Package 'htmltidy'

October 16, 2016

Title Tidy Up and Test XPath Queries on HTML and XML Content

Version 0.3.0

Maintainer Bob Rudis <bob@rud.is>

Description HTML documents can be beautiful and pristine. They can also be wretched, evil, malformed demon-spawn. Now, you can tidy up that HTML and XHTML before processing it with your favorite angle-bracket crunching tools, going beyond the limited tidying that 'libxml2' affords in the 'XML' and 'xml2' packages and taming even the ugliest HTML code generated by the likes of Google Docs and Microsoft Word. It's also possible to use the functions provided to format or ``pretty print'' HTML content as it is being tidied. Utilities are also included that make it possible to view formatted and ``pretty printed'' HTML/XML content from HTML/XML document objects, nodes, node sets and plain character HTML/XML using 'vkbeautify' (by Vadim Kiryukhin) and 'highlight.js' (by Ivan Sagalaev). Also (optionally) enables filtering of nodes via XPath or viewing an HTML/XML document in ``tree'' view using 'xml-viewer' (by Julian Gruber). See <https://github.com/vkiryukhin/vkBeautify> and <https://github.com/juliangruber/xml-viewer> for more information about 'vkbeautify' and 'xml-viewer', respectively.

Copyright file inst/COPYRIGHTS

URL https://github.com/hrbrmstr/htmltidy

BugReports https://github.com/hrbrmstr/htmltidy/issues

Depends R (>= 3.2.0)

License MIT + file LICENSE

LazyData true

Encoding UTF-8

NeedsCompilation yes

Suggests testthat, httr, rvest

LinkingTo Rcpp

Imports Rcpp, xml2, XML, htmlwidgets, htmltools

RoxygenNote 5.0.1

Author Bob Rudis [aut, cre],
Dave Raggett [ctb, aut] (Original HTML Tidy library),
Charles Reitzel [ctb, aut] (Modern HTML Tidy library),
Björn Höhrmann [ctb, aut] (HTML5 Support),
Kenton Russell [aut, ctb] (xml-viewer integration),
Vadim Kiryukhin [ctb, cph] (vkbeautify library),
Ivan Sagalaev [ctb, cph] (highlight.js library),
Julian Gruber [ctb, cph] (xml-viewer library)

Repository CRAN

Date/Publication 2016-10-16 17:39:11

R topics documented:

`highlight_styles` *List available HTML/XML highlight styles*

Description

Returns a character vector of available style sheets to use when displaying an XML document.

Usage

```
highlight_styles()
```

References

See https://highlightjs.org/static/demo/ for a demo of all highlight.js styles

Examples

```
highlight_styles()
```

htmltidy *Tidy Up and Test XPath Queries on HTML and XML Content*

Description

HTML documents can be beautiful and pristine. They can also be wretched, evil, malformed demon-spawn. Now, you can tidy up that HTML and XHTML before processing it with your favorite angle-bracket crunching tools, going beyond the limited tidying that 'libxml2' affords in the 'XML' and 'xml2' packages and taming even the ugliest HTML code generated by the likes of Google Docs and Microsoft Word. It's also possible to use the functions provided to format or "pretty print" HTML content as it is being tidied. Utilities are also included that make it possible to view formatted and "pretty printed" HTML/XML content from HTML/XML document objects, nodes, node sets and plain character HTML/XML using 'vkbeautify' (by Vadim Kiryukhin) and 'highlight.js' (by Ivan Sagalaev). Also (optionally) enables filtering of nodes via XPath or viewing an XML document in "tree" view using 'xml-viewer' (by Julian Gruber). See https://github.com/vkiryukhin/vkBeautify and https://github.com/juliangruber/xml-viewer for more information about 'vkbeautify' and 'xml-viewer', respectively.

Author(s)

Bob Rudis (bob@rud.is)

renderXmlview *Widget render function for use in Shiny*

Description

Widget render function for use in Shiny

Usage

```
renderXmlview(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

expr	expr
env	env
quoted	quoted

`tidy_html.response` *Tidy or "Pretty Print" HTML/XHTML Documents*

Description

Pass in HTML content as either plain or raw text or parsed objects (either with the `XML` or `xml2` packages) or as an `httr response` object along with an options list that specifies how the content will be tidied and get back tidied content of the same object type as passed in to the function.

Usage

```
## S3 method for class 'response'
tidy_html(content, options = list(TidyXhtmlOut = TRUE),
  verbose = FALSE)

tidy_html(content, options = list(TidyXhtmlOut = TRUE), verbose = FALSE)

## Default S3 method:
tidy_html(content, options = list(TidyXhtmlOut = TRUE),
  verbose = FALSE)

## S3 method for class 'character'
tidy_html(content, options = list(TidyXhtmlOut = TRUE),
  verbose = FALSE)

## S3 method for class 'raw'
tidy_html(content, options = list(TidyXhtmlOut = TRUE),
  verbose = FALSE)

## S3 method for class 'xml_document'
tidy_html(content, options = list(TidyXhtmlOut = TRUE),
  verbose = FALSE)

## S3 method for class 'HTMLInternalDocument'
tidy_html(content, options = list(TidyXhtmlOut
  = TRUE), verbose = FALSE)

## S3 method for class 'connection'
tidy_html(content, options = list(TidyXhtmlOut = TRUE),
  verbose = FALSE)
```

Arguments

`content`	accepts a character vector, raw vector or parsed content from the `xml2` or `XML` packages.
`options`	named list of options
`verbose`	output document errors? (default: `FALSE`)

Details

The default option `TixyXhtmlOut` will convert the input content to XHTML.

Currently supported options:

- Ones taking a logical value: `TidyAltText`, `TidyBodyOnly`, `TidyBreakBeforeBR`, `TidyCoerceEndTags`, `TidyDropEmptyElems`, `TidyDropEmptyParas`, `TidyFixBackslash`, `TidyFixComments`, `TidyGDocClean`, `TidyHideComments`, `TidyHtmlOut`, `TidyIndentContent`, `TidyJoinClasses`, `TidyJoinStyles`, `TidyLogicalEmphasis`, `TidyMakeBare`, `TidyMakeClean`, `TidyMark`, `TidyOmitOptionalTags`, `TidyReplaceColor`, `TidyUpperCaseAttrs`, `TidyUpperCaseTags`, `TidyWord2000`, `TidyXhtmlOut`
- Ones taking a character value: `TidyDoctype`, `TidyInlineTags`, `TidyBlockTags`, `TidyEmptyTags`, `TidyPreTags`
- Ones taking an integer value: `TidyIndentSpaces`, `TidyTabSize`, `TidyWrapLen`

File an issue if there are other `libtidy` options you'd like supported.

It is likely that the most used options will be:

- `TidyXhtmlOut` (logical),
- `TidyHtmlOut` (logical) and
- `TidyDocType` which should be one of "`omit`", "`html5`", "`auto`", "`strict`" or "`loose`".

You can clean up Microsoft Word (2000) and Google Docs HTML via logical settings for `TidyWord2000` and `TidyGDocClean`, respectively.

It may also be advantageous to remove all comments with `TidyHideComments`.

Value

Tidied HTML/XHTML content. The object type will be the same as that of the input type except when it is a `connection`, then a character vector will be returned.

Note

If document parsing errors are severe enough, `tidy_html()` will not be able to clean the document and will display the errors (this output can be captured with `sink()` or `capture.output()`) along with a warning and return a "best effort" cleaned version of the document.

References

http://api.html-tidy.org/tidy/quickref_5.1.25.html & https://github.com/htacg/tidy-html5/blob/master/include/tidyenum.h for definitions of the options supported above and https://www.w3.org/People/Raggett/tidy/ for an explanation of what "tidy" HTML is and some canonical examples of what it can do.

Examples

```
opts <- list(
  TidyDocType="html5",
  TidyMakeClean=TRUE,
  TidyHideComments=TRUE,
  TidyIndentContent=TRUE,
  TidyWrapLen=200
)

txt <- paste0(
  c("<html><head><style>p { color: red; }</style><body><!-- ===== body ====== -->",
"<p>Test</p></body><!--Default Zone --> <!--Default Zone End--></html>"),
  collapse="")

cat(tidy_html(txt, option=opts))

library(httr)
res <- GET("http://rud.is/test/untidy.html")

# look at the original, un-tidy source
cat(content(res, as="text", encoding="UTF-8"))

# see the tidied version
cat(tidy_html(content(res, as="text", encoding="UTF-8"),
              list(TidyDocType="html5", TidyWrapLen=200)))

# but, you could also just do:
cat(tidy_html(url("http://rud.is/test/untidy.html")))
```

`xmltreeview-shiny` *Shiny bindings for xmltreeview*

Description

Output and render functions for using xmltreeview within Shiny applications and interactive Rmd documents.

Usage

```
xmltreeviewOutput(outputId, width = "100%", height = "400px")

renderXmltreeview(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	output variable to read from
`width, height`	Must be a valid CSS unit (like \code'100%', \code'400px', \code'auto') or a number, which will be coerced to a string and have \code'px' appended.

expr	An expression that generates a xmltreeview
env	The environment in which to evaluate \codeexpr.
quoted	Is \codeexpr a quoted expression (with \codequote())? This is useful if you want to save an expression in a variable.

`xmlviewOutput` *Widget output function for use in Shiny*

Description

Widget output function for use in Shiny

Usage

```
xmlviewOutput(outputId, width = "100%", height = "400px")
```

Arguments

outputId	outputId
width	width
height	height

`xml_tree_view` *HTML/XML tree viewer*

Description

This uses the `xml-viewer` JavaScript module to provide a simple collapsible tree viewer for HTML/XML documents, nodes, node sets and plain character HTML/XML in an `htmlwidget` pane.

Usage

```
xml_tree_view(doc = NULL, scroll = FALSE, elementId = NULL,
  width = "100%", height = NULL)

html_tree_view(doc = NULL, scroll = FALSE, elementId = NULL,
  width = "100%", height = NULL)
```

Arguments

doc	xml2 document/node/nodeset, an HTMLInternalDocument/ XMLInternalDocument or atomic character vector of HTML/XML content
scroll	should the holding the HTML/XML content scroll (TRUE) or take up the full viewer/browser window (FALSE). Default is FALSE (take up the full viewer/browser window). If this is set to TRUE, height should be set to a value other than NULL.
elementId	element id
width	widget div width
height	widget div height

Note

Large HTML or XML content may take some time to render properly. It is suggested that this function be used on as minimal of a subset of HTML/XML as possible or used in a browser context vs an IDE viewer context.

References

xml-viewer

Examples

```
if (interactive()) {

# from ?xml2::read_xml
cd <- xml2::read_xml("http://www.xmlfiles.com/examples/cd_catalog.xml")

xml_tree_view(cd)

htmltools::browsable(
  htmltools::tagList(
    xml_tree_view(cd, width = "100%", height = "300px"),
    xml_view(cd)
  )
)
}
```

---

xml_view *HTML/XML pretty printer and viewer*

Description

This uses the vkbeautify and highlight.js javascript modules to format and "pretty print" HTML/XML documents, nodes, node sets and plain character HTML/XML in an htmlwidget pane.

Usage

```
xml_view(doc, style = "default", scroll = FALSE, add_filter = FALSE,
  apply_xpath = NULL, elementId = NULL, width = "100%", height = NULL)

html_view(doc, style = "default", scroll = FALSE, add_filter = FALSE,
  apply_xpath = NULL, elementId = NULL, width = "100%", height = NULL)
```

Arguments

doc	xml2 document/node/nodeset, an HTMLInternalDocument/ XMLInternalDocument or atomic character vector of HTML/XML content
style	CSS stylesheet to use (see higlight_styles())
scroll	should the holding the HTML/XML content scroll (TRUE) or take up the full viewer/browser window (FALSE). Default is FALSE (take up the full viewer/browser window). If this is set to TRUE, height should be set to a value other than NULL.
add_filter	show an XPath input box to enable live filtering? (default: FALSE)
apply_xpath	Add and apply an XPath query string to the view. If add_filter is TRUE then this query string will appear in the filter box and be applied to the passed in document.
elementId	element id
width	widget width (best to keep it at 100%)
height	widget height (kinda only useful for knitting since this is meant to be an interactive tool).

Note

Large HTML or XML content may take some time to render properly. It is suggested that this function be used on as minimal of a subset of HTML/XML as possible or used in a browser context vs an IDE viewer context.

References

highlight.js, vkbeautify

Examples

```
if (interactive()) {
library(xml2)

# plain text
txt <- paste0("<note><to>Tove</to><from>Jani</from><heading>Reminder</heading>",
              "<body>Don't forget me this weekend!</body></note>")
xml_view(txt)

# xml object
doc <- read_xml(txt)
xml_view(doc, style="obsidian")
```

```
# different style
xml_view(xml_find_all(doc, ".//to"), style="github-gist")

# some more complex daata
xml_view(read_xml(system.file("extdata/dwml.xml", package="htmltidy")))
xml_view(read_xml(system.file("extdata/getHistory.xml", package="htmltidy")),
         "androidstudio")
xml_view(read_xml(system.file("extdata/input.xml", package="htmltidy")),
         "sunburst")

# filter + apply an initial XPath query string
xml_view(read_xml(system.file("extdata/dwml.xml", package="xmlview")),
         add_filter=TRUE, apply_xpath=".//temperature")

doc <- read_xml("http://www.npr.org/rss/rss.php?id=1001")

str(doc)

xml_view(doc, add_filter=TRUE)
xml2::xml_find_all(doc, './/dc:creator', ns=xml2::xml_ns(doc))

xml_text(xml2::xml_find_all(doc, './/link[contains(., "soccer")]', ns=xml2::xml_ns(doc)))
}
```

Index