Package: html2pdfr (via r-universe)

September 21, 2024

R6GeneratorNote Generated by r6-generator-maven-plugin: remove this line if you want to make manual changes and dont want them to get overwritten

Type Package

Title R Wrapper For Openhtmltopdf Java Library

Version 0.4.5

Description HTML and CSS do a good job at automatically laying out and styling content particularly in tables, however it is not natively designed for pagination. This library converts HTML content into PDF and PNG formats for embedding into LaTeX documents, within the constraints of page sizes. It allows use of HTML table layout from HTML first libraries such as 'gt' and 'huxtable' within latex documents. It allows HTML content to grow in width up to the page dimensions, but preventing it from overflowing, and without forcing table layout to be wider than it would normally be. This heurisitic calculation of the output size up to fit within set limits is one of the differentiators between this and other HTML to PDF converters. Although the focus is on tables, any basic HTML content can be rendered, including simple SVG and MathML, up to the support of the underlying HTML rendering engine (https://github.com/danfickle/openhtmltopdf).

License LGPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Suggests here, readr, huxtable, dplyr, knitr, rmarkdown, testthat

Imports systemfonts, rJava, R6, fs, rappdirs, utils, magrittr, rlang, xml2, stringr

URL `https://terminological.github.io/html2pdfr/docs/index.html`, `https://github.com/terminological/html2pdfr`

BugReports `https://github.com/terminological/html2pdfr/issues`

RoxygenNote 7.3.1

Repository https://terminological.r-universe.dev

RemoteUrl https://github.com/terminological/html2pdfr

RemoteRef 0.4.5

RemoteSha 65dcb2445ca2cbad0044a375014dcafcd6b94c6d

Contents

`html2pdfr-package` *html2pdfr: R Wrapper For Openhtmltopdf Java Library*

Description

HTML and CSS do a good job at automatically laying out and styling content particularly in tables, however it is not natively designed for pagination. This library converts HTML content into PDF and PNG formats for embedding into LaTeX documents, within the constraints of page sizes. It allows use of HTML table layout from HTML first libraries such as 'gt' and 'huxtable' within latex documents. It allows HTML content to grow in width up to the page dimensions, but preventing it from overflowing, and without forcing table layout to be wider than it would normally be. This heurisitic calculation of the output size up to fit within set limits is one of the differentiators between this and other HTML to PDF converters. Although the focus is on tables, any basic HTML content can be rendered, including simple SVG and MathML, up to the support of the underlying HTML rendering engine (https://github.com/danfickle/openhtmltopdf).

Version: 0.4.5

Classes:

- `JavaApi`
- `HtmlConverter`

Author(s)

- Rob Challen rob.challen@bristol.ac.uk 0000-0002-5504-7768

`file_to_pdf` *Convert HTML document from a local file to a PDF document.*

Description

The HTML in ‘inFile‘ is assumed to be a complete document. Relative references are resolved with reference to the HTML file on the file system, so correctly located images etc whould be picked up without requiring a server. The resulting PDF size will be controlled by page media directives within the HTML, unless explicitly given here in ‘maxWidthInches‘ and ‘maxHeightInches‘. If the ‘cssSelector‘ parameter is given the HTML fragment at that selector will be used. In this case it is will be resized to fit within the given dimensions and shrink wrapped so that the content is smaller. If no dimensions are present this will default to A4.

Usage

```
file_to_pdf(
 inFile,
 outFile,
 cssSelector,
 xMarginInches,
 yMarginInches,
 maxWidthInches,
 maxHeightInches,
 formats,
 pngDpi,
 converter
)
```

Arguments

`inFile`	inFile the full path the the HTML file - (java expects a RCharacter)
`outFile`	outFile the full path of the output file - (defaulting to ‘tempfile(’html2pdfr_’)‘) - (java expects a RFile)
`cssSelector`	cssSelector the part of the page you want to convert to PDF. - (defaulting to ‘NA_character_‘) - (java expects a RCharacter)
`xMarginInches`	xMarginInches page width margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`yMarginInches`	yMarginInches page height margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`maxWidthInches`	maxWidthInches what is the maximum allowable width? - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`maxHeightInches`	maxHeightInches what is the maximum allowable height? (if the content is larger than this then it will overflow to another page) - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)

`formats`	formats If the outFile does not specify a file extension then you can do so here as "png" or "pdf" or both. - (defaulting to ‘c('pdf')‘) - (java expects a RCharacterVector)
`pngDpi`	pngDpi the dots per inch for png outputs if requested - (defaulting to ‘300‘) - (java expects a RNumeric)
`converter`	converter (optional) a configured HTML converter, only needed if manually specifying fonts. - (defaulting to ‘html2pdfr::html_converter()‘) - (java expects a HtmlConverter)

Value

RCharacterVector: the filename written to (with extension '.pdf' or '.png' if outFile did not have an extension).

Examples

```
library(testthat)
dest = tempfile(fileext='.html')
download.file('https://cran.r-project.org/banner.shtml',
 destfile = dest)
file_to_pdf(dest)
```

`HtmlConverter` *HtmlConverter*

Description

missing description

Version: 0.4.5

Generated: 2024-04-24T15:58:01.271789527

Arguments

`fontfiles`	fontfiles - (java expects a String)

Details

no details

Methods

Constructors:

- `J$HtmlConverter$new(fontfiles)`

Class methods:

- none
- `instance$clone()`

- `instance$print()`

Constructor `new()`**:** the default no-args constructor

Usage:

```
J = html2pdfr::JavaApi$get()
instance = J$HtmlConverter$new(fontfiles);
```

Arguments:

- fontfiles - (java expects a String)

Returns: the new R6 HtmlConverter object

Examples

```
## ----------------------------------
## Construct new instance of HtmlConverter
## ----------------------------------
## Not run:
J = html2pdfr::JavaApi$get()
# appropriate parameter values must be provided
instance = J$HtmlConverter$new(fontfiles)

## End(Not run)
```

`html_converter` *Create a new HtmlConverter*

Description

for creating PDF and PNG files from HTML. In general this will be created automatically. but if you have specific fonts you want to use then you may need to pass them to this function and specify the result in the ‘converter‘ parameter of the main functions.

Usage

```
html_converter(
 fontfiles,
 update
)
```

Arguments

`fontfiles`	fontfiles - a character vector of font files that will be imported into the converter. - (defaulting to ‘systemfonts::system_fonts()$path‘) - (java expects a RCharacterVector)
`update`	update - (defaulting to ‘FALSE‘) - (java expects a RLogical)

Value

R6 HtmlConverter object:

Examples

```
conv = html2pdfr::html_converter()
```

`html_document_to_pdf` *Convert HTML document from a string to a PDF document.*

Description

The HTML in ‘html‘ is assumed to be a complete document. Relative references are resolved with reference to ‘baseUri‘ if it is given (which could be a ‘file://‘ URI). The resulting PDF size will be controlled by page media directives within the HTML, unless explicitly given here in ‘maxWidthInches‘ and ‘maxHeightInches‘. If the ‘cssSelector‘ parameter is given the HTML fragment at that selector will be used. In this case it is will be resized to fit within the given dimensions and shrink wrapped so that the content is smaller. If no dimensions are present this will default to A4.

Usage

```
html_document_to_pdf(
 html,
 outFile,
 baseUri,
 cssSelector,
 xMarginInches,
 yMarginInches,
 maxWidthInches,
 maxHeightInches,
 formats,
 pngDpi,
 converter
)
```

Arguments

`html`	html the html document as a string - (java expects a RCharacter)
`outFile`	outFile the full path of the output file - (defaulting to ‘tempfile('html2pdfr_')‘) - (java expects a RFile)
`baseUri`	baseUri the URI from which to interpret relative links in the html content. - (defaulting to ‘NA_character_‘) - (java expects a RCharacter)
`cssSelector`	cssSelector the part of the page you want to convert to PDF. - (defaulting to ‘NA_character_‘) - (java expects a RCharacter)

`xMarginInches`	xMarginInches page width margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`yMarginInches`	yMarginInches page height margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`maxWidthInches`	maxWidthInches what is the maximum allowable width? - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`maxHeightInches`	maxHeightInches what is the maximum allowable height? (if the content is larger than this then it will overflow to another page) - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`formats`	formats If the outFile does not specify a file extension then you can do so here as "png" or "pdf" or both. - (defaulting to ‘c(’pdf’)‘) - (java expects a RCharacterVector)
`pngDpi`	pngDpi the dots per inch for png outputs if requested - (defaulting to ‘300‘) - (java expects a RNumeric)
`converter`	converter (optional) a configured HTML converter, only needed if manually specifying fonts. - (defaulting to ‘html2pdfr::html_converter()‘) - (java expects a HtmlConverter)

Value

RCharacterVector: the filename written to (with extension ’.pdf’ or ’.png’ if outFile did not have an extension).

Examples

```
library(testthat)
library(readr)
html = read_file('https://fred-wang.github.io/MathFonts/mozilla_mathml_test/')
html_document_to_pdf(html, baseUri = 'https://fred-wang.github.io/MathFonts/mozilla_mathml_test/')
```

`html_fragment_to_pdf` *Render HTML fragment from a string to a PDF image.*

Description

This is the simple rendering function that will output a PDF file (potentially many pages) and a set of PNG files from HTML content. This is primarily used to render HTML content (e.g. a table) that is being included in a larger document. In this case the HTML fragment will not specify page dimensions which need to be provided (defaults to A4 size with 1 inch margins). The result can be embedded into an existing page using latex’s includegraphics directive exactly the same way as a graphical figure might be used. The sizing of the output will always be smaller than the dimensions of a page, but will shrink to fit the content.

Usage

```
html_fragment_to_pdf(
 htmlFragment,
 outFile,
 xMarginInches,
 yMarginInches,
 maxWidthInches,
 maxHeightInches,
 formats,
 pngDpi,
 converter
)
```

Arguments

`htmlFragment`	htmlFragment a HTML fragment, e.g. usually the table element, but may be the whole page. - (java expects a RCharacter)
`outFile`	outFile the full path with or without extension (if no extension specified then ‘formats‘ parameter will apply) - (defaulting to ‘tempfile('html2pdfr_')‘) - (java expects a RFile)
`xMarginInches`	xMarginInches page width margins - (defaulting to ‘1.0‘) - (java expects a RNumeric)
`yMarginInches`	yMarginInches page height margins - (defaulting to ‘1.0‘) - (java expects a RNumeric)
`maxWidthInches`	maxWidthInches what is the maximum allowable width? (default is A4) - (defaulting to ‘8.27‘) - (java expects a RNumeric)
`maxHeightInches`	maxHeightInches what is the maximum allowable height? (if the content is larger than this then it will overflow to another page) - (defaulting to ‘11.69‘) - (java expects a RNumeric)
`formats`	formats If the outFile does not specify a file extension then you can do so here as "png" or "pdf" or both. - (defaulting to ‘c('pdf','png')‘) - (java expects a RCharacterVector)
`pngDpi`	pngDpi the dots per inch for png outputs if requested. - (defaulting to ‘300‘) - (java expects a RNumeric)
`converter`	converter (optional) a configured HTML converter, only needed if manually specifying fonts. - (defaulting to ‘html2pdfr::html_converter()‘) - (java expects a HtmlConverter)

Value

RCharacterVector: the filename(s) written to (with extension '.pdf' or '.png' if outFile did not have an extension).

Examples

```
library(testthat)
library(dplyr)
html = iris %>% group_by(Species) %>%
summarise(across(everything(), mean)) %>%
huxtable::as_hux() %>%
huxtable::theme_article() %>%
huxtable::to_html()
html_fragment_to_pdf(html)
```

`JavaApi` *R Wrapper For Openhtmltopdf Java Library*

Description

HTML and CSS do a good job at automatically laying out and styling content particularly in tables, however it is not natively designed for pagination. This library converts HTML content into PDF and PNG formats for embedding into LaTeX documents, within the constraints of page sizes. It allows use of HTML table layout from HTML first libraries such as 'gt' and 'huxtable' within latex documents. It allows HTML content to grow in width up to the page dimensions, but preventing it from overflowing, and without forcing table layout to be wider than it would normally be. This heurisitic calculation of the output size up to fit within set limits is one of the differentiators between this and other HTML to PDF converters. Although the focus is on tables, any basic HTML content can be rendered, including simple SVG and MathML, up to the support of the underlying HTML rendering engine (https://github.com/danfickle/openhtmltopdf).

Version: 0.4.5

Generated: 2024-04-24T15:58:01.201546101

Arguments

`logLevel`	optional - the slf4j log level as a string - one of OFF (most specific, no logging), FATAL (most specific, little data), ERROR, WARN, INFO, DEBUG, TRACE (least specific, a lot of data), ALL (least specific, all data)

Usage

```
J = html2pdfr::JavaApi$get(logLevel)
```

Package initialisation and control

- `JavaApi$installDependencies()`
- `JavaApi$rebuildDependencies()`
- `JavaApi$versionInformation()`
- `J = JavaApi$get(logLevel)`
- `J$changeLogLevel(logLevel)`

- `J$reconfigureLog(log4jproperties)`
- `J$printMessages()`

Package classes and static methods

- `J$HtmlConverter$new(fontfiles)`
- `J$HtmlConverter$htmlConverter(fontfiles, update)`
- `J$HtmlConverter$urlToPdf(htmlUrl, outFile, cssSelector, xMarginInches, yMarginInches, maxWidthInches, maxHeightInches, formats, pngDpi, converter)`
- `J$HtmlConverter$fileToPdf(inFile, outFile, cssSelector, xMarginInches, yMarginInches, maxWidthInches, maxHeightInches, formats, pngDpi, converter)`
- `J$HtmlConverter$htmlDocumentToPdf(html, outFile, baseUri, cssSelector, xMarginInches, yMarginInches, maxWidthInches, maxHeightInches, formats, pngDpi, converter)`
- `J$HtmlConverter$htmlFragmentToPdf(htmlFragment, outFile, xMarginInches, yMarginInches, maxWidthInches, maxHeightInches, formats, pngDpi, converter)`

Package initialisation and control

Package method `JavaApi$installDependencies()`: This package level method checks for, and installs any dependencies needed for the running of the package. It is called automatically on first package load and so in general does not need to be used directly.

Usage:

```
html2pdfr::JavaApi$installDependencies()
```

Arguments:

- none

Returns: nothing. called for side effects.

Package method `JavaApi$rebuildDependencies()`: This package level method removes existing dependencies and re-installs dependencies needed for the running of the package. It is called automatically on first package load and so in general does not need to be called.

Usage:

```
html2pdfr::JavaApi$rebuildDependencies()
```

Arguments:

- none

Returns: nothing. called for side effects.

Package method `JavaApi$versionInformation()`: This package level method returns debugging version information for the package

Usage:

```
html2pdfr::JavaApi$versionInformation()
```

Arguments:

- none

Returns: A list containing a set of versioning information about this package.

Package method `JavaApi$get()`**:** This is the main entry point for the package and the root of the Java API in this package. All classes defined in the package are made available as items under this root. The JavaApi object manages the communication between R and Java.

Usage:

```
J = html2pdfr::JavaApi$get()
# package classes and functions are nested under the `J` api object.
```

Arguments:

- logLevel The desired verbosity of the package. One of "OFF", "FATAL", "ERROR", "WARN", "INFO", "DEBUG", "TRACE", "ALL".

Returns: A R6 html2pdfr::JavaApi object containing the access point to the objects and functions defined in this package

Api method `J$changeLogLevel(logLevel)`**:** Once the package is initialised the log level can be changed to increase the level of messages from the api.

Usage:

```
J = html2pdfr::JavaApi$get()
J$changeLogLevel("DEBUG")
```

Arguments:

- logLevel The desired verbosity of the package. One of "OFF", "FATAL", "ERROR", "WARN", "INFO", "DEBUG", "TRACE", "ALL".

Returns: nothing. used for side effects.

Api method `J$reconfigureLog(log4jproperties)`**:** Experimental / Advanced use: Once the package is initialised the log configureation can be changed to log to an external file for example.

Usage:

```
J = html2pdfr::JavaApi$get()
prp = fs::path(getwd(),"log4j.properties")
if (fs::file_exists(prp)) {
 J$changeLogLevel(prp)
}
```

Arguments:

- log4jproperties a full path to a log4jproperies file

Returns: nothing. used for side effects.

Api method `J$printMessages()`**:** Experimental / Internal use: Messages from Java to R are queued and printed after each function call. It is unlikely that any will be not printed so in normal circumstances this function should do nothing.

Usage:

```
J = html2pdfr::JavaApi$get()
J$printMessages()
```

Arguments:

- none

Returns: nothing. used for side effects.

Static methods and constructors

Method `HtmlConverter$new()`**:** the default no-args constructor

Usage:

```
J = html2pdfr::JavaApi$get()
J$HtmlConverter$new(fontfiles)
```

Arguments:

- fontfiles - (java expects a String)

Returns: R6 HtmlConverter object:

Method `HtmlConverter$htmlConverter()`**:** Create a new HtmlConverter
for creating PDF and PNG files from HTML. In general this will be created automatically. but if you have specific fonts you want to use then you may need to pass them to this function and specify the result in the ‘converter‘ parameter of the main functions.

Usage:

```
J = html2pdfr::JavaApi$get()
J$HtmlConverter$htmlConverter(fontfiles, update)
# this method is also exposed as a package function:
html2pdfr::html_converter(fontfiles, update)
```

Arguments:

- fontfiles - a character vector of font files that will be imported into the converter. - (defaulting to ‘systemfonts::system_fonts()$path‘) - (java expects a RCharacterVector)
- update - (defaulting to ‘FALSE‘) - (java expects a RLogical)

Returns: R6 HtmlConverter object:

Examples:

```
conv = html2pdfr::html_converter()
```

Method `HtmlConverter$urlToPdf()`**:** Convert HTML document from a URL to a PDF document.
The URL is assumed to be a complete document. The resulting PDF size will be controlled by page media directives within the HTML, unless explicitly given here in ‘maxWidthInches‘ and ‘maxHeightInches‘. If the ‘cssSelector‘ parameter is given the HTML fragment at that selector will be used. In this case it is will be resized to fit within the given dimensions and shrink wrapped so that the content is smaller. If no dimensions are present this will default to A4.

Usage:

```
J = html2pdfr::JavaApi$get()
J$HtmlConverter$urlToPdf(htmlUrl, outFile, cssSelector, xMarginInches, yMarginInches, maxWidthInch
# this method is also exposed as a package function:
html2pdfr::url_to_pdf(htmlUrl, outFile, cssSelector, xMarginInches, yMarginInches, maxWidthInches,
```

Arguments:

- htmlUrl the URL - (java expects a RCharacter)
- outFile the full path of the output file - (defaulting to ‘tempfile(’html2pdfr_’)‘) - (java expects a RFile)
- cssSelector the part of the page you want to convert to PDF. - (defaulting to ‘NA_character_‘) - (java expects a RCharacter)
- xMarginInches page width margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- yMarginInches page height margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- maxWidthInches what is the maximum allowable width? - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- maxHeightInches what is the maximum allowable height? (if the content is larger than this then it will overflow to another page) - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- formats If the outFile does not specify a file extension then you can do so here as "png" or "pdf" or both. - (defaulting to ‘c(’pdf’)‘) - (java expects a RCharacterVector)
- pngDpi the dots per inch for png outputs if requested - (defaulting to ‘300‘) - (java expects a RNumeric)
- converter (optional) a configured HTML converter, only needed if manually specifying fonts. - (defaulting to ‘html2pdfr::html_converter()‘) - (java expects a HtmlConverter)

Returns: RCharacterVector: the filename(s) written to (with extension ’.pdf’ or ’.png’ if outFile did not have an extension).

Examples:

```
library(testthat)
url_to_pdf('https://cran.r-project.org/banner.shtml')
```

Method `HtmlConverter$fileToPdf()`**:** Convert HTML document from a local file to a PDF document.

The HTML in ‘inFile‘ is assumed to be a complete document. Relative references are resolved with reference to the HTML file on the file system, so correctly located images etc whould be picked up without requiring a server. The resulting PDF size will be controlled by page media directives within the HTML, unless explicitly given here in ‘maxWidthInches‘ and ‘maxHeight-Inches‘. If the ‘cssSelector‘ parameter is given the HTML fragment at that selector will be used. In this case it is will be resized to fit within the given dimensions and shrink wrapped so that the content is smaller. If no dimensions are present this will default to A4.

Usage:

```
J = html2pdfr::JavaApi$get()
J$HtmlConverter$fileToPdf(inFile, outFile, cssSelector, xMarginInches, yMarginInches, maxWidthInch
# this method is also exposed as a package function:
html2pdfr::file_to_pdf(inFile, outFile, cssSelector, xMarginInches, yMarginInches, maxWidthInches,
```

Arguments:

- inFile the full path the the HTML file - (java expects a RCharacter)
- outFile the full path of the output file - (defaulting to ‘tempfile(’html2pdfr_’)‘) - (java expects a RFile)
- cssSelector the part of the page you want to convert to PDF. - (defaulting to ‘NA_character_‘) - (java expects a RCharacter)
- xMarginInches page width margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- yMarginInches page height margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- maxWidthInches what is the maximum allowable width? - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- maxHeightInches what is the maximum allowable height? (if the content is larger than this then it will overflow to another page) - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- formats If the outFile does not specify a file extension then you can do so here as "png" or "pdf" or both. - (defaulting to ‘c(’pdf’)‘) - (java expects a RCharacterVector)
- pngDpi the dots per inch for png outputs if requested - (defaulting to ‘300‘) - (java expects a RNumeric)
- converter (optional) a configured HTML converter, only needed if manually specifying fonts. - (defaulting to ‘html2pdfr::html_converter()‘) - (java expects a HtmlConverter)

Returns: RCharacterVector: the filename written to (with extension ’.pdf’ or ’.png’ if outFile did not have an extension).

Examples:

```
library(testthat)
dest = tempfile(fileext='.html')
download.file('https://cran.r-project.org/banner.shtml',
 destfile = dest)
file_to_pdf(dest)
```

Method `HtmlConverter$htmlDocumentToPdf()`**:** Convert HTML document from a string to a PDF document.

The HTML in ‘html‘ is assumed to be a complete document. Relative references are resolved with reference to ‘baseUri‘ if it is given (which could be a ‘file://‘ URI). The resulting PDF size will be controlled by page media directives within the HTML, unless explicitly given here in ‘maxWidthInches‘ and ‘maxHeightInches‘. If the ‘cssSelector‘ parameter is given the HTML fragment at that selector will be used. In this case it is will be resized to fit within the given dimensions and shrink wrapped so that the content is smaller. If no dimensions are present this will default to A4.

Usage:

```
J = html2pdfr::JavaApi$get()
J$HtmlConverter$htmlDocumentToPdf(html, outFile, baseUri, cssSelector, xMarginInches, yMarginInche
# this method is also exposed as a package function:
html2pdfr::html_document_to_pdf(html, outFile, baseUri, cssSelector, xMarginInches, yMarginInches,
```

Arguments:

- html the html document as a string - (java expects a RCharacter)
- outFile the full path of the output file - (defaulting to ‘tempfile(’html2pdfr_’)‘) - (java expects a RFile)
- baseUri the URI from which to interpret relative links in the html content. - (defaulting to ‘NA_character_‘) - (java expects a RCharacter)
- cssSelector the part of the page you want to convert to PDF. - (defaulting to ‘NA_character_‘) - (java expects a RCharacter)
- xMarginInches page width margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- yMarginInches page height margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- maxWidthInches what is the maximum allowable width? - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- maxHeightInches what is the maximum allowable height? (if the content is larger than this then it will overflow to another page) - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
- formats If the outFile does not specify a file extension then you can do so here as "png" or "pdf" or both. - (defaulting to ‘c(’pdf’)‘) - (java expects a RCharacterVector)
- pngDpi the dots per inch for png outputs if requested - (defaulting to ‘300‘) - (java expects a RNumeric)
- converter (optional) a configured HTML converter, only needed if manually specifying fonts. - (defaulting to ‘html2pdfr::html_converter()‘) - (java expects a HtmlConverter)

Returns: RCharacterVector: the filename written to (with extension ’.pdf’ or ’.png’ if outFile did not have an extension).

Examples:

```
library(testthat)
library(readr)
html = read_file('https://fred-wang.github.io/MathFonts/mozilla_mathml_test/')
html_document_to_pdf(html, baseUri = 'https://fred-wang.github.io/MathFonts/mozilla_mathml_test/')
```

Method `HtmlConverter$htmlFragmentToPdf()`**:** Render HTML fragment from a string to a PDF image.

This is the simple rendering function that will output a PDF file (potentially many pages) and a set of PNG files from HTML content. This is primarily used to render HTML content (e.g. a table) that is being included in a larger document. In this case the HTML fragment will not specify page dimensions which need to be provided (defaults to A4 size with 1 inch margins). The result can be embedded into an existing page using latex’s includegraphics directive exactly the same way as a graphical figure might be used. The sizing of the output will always be smaller than the dimensions of a page, but will shrink to fit the content.

Usage:

```
J = html2pdfr::JavaApi$get()
J$HtmlConverter$htmlFragmentToPdf(htmlFragment, outFile, xMarginInches, yMarginInches, maxWidthInc
# this method is also exposed as a package function:
html2pdfr::html_fragment_to_pdf(htmlFragment, outFile, xMarginInches, yMarginInches, maxWidthInche
```

Arguments:

- htmlFragment a HTML fragment, e.g. usually the table element, but may be the whole page. - (java expects a RCharacter)
- outFile the full path with or without extension (if no extension specified then ‘formats‘ parameter will apply) - (defaulting to ‘tempfile('html2pdfr_')‘) - (java expects a RFile)
- xMarginInches page width margins - (defaulting to ‘1.0‘) - (java expects a RNumeric)
- yMarginInches page height margins - (defaulting to ‘1.0‘) - (java expects a RNumeric)
- maxWidthInches what is the maximum allowable width? (default is A4) - (defaulting to ‘8.27‘) - (java expects a RNumeric)
- maxHeightInches what is the maximum allowable height? (if the content is larger than this then it will overflow to another page) - (defaulting to ‘11.69‘) - (java expects a RNumeric)
- formats If the outFile does not specify a file extension then you can do so here as "png" or "pdf" or both. - (defaulting to ‘c('pdf','png')‘) - (java expects a RCharacterVector)
- pngDpi the dots per inch for png outputs if requested. - (defaulting to ‘300‘) - (java expects a RNumeric)
- converter (optional) a configured HTML converter, only needed if manually specifying fonts. - (defaulting to ‘html2pdfr::html_converter()‘) - (java expects a HtmlConverter)

Returns: RCharacterVector: the filename(s) written to (with extension '.pdf' or '.png' if outFile did not have an extension).

Examples:

```
library(testthat)
library(dplyr)
html = iris %>% group_by(Species) %>%
summarise(across(everything(), mean)) %>%
huxtable::as_hux() %>%
huxtable::theme_article() %>%
huxtable::to_html()
html_fragment_to_pdf(html)
```

Author(s)

<rob.challen@bristol.ac.uk>

Examples

```
## -----------------------------------
## Check library dependencies for html2pdfr
## -----------------------------------
```

```
html2pdfr::JavaApi$installDependencies()

## -----------------------------------
## Construct a html2pdfr Java API instance
## -----------------------------------

J = html2pdfr::JavaApi$get()
# or a more verbose configuration
# J = html2pdfr::JavaApi$get("DEBUG")


## -----------------------------------
## Method `J$HtmlConverter$new(...)`
## -----------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$HtmlConverter$new(fontfiles)
# or alternatively:
html2pdfr::new(fontfiles)

## End(Not run)

## -----------------------------------
## Method `J$HtmlConverter$htmlConverter(...)`
## Aliased as `html2pdfr::html_converter(...)`
## -----------------------------------
conv = html2pdfr::html_converter()

## -----------------------------------
## Method `J$HtmlConverter$urlToPdf(...)`
## Aliased as `html2pdfr::url_to_pdf(...)`
## -----------------------------------
library(testthat)
url_to_pdf('https://cran.r-project.org/banner.shtml')

## -----------------------------------
## Method `J$HtmlConverter$fileToPdf(...)`
## Aliased as `html2pdfr::file_to_pdf(...)`
## -----------------------------------
library(testthat)
dest = tempfile(fileext='.html')
download.file('https://cran.r-project.org/banner.shtml',
 destfile = dest)
file_to_pdf(dest)

## -----------------------------------
## Method `J$HtmlConverter$htmlDocumentToPdf(...)`
## Aliased as `html2pdfr::html_document_to_pdf(...)`
## -----------------------------------
library(testthat)
library(readr)
html = read_file('https://fred-wang.github.io/MathFonts/mozilla_mathml_test/')
html_document_to_pdf(html, baseUri = 'https://fred-wang.github.io/MathFonts/mozilla_mathml_test/')
```

```
## -----------------------------------
## Method `J$HtmlConverter$htmlFragmentToPdf(...)`
## Aliased as `html2pdfr::html_fragment_to_pdf(...)`
## -----------------------------------
library(testthat)
library(dplyr)
html = iris %>% group_by(Species) %>%
summarise(across(everything(), mean)) %>%
huxtable::as_hux() %>%
huxtable::theme_article() %>%
huxtable::to_html()
html_fragment_to_pdf(html)
```

---

url_to_pdf *Convert HTML document from a URL to a PDF document.*

Description

The URL is assumed to be a complete document. The resulting PDF size will be controlled by page media directives within the HTML, unless explicitly given here in ‘maxWidthInches‘ and ‘maxHeightInches‘. If the ‘cssSelector‘ parameter is given the HTML fragment at that selector will be used. In this case it is will be resized to fit within the given dimensions and shrink wrapped so that the content is smaller. If no dimensions are present this will default to A4.

Usage

```
url_to_pdf(
 htmlUrl,
 outFile,
 cssSelector,
 xMarginInches,
 yMarginInches,
 maxWidthInches,
 maxHeightInches,
 formats,
 pngDpi,
 converter
)
```

Arguments

`htmlUrl`	htmlUrl the URL - (java expects a RCharacter)
`outFile`	outFile the full path of the output file - (defaulting to ‘tempfile(’html2pdfr_’)‘) - (java expects a RFile)
`cssSelector`	cssSelector the part of the page you want to convert to PDF. - (defaulting to ‘NA_character_‘) - (java expects a RCharacter)

`xMarginInches`	xMarginInches page width margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`yMarginInches`	yMarginInches page height margins - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`maxWidthInches`	maxWidthInches what is the maximum allowable width? - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`maxHeightInches`	maxHeightInches what is the maximum allowable height? (if the content is larger than this then it will overflow to another page) - (defaulting to ‘NA_real_‘) - (java expects a RNumeric)
`formats`	formats If the outFile does not specify a file extension then you can do so here as "png" or "pdf" or both. - (defaulting to ‘c(’pdf’)‘) - (java expects a RCharacterVector)
`pngDpi`	pngDpi the dots per inch for png outputs if requested - (defaulting to ‘300‘) - (java expects a RNumeric)
`converter`	converter (optional) a configured HTML converter, only needed if manually specifying fonts. - (defaulting to ‘html2pdfr::html_converter()‘) - (java expects a HtmlConverter)

Value

RCharacterVector: the filename(s) written to (with extension ’.pdf’ or ’.png’ if outFile did not have an extension).

Examples

```
library(testthat)
url_to_pdf('https://cran.r-project.org/banner.shtml')
```

Index