipage environment
...	Additional arguments

Value

a 'renderer' object, suitable for the 'renderer' argument of highlight.

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

the sweave driver HighlightWeaveLatex makes extensive use if this renderer.

Examples

```
## Not run:
r <- renderer_latex(document = T )
r$space()
r$newline()
r$boxes()
r$translator( "# the hash symbol gets a latex box" )

## End(Not run)
```

---

renderer_verbatim *A dummy renderer*

Description

The most simple renderer. It simply writes tokens as they are on the input file

Usage

```
renderer_verbatim(translator = translator_verbatim,
formatter = formatter_verbatim, space = space_verbatim,
newline = newline_verbatim, header = header_verbatim,
footer = footer_verbatim, ...)
translator_verbatim(x, size)
formatter_verbatim( tokens, styles, ...)
space_verbatim()
newline_verbatim()
header_verbatim()
footer_verbatim()
```

Arguments

translator	Dummy translator. Returns the input as it is.
formatter	Dummy formatter. Returns the tokens as they are, the styles argument is not used.
space	return a space character
newline	return a newline character
header	return an empty character vector
footer	return a newline character
...	Additional ignored arguments.
x	The input of the translator, returned as is
tokens	The tokens to format. They are returned as is by the formatter
styles	The styles to use to format the tokens. They are not used
size	font size. ignored

Value

A 'renderer' object.

Author(s)

Romain Francois <romain@r-enthusiasts.com>

Examples

```
## Not run:
r <- renderer_verbatim()
r$space()
r$header()
r$translator( "rnorm" )
r$formatter( "rnorm", "blue")

## End(Not run)
```

---

simple_detective *Simple detective*

Description

This detective only uses semantic information to make its investigation.

Usage

```
simple_detective(x, ...)
```

Arguments

x	output of the parser. The detective is only interested in the 'token.desc' column of the data.
...	Ignored

Value

a vector of styles grouping similar tokens together

Note

the '"default"' stylesheet is well suited to work with this detective

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

dummy_detective

Examples

```
## Not run:
p <- parser( text = deparse( jitter ) )
simple_detective( p )

## End(Not run)
```

---

styler *Style definition generator*

Description

This generates style definitions either by including a language specific style file (e.g. sty file for latex) or by parsing a css stylesheet

Usage

```
styler(stylesheet, extension = "css", assistant)
```

Arguments

stylesheet	name of the stylesheet.
extension	extension of the language specific format for the stylesheet.
assistant	function to which the styler delegates understanding of the parser output

Details

First, the function attempts to retrieve a language specific stylesheet using the getStyleFile function. If a language specific stylesheet is found, it returns the content of the file as a character vector.

Second, the function attemps to find a css stylesheet using getStyleFile, parse the css declarations using the css.parser function, and delegates to the 'assistant' which is responsible to translate the results of the css parser into language specific declarations.

Value

a character vector containing style declarations in the target language

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

`styler_assistant_latex` gives a concrete implementation of the assistant for the latex language

Examples

```
## Not run:
styler( "default", "sty", styler_assistant_latex )

## End(Not run)
```

`styler_assistant_latex` *latex styler assistant*

Description

This function takes the output of the `css.parser` and produces latex style definitions from it.

Usage

```
styler_assistant_latex(x)
```

Arguments

`x`	output from `css.parser`

Details

The function create a new latex command for each css declaration, i.e. each item of the list '`x`' it is passed.

The assistant currently honours the following css settings: color, '`text-decoration:underline`', '`font-weight:bold[er]`' and '`font-style:italic`'

Value

a vector of latex style definitions corresponding to (a subset of) the output of the parser

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

`styler`

Examples

```
## Not run:
styler( "default", "sty", styler_assistant_latex )
css.out <- css.parser( getStyleFile( "default" ) )
styler_assistant_latex( css.out )

## End(Not run)
```

---

translator_latex *LaTeX translator*

Description

This function translates character vectors so that they nicely print in LaTeX. In particular this uses latex boxes.

Usage

```
translator_latex(x, size = c("normalsize", "tiny", "scriptsize",
    "footnotesize", "small", "large", "Large", "LARGE", "huge",
    "Huge") )
```

Arguments

x	text to translate
size	font size

Value

translated text

Author(s)

Romain Francois <romain@r-enthusiasts.com>

See Also

the latex renderer: renderer_latex uses this translator.

Examples

```
## Not run:
# the code is probably the best description of what this does
translator_latex
translator_latex( "#<" )

## End(Not run)
```

Index