Package ‘rstudioapi’

January 24, 2016

Title Safely Access the RStudio API

Description Access the RStudio API (if available) and provide informative error messages when it's not.

Version 0.5

Date 2016-01-24

Maintainer JJ Allaire <`jj@rstudio.com`>

License MIT + file LICENSE

RoxygenNote 5.0.1

Suggests testthat

NeedsCompilation no

Author Hadley Wickham [aut],
JJ Allaire [aut, cre],
Kevin Ushey [ctb],
RStudio [cph]

Repository CRAN

Date/Publication 2016-01-24 15:45:34

R topics documented:

askForPassword *Ask the user for a password interactively*

Description

Ask the user for a password interactively.

Usage

```
askForPassword(prompt)
```

Arguments

`prompt`	Single element character vector containing the prompt to be displayed

Details

RStudio also sets the global `askpass` option to the `rstudioapi::askForPassword` function so that it can be invoked in a front-end independent manner.

Note

The `askForPassword` function was added in version 0.99.853 of RStudio.

Examples

```
## Not run:
rstudioapi::askForPassword("Please enter your password")

## End(Not run)
```

callFun *Call an RStudio API function*

Description

This function will return an error if RStudio is not running, or the function is not available. If you want to fall back to different behavior, use `hasFun`.

Usage

```
callFun(fname, ...)
```

Arguments

`fname`	name of the RStudio function to call.
`...`	Other arguments passed on to the function

Examples

```
if (rstudioapi::isAvailable()) {
  rstudioapi::callFun("versionInfo")
}
```

`document_position` *Create a Document Position*

Description

Creates a `document_position`, which can be used to indicate e.g. the row + column location of the cursor in a document.

Usage

```
document_position(row, column)

is.document_position(x)

as.document_position(x)
```

Arguments

`row`	The row (using 1-based indexing).
`column`	The column (using 1-based indexing).
`x`	An object coercable to `document_position`.

`document_range` *Create a Range*

Description

A `document_range` is a pair of `document_position` objects, with each position indicating the `start` and end of the range, respectively.

Usage

```
document_range(start, end = NULL)

is.document_range(x)

as.document_range(x)
```

Arguments

`start`	A `document_position` indicating the start of the range.
`end`	A `document_position` indicating the end of the range.
`x`	An object coercable to `document_range`.

Value

An R `list` with class `document_range` and fields:

`start:`	The start position.
`end:`	The end position.

`getActiveDocumentContext` *Get the Active Document Context*

Description

Returns information about the currently active RStudio document.

Usage

```
getActiveDocumentContext()
```

Details

The `selection` field returned is a list of document selection objects. A document selection is just a pairing of a document `range`, and the `text` within that range.

Value

A `list` with elements:

`id`	The document ID.
`path`	The path to the document on disk.
`contents`	The contents of the document.
`selection`	A `list` of selections. See **Details** for more information.

Note

The `getActiveDocumentContext` function was added with version 0.99.796 of RStudio.

`getActiveProject` *Path to Active RStudio Project*

Description

Returns the path to the currently active RStudio project.

Usage

```
getActiveProject()
```

Value

Returns a single element character vector with the path of the currently active RStudio project. Returns `NULL` if no project is active.

Note

The `getActiveProject` function was added in version 0.99.854 of RStudio.

Examples

```
## Not run:
rstudioapi::getActiveProject()

## End(Not run)
```

`getVersion` *Return the current version of the RStudio API*

Description

Return the current version of the RStudio API

Usage

```
getVersion()
```

Value

A `numeric_version` which you can compare to a string and get correct results.

Examples

```
## Not run:
if (rstudioapi::getVersion() < "0.98.100") {
  message("Your version of RStudio is quite old")
}

## End(Not run)
```

hasFun *Exists/get for RStudio functions*

Description

These are specialized versions of `get` and `exists` that look in the rstudio package namespace. If RStudio is not running, `hasFun` will return `FALSE`.

Usage

```
hasFun(name, version_needed = NULL, ...)

findFun(name, version_needed = NULL, ...)
```

Arguments

`name`	name of object to look for
`version_needed`	An optional version specification. If supplied, ensures that RStudio is at least that version. This is useful if function behavior has changed over time.
`...`	other arguments passed on to `exists` and `get`

Examples

```
rstudioapi::hasFun("viewer")
```

insertText *Modify the Contents of a Document*

Description

Use these functions to modify the contents of a document open in RStudio.

Usage

```
insertText(location, text, id = NULL)

modifyRange(location, text, id = NULL)

setDocumentContents(text, id = NULL)
```

Arguments

`location`	An object specifying the positions, or ranges, wherein text should be inserted. See **Details** for more information.
`text`	A character vector, indicating what text should be inserted at each aforementioned range. This should either be length one (in which case, this text is applied to each range specified); otherwise, it should be the same length as the `ranges` list.
`id`	The document id. When `NULL` or blank, the mutation will apply to the currently open, or last focused, RStudio document. Use the `id` returned from `getActiveDocumentContext()` to ensure that the operation is applied on the intended document.

Details

`location` should be a (list of) `document_position` or `document_range` object(s), or numeric vectors coercable to such objects.

To operate on the current selection in a document, call `insertText()` with only a text argument, e.g.

```
insertText("# Hello\n")
insertText(text = "# Hello\n")
```

Otherwise, specify a (list of) positions or ranges, as in:

```
# insert text at the start of the document
insertText(c(1, 1), "# Hello\n")

# insert text at the end of the document
insertText(Inf, "# Hello\n")

# comment out the first 5 rows
pos <- Map(c, 1:5, 1)
insertText(pos, "# ")

# uncomment the first 5 rows, undoing the previous action
rng <- Map(c, Map(c, 1:5, 1), Map(c, 1:5, 3))
modifyRange(rng, "")
```

`modifyRange` is a synonym for `insertText`, but makes its intent clearer when working with ranges, as performing text insertion with a range will replace the text previously existing in that range with new text. For clarity, prefer using `insertText` when working with `document_position`s, and `modifyRange` when working with `document_range`s.

Note

The `insertText`, `modifyRange` and `setDocumentContents` functions were added with version 0.99.796 of RStudio.

`isAvailable` *Check if RStudio is running.*

Description

Check if RStudio is running.

Usage

```
isAvailable(version_needed = NULL)

verifyAvailable(version_needed = NULL)
```

Arguments

`version_needed`	An optional version specification. If supplied, ensures that RStudio is at least that version.

Value

`isAvailable` a boolean; `verifyAvailable` an error message if RStudio is not running

Examples

```
rstudioapi::isAvailable()
## Not run: rstudioapi::verifyAvailable()
```

`navigateToFile` *Navigate to File*

Description

Open a file in RStudio, optionally at a specified location.

Usage

```
navigateToFile(file, line = 1L, column = 1L)
```

Arguments

`file`	Path to the file to open)
`line`	Optional; integer specifying the line number on which to place the cursor
`column`	Optional; integer specifying the column number on which to place the cursor

Details

The `navigateToFile` opens a file in RStudio. If the file is already open, its tab or window is activated.

Once the file is open, the cursor is moved to the specified location.

Note that if your intent is to navigate to a particular function within a file, you can also cause RStudio to navigate there by invoking `View` on the function, which has the advantage of falling back on deparsing if the file is not available.

Note

The `navigateToFile` function was added in version 0.99.719 of RStudio.

`previewRd` *Preview an Rd topic in the Help pane*

Description

Preview an Rd topic in the Help pane

Usage

```
previewRd(rdFile)
```

Arguments

`rdFile`	Single element character vector containing the name of the Rd file to be displayed

Note

The `previewRd` function was added in version 0.98.191 of RStudio.

Examples

```
## Not run:
rstudioapi::previewRd("~/MyPackage/man/foo.Rd")

## End(Not run)
```

sourceMarkers *Display Source Markers*

Description

Display user navigable source markers in a pane within RStudio

Usage

```
sourceMarkers(name, markers, basePath = NULL,
              autoSelect = c("none", "first", "error"))
```

Arguments

`name`	Name of marker set (will replace any markers of the same name previously shown)
`markers`	List or data frame containing source markers (see below for details on how to specify markers)
`basePath`	Optional. If all source files are within a base path then specifying that path here will result in file names being displayed as relative paths. Note that in this case markers still need to specify source file names as full paths.
`autoSelect`	Optional. Automatically select and drive focus to either the first marker or the first marker that is an error.

Details

The `markers` argument can contains either a list of marker lists or a data frame with the appropriate marker columns. The fields in a marker are as follows (all are required):

`type`	Marker type ("error", "warning", "info", "style", or "usage")
`file`	Path to source file
`line`	Line number witin source file
`column`	Column number within line
`message`	Short descriptive message

Note that if the `message` field is of class "html" (i.e. `inherits(message, "html") == TRUE`) then it's contents will be treated as HTML.

Note

The `sourceMarkers` function was added in version 0.99.225 of RStudio.

`versionInfo` *RStudio Version Information*

Description

Provides information about the currently running version of RStudio, including it's specific version number and whether it is running in desktop or server mode.

Usage

```
versionInfo()
```

Value

Returns a list with the following elements:

`version`	A package version object that can be used in comparisons. This is the same value which would be returned from
`mode`	Current program mode (either "desktop" or "server")
`citation`	An object inheriting from class `bibentry`

Note

The `versionInfo` function was added in version 0.97.124 of RStudio.

Examples

```
## Not run:
require(rstudioapi)
ver <- versionInfo()

# Test specific version constraint
if (ver$version >= "0.97") {
   # do some 0.97 dependent stuff
}

# Check current mode
desktopMode <- ver$mode == "desktop"
serverMode <- ver$mode == "server"

# Get the citation
ver$citation


## End(Not run)
```

`viewer` *View local web content within RStudio*

Description

View local web content within RStudio. Content can be served from static files in the R session temporary directory or can be a Shiny, Rook, OpenCPU, or any other type of localhost web application.

Usage

```
viewer(url, height = NULL)
```

Arguments

`url`	Application URL. This can be either a localhost URL or a path to a file within the R session temporary directory (i.e. a path returned by `tempfile`).
`height`	Desired height. Specifies a desired height for the Viewer pane (the default is `NULL` which makes no change to the height of the pane). This value can be numeric or the string `"maximize"` in which case the Viewer will expand to fill all vertical space. See details below for a discussion of constraints imposed on the height.

Details

RStudio also sets the global `viewer` option to the `rstudioapi::viewer` function so that it can be invoked in a front-end independent manner.

Applications are displayed within the Viewer pane. The application URL must either be served from localhost or be a path to a file within the R session temporary directory. If the URL doesn't conform to these requirements it is displayed within a standard browser window.

The `height` parameter specifies a desired height, however it's possible the Viewer pane will end up smaller if the request can't be fulfilled (RStudio ensures that the pane paired with the Viewer maintains a minimum height). A height of 400 pixels or lower is likely to succeed in a large proportion of configurations.

A very large height (e.g. 2000 pixels) will allocate the maximum allowable space for the Viewer (while still preserving some view of the pane above or below it). The value `"maximize"` will force the Viewer to full height. Note that this value should only be specified in cases where maximum vertical space is essential, as it will result in one of the user's other panes being hidden.

Viewer Detection

When a page is displayed within the Viewer it's possible that the user will choose to pop it out into a standalone browser window. When rendering inside a standard browser you may want to make different choices about how content is laid out or scaled. Web pages can detect that they are running inside the Viewer pane by looking for the `viewer_pane` query parameter, which is automatically injected into URLs when they are shown in the Viewer. For example, the following URL:

```
http://localhost:8100
```

When rendered in the Viewer pane is transformed to:

```
http://localhost:8100?viewer_pane=1
```

To provide a good user experience it's strongly recommended that callers take advantage of this to automatically scale their content to the current size of the Viewer pane. For example, re-rendering a JavaScript plot with new dimensions when the size of the pane changes.

Note

The `viewer` function was added in version 0.98.423 of RStudio. The ability to specify `maximize` for the `height` parameter was introduced in version 0.99.1001 of RStudio.

Examples

```
## Not run:

# run an application inside the IDE
rstudioapi::viewer("http://localhost:8100")

# run an application and request a height of 500 pixels
rstudioapi::viewer("http://localhost:8100", height = 500)

# probe for viewer option then fall back to browseURL
viewer <- getOption("viewer")
if (!is.null(viewer))
   viewer("http://localhost:8100")
else
   utils::browseURL("http://localhost:8100")

# generate a temporary html file and display it
dir <- tempfile()
dir.create(dir)
htmlFile <- file.path(dir, "index.html")
# (code to write some content to the file)
rstudioapi::viewer(htmlFile)

## End(Not run)
```

Index