Package 'prettifyAddins'

November 18, 2021

Type Package

Title 'RStudio' Addins to Prettify 'JavaScript', 'C++', 'Python', and More

Version 2.3.0

Description Provides 'RStudio' addins to prettify 'HTML', 'CSS', 'SCSS', 'JavaScript', 'JSX', 'Markdown', 'C(++)', 'LaTeX', 'Python', 'Julia', 'XML', 'Java', 'JSON', 'Ruby', and to reindent 'C(++)', 'Fortran', 'Java', 'Julia', 'Python', 'SAS', 'Scala', 'Shell', 'SQL' and ``TypeScript''. Two kinds of addins are provided: 'Prettify' and 'Indent'. The 'Indent' addins only reindent the code, while the 'Prettify' addins also modify the code, e.g. trailing semicolons are added to 'JavaScript' code when they are missing.

License GPL-3

Encoding UTF-8

Imports shiny, rstudioapi, tools, xml2, webdriver, XRJulia, httr, utils

Suggests V8, shinythemes, shinyAce, miniUI, testthat

URL https://github.com/stla/prettifyAddins

BugReports https://github.com/stla/prettifyAddins/issues

RoxygenNote 7.1.2

NeedsCompilation no

Author Stéphane Laurent [aut, cre],
James Long and contributors [ctb, cph] ('Prettier' library),
Zeb Zhao [ctb, cph] ('indent.js' library),
Marijn Haverbeke [ctb, cph] ('CodeMirror' library),
George Leslie-Waksman and other contributors [ctb, cph]
('sql-formatter' library)

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Repository CRAN

Date/Publication 2021-11-17 23:50:02 UTC

R topics documented:

getPrettifiableLanguages 2
prettifyAddins 2
prettifyAddins-imports 3
prettifyCLANG 3
prettifyJulia 4
prettifyLaTeX 4
prettifyPython 5
prettifyXML 6
prettify_FCA 6
prettify_Shiny 7
prettify_V8 8
reindent_PhantomJS 9
reindent_Shiny 10
reindent_V8 11

Index **12**

`getPrettifiableLanguages` *Prettifiable languages*

Description

Returns the list of languages that are supported by this package.

Usage

```
getPrettifiableLanguages()
```

`prettifyAddins` *Prettify Addins*

Description

This package provides some RStudio addins: `Prettify` addins and `Indent` addins. To run an addin, select it from the Addins menu within RStudio. The `Indent` addins only reindent the code, while the `Prettify` addins also modify the code, e.g. they add trailing semi-colons to JavaScript code when they are missing.

Examples

```
# get the list of supported languages:
getPrettifiableLanguages()
```

prettifyAddins-imports *Install PhantomJS*

Description

This function is imported from the 'webdriver' package. Follow the link to its documentation: install_phantomjs

prettifyCLANG *Prettify C, C++, Java*

Description

Prettify some C, C++ or Java code.

Usage

```
prettifyCLANG(contents = NA, language = NA, tabSize = NULL)
```

Arguments

contents	the code to be prettified; there are three possibilities for this argument: NA (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
language	the language of the code; when the contents is read from a file, this option is ignored, because the language is obtained from the extension of the file
tabSize	number of spaces of the indentation (usually 2 or 4); if NULL (the default), there are two possibilities: if the contents is read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to 2

Value

The pretty code in a character string.

Note

This function requires the command line utility clang-format.

prettifyJulia *Prettify Julia*

Description

Prettify Julia code.

Usage

```
prettifyJulia(contents = NA, tabSize = NULL)
```

Arguments

contents	the code to be prettified; there are three possibilities for this argument: NA (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
tabSize	number of spaces of the indentation (usually 2 or 4); if NULL (the default), there are two possibilities: if the contents is read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to 2

Value

The pretty code in a character string.

Note

This function requires that Julia is installed on your system and that the Julia package JuliaFormatter is installed.

prettifyLaTeX *Prettify LaTeX*

Description

Prettify LaTeX code, including Sweave code, sty files, cls files, and bib files.

Usage

```
prettifyLaTeX(contents = NA, tabSize = NULL, log = FALSE)
```

Arguments

`contents`	the code to be prettified; there are three possibilities for this argument: `NA` (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
`tabSize`	number of spaces of the indentation (usually `2` or `4`); if `NULL` (the default), there are two possibilities: if the contents is read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to `2`
`log`	logical, whether to generate a log file (it will be named `indent.log`)

Value

The pretty code in a character string.

Note

This function requires the command line utility `latexindent`.

`prettifyPython` *Prettify Python*

Description

Prettify Python code.

Usage

```
prettifyPython(contents = NA)
```

Arguments

`contents`	the code to be prettified; there are three possibilities for this argument: `NA` (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector

Value

The pretty code in a character string.

Note

This function requires black.

prettifyXML *Prettify XML*

Description

Prettify some XML or SVG code.

Usage

```
prettifyXML(contents = NA, tabSize = NULL)
```

Arguments

contents	the code to be prettified; there are three possibilities for this argument: NA (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
tabSize	number of spaces of the indentation (usually 2 or 4); if NULL (the default), there are two possibilities: if the contents are read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to 2

Details

The code is prettified with the help of the command line utility `xmllint` if it is available, otherwise the `xml2` is used.

Value

The pretty code in a character string.

prettify_FCA *Prettify Java, JSON or Ruby*

Description

Prettify Java code, JSON code or Ruby code.

Usage

```
prettify_FCA(contents = NA, language = NA)
```

Arguments

contents	the code to be prettified; there are three possibilities for this argument: NA (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
language	the language of the code, such as "json"; see getPrettifiableLanguages; if the contents are read from a file and language=NA, then the language is guessed from the file extension

Value

The pretty code in a character string.

Note

This function requires a connection to Internet.

Examples

```
library(prettifyAddins)

code <- c(
  "{a: [0,1,  2   ],",
  "f:  function( x){return x+1}}" # this function will be prettified too
)

## Not run:
cat(prettify_FCA(code, "json"))
## End(Not run)
```

---

prettify_Shiny *Prettify code using Shiny*

Description

Prettify some code using a Shiny app.

Usage

```
prettify_Shiny(contents = NA, language = NA, tabSize = NULL, themeInfo = NULL)
```

Arguments

contents	the code to be prettified; there are three possibilities for this argument: NA (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
language	the language of the code, such as "javascript" or "JavaScript"; see getPrettifiableLanguages; if the contents are read from a file and language=NA, then the language is guessed from the file extension

tabSize	number of spaces of the indentation (usually 2 or 4); if NULL (the default), there are two possibilities: if the contents are read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to 2
themeInfo	this argument is not important, it controls the theme of the Shiny app; it must be NULL or a list with two fields: editor, the name of a theme, and dark, a logical value, which tells whether the theme is dark

Value

The pretty code in a character string.

Examples

```
library(prettifyAddins)

code <- c(
  "function f(x){",
  "return x+1",
  "}"
)
if(interactive()){
  cat(prettify_Shiny(code, "javascript"))
}
```

---

prettify_V8 *Prettify code using V8*

Description

Prettify some code using the V8 package.

Usage

```
prettify_V8(contents = NA, language = NA, tabSize = NULL)
```

Arguments

contents	the code to be prettified; there are three possibilities for this argument: NA (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
language	the language of the code, such as "javascript"; see getPrettifiableLanguages; if the contents are read from a file and language=NA, then the language is guessed from the file extension
tabSize	number of spaces of the indentation (usually 2 or 4); if NULL (the default), there are two possibilities: if the contents are read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to 2

Value

The pretty code in a character string.

Examples

```
library(prettifyAddins)

code <- c(
  "function f(x){",
  "return x+1",
  "}"
)
cat(prettify_V8(code, "JavaScript"))
```

`reindent_PhantomJS` *Reindent code using PhantomJS*

Description

Reindent some code using PhantomJS.

Usage

```
reindent_PhantomJS(contents = NA, language = NA, tabSize = NULL)
```

Arguments

`contents`	the code to be reindented; there are three possibilities for this argument: `NA` (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
`language`	the language of the code, such as `"python"`; see `getPrettifiableLanguages`; if the contents are read from a file and `language=NA`, then the language is guessed from the file extension
`tabSize`	number of spaces of the indentation (usually `2` or `4`); if `NULL` (the default), there are two possibilities: if the contents are read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to `2`

Value

The reindented code in a character string.

Note

This function requires the 'phantomjs' command-line utility.

Examples

```
library(prettifyAddins)

code <- c(
  'if test == 1:',
  'print "it is one"',
  'else:',
  'print "it is not one"'
)

## Not run:
cat(reindent_PhantomJS(code, "python"))
## End(Not run)
```

`reindent_Shiny` *Reindent code using Shiny*

Description

Reindent some code using a Shiny app.

Usage

```
reindent_Shiny(contents = NA, language = NA, tabSize = NULL, themeInfo = NULL)
```

Arguments

`contents`	the code to be reindented; there are three possibilities for this argument: `NA` (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
`language`	the language of the code, such as `"javascript"`; see `getPrettifiableLanguages`; if the contents are read from a file and `language=NA`, then the language is guessed from the file extension
`tabSize`	number of spaces of the indentation (usually 2 or 4); if `NULL` (the default), there are two possibilities: if the contents are read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to 2
`themeInfo`	this argument is not important, it controls the theme of the Shiny app; it must be `NULL` or a list with two fields: `editor`, the name of a theme, and `dark`, a logical value, which tells whether the theme is dark

Value

The reindented code in a character string.

reindent_V8 *Reindent code using V8*

Description

Reindent some code using the V8 package.

Usage

```
reindent_V8(contents = NA, language = NA, tabSize = NULL)
```

Arguments

contents	the code to be reindented; there are three possibilities for this argument: NA (default), to use the file currently opened in RStudio; the path to a file; or the code given as a character vector
language	the language of the code, such as "javascript"; see getPrettifiableLanguages; if the contents are read from a file and language=NA, then the language is guessed from the file extension
tabSize	number of spaces of the indentation (usually 2 or 4); if NULL (the default), there are two possibilities: if the contents are read from the current file in RStudio, then the number of spaces will be the one you use in RStudio; otherwise it is set to 2

Value

The reindented code in a character string.

Examples

```
library(prettifyAddins)

code <- c(
  "function f(x){",
  "return x+1",
  "}"
)
cat(reindent_V8(code, "javascript"))
```

Index

getPrettifiableLanguages, 2, *7–11*

install_phantomjs, *3*
install_phantomjs
        *(*prettifyAddins-imports*)*, 3

prettify_FCA, 6
prettify_Shiny, 7
prettify_V8, 8
prettifyAddins, 2
prettifyAddins-imports, 3
prettifyCLANG, 3
prettifyJulia, 4
prettifyLaTeX, 4
prettifyPython, 5
prettifyXML, 6

reindent_PhantomJS, 9
reindent_Shiny, 10
reindent_V8, 11