Package ‘rstudioapi’

March 19, 2019

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

Version  0.10
Maintainer  Kevin Ushey <kevin@rstudio.com>
License  MIT + file LICENSE

URL  https://github.com/rstudio/rstudioapi
BugReports  https://github.com/rstudio/rstudioapi/issues
RoxygenNote  6.1.1
Suggests  testthat, knitr, rmarkdown
VignetteBuilder  knitr

Encoding  UTF-8
NeedsCompilation  no

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

Repository  CRAN
Date/Publication  2019-03-19 20:30:03 UTC

R topics documented:

addTheme ................................................. 3
applyTheme .............................................. 4
askForPassword ........................................... 4
askForSecret ............................................. 5
build-tools ............................................. 5
callFun .................................................. 6
convertTheme ............................................ 7
createProjectTemplate ........................................... 8
dictionaries ......................................................... 8
document_position ................................................ 9
document_range .................................................... 10
executeCommand ................................................... 10
file-dialogs .......................................................... 11
getActiveProject ................................................... 12
getThemeInfo ......................................................... 12
getThemes ............................................................ 13
getVersion ............................................................. 13
hasColorConsole .................................................... 14
hasFun ................................................................. 15
isAvailable .......................................................... 15
jobAdd ................................................................. 16
jobAddOutput ........................................................ 17
jobAddProgress ..................................................... 17
jobRemove ........................................................... 18
jobRunScript ........................................................ 18
jobSetProgress ...................................................... 19
jobSetState ........................................................... 19
jobSetStatus ........................................................ 20
launcher ............................................................... 21
launcherConfig ....................................................... 21
launcherContainer .................................................. 22
launcherControlJob ................................................ 22
launcherGetJob ....................................................... 23
launcherHostMount ................................................ 23
launcherNfsMount ................................................... 24
launcherPlacementConstraint .................................... 24
launcherResourceLimit ............................................ 25
launcherSubmitJob .................................................. 25
launcherSubmitR ...................................................... 25
navigateToFile ....................................................... 27
persistent-values ................................................... 27
previewRd ............................................................. 28
previewSql ........................................................... 29
primary_selection .................................................. 29
projects ............................................................... 30
readPreference ....................................................... 30
removeTheme ........................................................ 31
restartSession ....................................................... 32
rstudio-documents ................................................ 32
rstudio-editors ...................................................... 34
savePlotAsImage ..................................................... 35
sendToConsole ....................................................... 36
showDialog ........................................................... 36
showPrompt .......................................................... 37
showQuestion ........................................................ 37
addTheme

Description

Adds a custom editor theme to RStudio and returns the name of the newly added theme.

Usage

addTheme(themePath, apply = FALSE, force = FALSE, globally = FALSE)

Arguments

- **themePath**: A full or relative path or URL to an rstheme or tmtheme to be added.
- **apply**: Whether to immediately apply the newly added theme. Setting this to TRUE has the same impact as running `{rstudioapi::addTheme(<themePath>); rstudioapi::applyTheme(<themePath>)}`. Default: FALSE.
- **force**: Whether to force the operation and overwrite an existing file with the same name. Default: FALSE.
- **globally**: Whether to install this theme for the current user or all users. If set to TRUE this will attempt to install the theme for all users, which may require administrator privileges. Default: FALSE.
**Note**

The `applyTheme` function was introduced in RStudio 1.2.879.

---

### applyTheme

**Apply an Editor Theme to RStudio**

**Description**

Applies the specified editor theme to RStudio.

**Usage**

```r
applyTheme(name)
```

**Arguments**

- `name` The unique name of the theme to apply.

**Note**

The `applyTheme` function was introduced in RStudio 1.2.879.

---

### askForPassword

**Ask the user for a password interactively**

**Description**

Ask the user for a password interactively.

**Usage**

```r
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.
askForSecret

Examples

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

## End(Not run)
```

Description

Shows a dialog box asking for a secret with support to remember such secret using the `keyring` package.

Usage

```r
askForSecret(name, message = paste(name, ".", sep = ""),
             title = paste(name, "Secret"))
```

Arguments

- `name`: The name of the secret.
- `message`: A character vector with the contents to display in the main dialog area.
- `title`: The title to display in the dialog box.

Note

The `askForSecret` function was added in version 1.1.419 of RStudio.

build-tools

Build Tools

Description

Check, install, and use build tools as required.

Usage

```r
buildToolsCheck()

buildToolsInstall(action)

buildToolsExec(expr)
```
Arguments

- **action**: The action (as a string) being taken that will require installation of build tools.
- **expr**: An R expression (unquoted) to be executed with build tools available and on the PATH.

Details

These functions are intended to be used together – one should first check whether build tools are available, and when not, prompt for installation. For example:

```r
compile_model <- function(...) {
  if (rstudioapi::isAvailable()) {
    if (!rstudioapi::buildToolsCheck())
      rstudioapi::buildToolsInstall("Model compilation")
    rstudioapi::buildToolsExec({
      # code requiring build tools here
    })
  }
}
```

The action parameter is used to communicate (with a prompt) the operation being performed that requires build tool installation. Setting it to `NULL` or the empty string will suppress that prompt.

Note

The `buildToolsCheck()`, `buildToolsInstall()`, and `buildToolsExec()` functions were added with version 1.2.962 of RStudio.

---

**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.
**convertTheme**

**Examples**

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

---

**convertTheme**

_Convert a tmTheme to an RStudio Theme_

**Description**

Converts a tmTheme to an rstheme and optionally adds and applies it to RStudio and returns the name of the theme.

**Usage**

```r
convertTheme(themePath, add = TRUE, outputLocation = NULL, apply = FALSE, force = FALSE, globally = FALSE)
```

**Arguments**

- `themePath` A full or relative path to the tmTheme file to be converted.
- `add` Whether to add the newly converted theme to RStudio. Setting this to true will have the same impact as running `{ rstudioapi::convertTheme(<themePath>, outputLocation = <convertedThemePath>); rstudioapi::addtheme(<convertedThemePath>) }`. Default: TRUE.
- `outputLocation` A full or relative path where a copy of the converted theme will be saved. If this value is NULL, no copy will be saved. Default: NULL.
- `apply` Whether to immediately apply the newly added theme. This parameter cannot be set to TRUE if add is set to FALSE. Setting this and add to TRUE has the same impact as running `{ rstudioapi::convertTheme(<themePath>, outputLocation = <convertedThemePath>); rstudioapi::applytheme(<themeName>) }`. Default: FALSE.
- `force` Whether to force the operation and overwrite an existing file with the same name. Default: FALSE.
- `globally` Whether to install this theme for the current user or all users. If set to TRUE this will attempt to install the theme for all users, which may require administrator privileges. Only applies when add is TRUE. Default: FALSE.

**Note**

The convertTheme function was introduced in RStudio 1.2.879.
 dictionaries

createProjectTemplate  Create a Project Template

Description

Create a project template. See https://rstudio.github.io/rstudio-extensions/rstudio_project_templates.html for more information.

Usage

createProjectTemplate(package = ".", binding, title,
             subtitle = paste("Create a new", title), caption = paste("Create",
                title), icon = NULL, open_files = NULL, overwrite = FALSE,
                edit = TRUE)

Arguments

package  The path to an R package source.
binding  The R skeleton function to associate with this project template. This is the name of the function that will be used to initialize the project.
title  The title to be shown within the New Project... wizard.
subtitle  (optional) The subtitle to be shown within the New Project... wizard.
caption  (optional) The caption to be shown on the landing page for this template.
icon  (optional) The path to an icon, on disk, to be used in the dialog. Must be an .png of size less than 64KB.
open_files  (optional) Files that should be opened by RStudio when the project is generated. Shell-style globs can be used to indicate when multiple files matching some pattern should be opened – for example, OpenFiles: R/*.R would indicate that RStudio should open all .R files within the R folder of the generated project.
overwrite  Boolean; overwrite a pre-existing template file if one exists?
edit  Boolean; open the file for editing after creation?

 dictionaries  Interact with RStudio’s Dictionaries

Description

Interact with the hunspell dictionaries used by RStudio for spell checking.

Usage

dictionariesPath()

userDictionariesPath()
Details

dictionariesPath() gives a path to the dictionaries installed and distributed with RStudio.
userDictionariesPath() gives the path where users can provide their own custom hunspell
dictionaries. See:
https://support.rstudio.com/hc/en-us/articles/200551916-Spelling-Dictionaries
for more information.

Note

The dictionariesPath() and userDictionariesPath() functions were introduced with RStudio
1.2.1202.

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.

---

**executeCommand**  
*Execute Command*

**Description**

Executes an arbitrary RStudio command.

**Usage**

executeCommand(commandId, quiet = FALSE)

**Arguments**

- **commandId** 
  The ID of the command to execute.

- **quiet** 
  Whether to show an error if the command does not exist.
Details

Most menu commands and many buttons in RStudio can be invoked from the API using this method. The `quiet` command governs the behavior of the function when the command does not exist. By default, an error is shown if you attempt to invoke a non-existent command. You should set this to `FALSE` when invoking a command that may not be available if you don’t want your users to see an error.

The command is run asynchronously, so no status is returned.

Note

The `executeCommand` function was introduced in RStudio 1.2.1261.

---

**file-dialogs**

*Select a File / Folder*

---

**Description**

Prompt the user for the path to a file or folder, using the system file dialogs with RStudio Desktop, and RStudio's own web dialogs with RStudio Server.

**Usage**

```r
selectFile(caption = "Select File", label = "Select",
          path = getActiveProject(), filter = "All Files (*)",
          existing = TRUE)

selectDirectory(caption = "Select Directory", label = "Select",
                path = getActiveProject())
```

**Arguments**

- `caption` The window title.
- `label` The label to use for the ‘Accept’ / ‘OK’ button.
- `path` The initial working directory, from which the file dialog should begin browsing. Defaults to the current RStudio project directory.
- `filter` A glob filter, to be used when attempting to open a file with a particular extension. For example, to scope the dialog to R files, one could use `R Files (*.R)` here.
- `existing` Boolean; should the file dialog limit itself to existing files on the filesystem, or allow the user to select the path to a new file?

**Details**

When the selected file resolves within the user’s home directory, RStudio will return an aliased path – that is, prefixed with `~/`. 
Note

The `selectFile` and `selectDirectory` functions were added in version 1.1.287 of RStudio.

---

**getActiveProject**  
*Path to Active RStudio Project*

**Description**

Returns the path to the currently active RStudio project.

**Usage**

```r
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.

---

**getThemeInfo**  
*Retrieve Themes*

**Description**

Retrieves a list with information about the current color theme used by RStudio.

**Usage**

```r
getThemeInfo()
```
getThemes

Details

A list is returned with the following elements:

- **editor** The name of the current editor theme, such as Textmate.
- **global** The name of the current global theme. One of Modern, Classic, or Sky.
- **dark** TRUE if the editor theme is dark, FALSE otherwise.
- **foreground** The current editor theme’s default text foreground color, formatted as a CSS-compatible color string, such as rgb(1, 22, 39). Supported since RStudio 1.2.1214.
- **background** The current editor theme’s default text background color, formatted as a CSS-compatible color string. Supported since RStudio 1.2.1214.

---

getThemes

Get Theme List

Description

Retrieves a list of the names of all the editor themes installed for RStudio.

Usage

getThemes()

Note

The getThemes function was introduced in RStudio 1.2.879.

---

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

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

## End(Not run)
```

---

**hasColorConsole**

*Check if Console Supports ANSI Color Escapes*

**Description**

Check if Console Supports ANSI Color Escapes

**Usage**

```r
hasColorConsole()
```

**Value**

a boolean

**Note**

The `hasColorConsole` function was added in version 1.1.216 of RStudio.

**Examples**

```r
## Not run:
if (rstudioapi::hasColorConsole()) {
  message("RStudio console supports ANSI color sequences.")
}

## End(Not run)
```
hasFun

**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**

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

```r
findFirst(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**

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

---

isAvailable

**Check if RStudio is running.**

**Description**

Check if RStudio is running.

**Usage**

```r
isAvailable(version_needed = NULL)
```

```r
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
jobAdd

Examples

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

---

**jobAdd**  
*Add a Job*

---

**Description**

Inform RStudio’s Jobs pane that a job has been added.

**Usage**

```r
jobAdd(name, status = "", progressUnits = 0L, actions = NULL,  
running = FALSE, autoRemove = TRUE, show = TRUE)
```

**Arguments**

- **name**: The job’s name.
- **status**: The initial status text for the job; optional.
- **progressUnits**: The integer number of units of work in the job; for example, 100L if the job’s progress is expressed in percentages. Use 0L if the number of units of work is unknown.
- **actions**: A list of actions that can be performed on the job (see Actions).
- **running**: Whether the job is currently running.
- **autoRemove**: Whether to remove the job from the Jobs pane when it’s complete.
- **show**: Whether to show the job in the Jobs pane.

**Value**

An ID representing the newly added job, used as a handle to provide further updates of the job’s status.

**Actions**

The `actions` parameter is a named list of functions that the user can invoke on the job; for example:

```r
actions = list(stop = function(id) { ... })
```

The function will be passed a parameter named `id` with the job ID that invoked it.

There are two special action names:

- **stop** If there is an action named `stop`, then the job will have a Stop button in the Jobs pane, and pressing that button will invoke the `stop` action.
- **info** If there is an action named `info`, then the job will have an informational link in the Jobs pane rather than an output display, and clicking the link will invoke the `info` action.
jobAddOutput

See Also

Other jobs: jobAddOutput, jobAddProgress, jobRemove, jobRunScript, jobSetProgress, jobSetState, jobSetStatus

---

jobAddOutput  Add Job Output

Description

Adds text output to a job.

Usage

```
jobAddOutput(job, output, error = FALSE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>job</td>
<td>The ID of the job that has emitted text.</td>
</tr>
<tr>
<td>output</td>
<td>The text output emitted by the job.</td>
</tr>
<tr>
<td>error</td>
<td>Whether the output represents an error.</td>
</tr>
</tbody>
</table>

See Also

Other jobs: jobAddProgress, jobAdd, jobRemove, jobRunScript, jobSetProgress, jobSetState, jobSetStatus

---

jobAddProgress  Add Job Progress

Description

Adds incremental progress units to a job.

Usage

```
jobAddProgress(job, units)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>job</td>
<td>The ID of the job to update progress for.</td>
</tr>
<tr>
<td>units</td>
<td>The integer number of new progress units completed.</td>
</tr>
</tbody>
</table>

See Also

Other jobs: jobAddOutput, jobAdd, jobRemove, jobRunScript, jobSetProgress, jobSetState, jobSetStatus
### jobRemove

**Remove a Job**

Description

Remove a job from RStudio's Jobs pane.

Usage

```r
jobRemove(job)
```

Arguments

- **job**
  The ID of the job to remove.

See Also

Other jobs: `jobAddOutput`, `jobAddProgress`, `jobAdd`, `jobRunScript`, `jobSetProgress`, `jobSetState`, `jobSetStatus`

### jobRunScript

**Run R Script As Job**

Description

Starts an R script as a background job.

Usage

```r
jobRunScript(path, name = NULL, encoding = "unknown",
              workingDir = NULL, importEnv = FALSE, exportEnv = ")
```

Arguments

- **path**
  The path to the R script to be run.
- **name**
  A name for the background job. When NULL (the default), the filename of the script is used as the job name.
- **encoding**
  The text encoding of the script, if known.
- **workingDir**
  The working directory in which to run the job. When NULL (the default), the parent directory of the R script is used.
- **importEnv**
  Whether to import the global environment into the job.
- **exportEnv**
  The name of the environment in which to export the R objects created by the job. Use "" (the default) to skip export, "R_GlobalEnv" to export to the global environment, or the name of an environment object to create an object with that name.
See Also

Other jobs: jobAddOutput, jobAddProgress, jobAdd, jobRemove, jobSetProgress, jobSetState, jobSetStatus

---

jobSetProgress  
Set Job Progress

Description

Updates the progress for a job.

Usage

jobSetProgress(job, units)

Arguments

- **job**: The ID of the job to set progress for.
- **units**: The integer number of total units of work completed so far.

See Also

Other jobs: jobAddOutput, jobAddProgress, jobAdd, jobRemove, jobRunScript, jobSetState, jobSetStatus

---

jobSetState  
Set Job State

Description

Changes the state of a job.

Usage

jobSetState(job, state = c("idle", "running", "succeeded", "cancelled", "failed"))

Arguments

- **job**: The ID of the job on which to change state.
- **state**: The new job state.
jobSetStatus

States

The following states are supported:

idle  The job is waiting to run.
running  The job is actively running.
succeeded  The job has finished successfully.
cancelled  The job was cancelled.
failed  The job finished but did not succeed.

See Also

Other jobs: jobAddOutput, jobAddProgress, jobAdd, jobRemove, jobRunScript, jobSetProgress, jobSetState

jobSetStatus  Set Job Status

Description

Update a job’s informational status text.

Usage

jobSetStatus(job, status)

Arguments

job  The ID of the job to update.
status  Text describing job’s new status.

See Also

Other jobs: jobAddOutput, jobAddProgress, jobAdd, jobRemove, jobRunScript, jobSetProgress, jobSetState
launch

**Retrieve Launcher Information**

**Description**

Retrieve information about the launcher, as well as the different clusters that the launcher has been configured to use.

Check if the RStudio launcher is available and configured to support 'ad-hoc' jobs; that is, jobs normally launched by the user through the RStudio IDE’s user interface.

Retrieve information on launcher jobs.

**Usage**

```r
launcherGetInfo()

launcherAvailable()

launcherGetJobs(statuses = NULL, fields = NULL, tags = NULL,
                 includeSessions = FALSE)
```

**Arguments**

- **statuses**
  
  Return only jobs whose status matches one of statuses. Valid statuses are: Pending, Running, Suspended, Failed, Finished, Killed, Canceled. When NULL, all jobs are returned.

- **fields**
  
  Return a subset of fields associated with each job object. When NULL, all fields associated with a particular job are returned.

- **tags**
  
  An optional set of tags. Only jobs that have been assigned one of these requested tags will be returned.

- **includeSessions**
  
  Boolean; include jobs which are also operating as RStudio R sessions?

**Define a Launcher Configuration**

**Description**

Define a launcher configuration, suitable for use with the `config` argument to `launcherSubmitJob()`.

**Usage**

```r
launcherConfig(name, value = NULL)
```
launcherControlJob

Arguments

name The name of the launcher configuration.
value The configuration value. Must either be an integer, float, or string.

See Also

Other job submission: launcherContainer, launcherHostMount, launcherNfsMount, launcherPlacementConstraint, launcherResourceLimit, launcherSubmitJob, launcherSubmitR

launcherContainer Define a Launcher Container

Description

Define a launcher container, suitable for use with the container argument to launcherSubmitJob().

Usage

launcherContainer(image, runAsUserId = NULL, runAsGroupId = NULL)

Arguments

image The container image to use.
runAsUserId The user id to run as within the container. Defaults to the container-specified user.
runAsGroupId The group id to run as within the container. Defaults to the container-specified group.

See Also

Other job submission: launcherConfig, launcherHostMount, launcherNfsMount, launcherPlacementConstraint, launcherResourceLimit, launcherSubmitJob, launcherSubmitR

launcherControlJob Interact with (Control) a Job

Description

Interact with a job.

Usage

launcherControlJob(jobId, operation = c("suspend", "resume", "stop", "kill", "cancel"))
### launcherGetJob

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobId</td>
<td>The job id.</td>
</tr>
<tr>
<td>operation</td>
<td>The operation to execute. The operation should be one of {&quot;suspend&quot;, &quot;resume&quot;, &quot;stop&quot;, &quot;kill&quot;, &quot;cancel&quot;}. Note that different launcher plugins support different subsets of these operations – consult your launcher plugin documentation to see which operations are supported.</td>
</tr>
</tbody>
</table>

#### Description

Retrieve information on a job with id jobId.

#### Usage

`launcherGetJob(jobId)`

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobId</td>
<td>The id of a launcher job.</td>
</tr>
</tbody>
</table>

### launcherHostMount

#### Description

Define a launcher host mount, suitable for use with the mounts argument to `launcherSubmitJob()`. This can be used to mount a path from the host into the generated container.

#### Usage

`launcherHostMount(path, mountPath, readOnly = TRUE)`

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>The host path to be mounted.</td>
</tr>
<tr>
<td>mountPath</td>
<td>The destination path for the mount in the container.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Boolean; should the path be mounted read-only?</td>
</tr>
</tbody>
</table>

#### See Also

Other job submission: `launcherConfig`, `launcherContainer`, `launcherNfsMount`, `launcherPlacementConstraint`, `launcherResourceLimit`, `launcherSubmitJob`, `launcherSubmitR`
Define a Launcher NFS Mount

**Description**
Define a launcher NFS mount, suitable for use with the mounts argument to `launcherSubmitJob()`. This can be used to mount a path from a networked filesystem into a newly generated container.

**Usage**
```
launcherNfsMount(host, path, mountPath, readOnly = TRUE)
```

**Arguments**
- **host**: The host name, or IP address, of the NFS server.
- **path**: The NFS path to be mounted.
- **mountPath**: The destination path for the mount in the container.
- **readOnly**: Boolean; should the path be mounted read-only?

**See Also**
Other job submission: `launcherConfig`, `launcherContainer`, `launcherHostMount`, `launcherPlacementConstraint`, `launcherResourceLimit`, `launcherSubmitJob`, `launcherSubmitR`

Define a Launcher Placement Constraint

**Description**
Define a launcher placement constraint, suitable for use with the placementConstraints argument to `launcherSubmitJob()`.

**Usage**
```
launcherPlacementConstraint(name, value = NULL)
```

**Arguments**
- **name**: The name of this placement constraint.
- **value**: The value of the constraint. A job will only be placed on a requested node if the requested placement constraint is present.

**See Also**
Other job submission: `launcherConfig`, `launcherContainer`, `launcherHostMount`, `launcherNfsMount`, `launcherResourceLimit`, `launcherSubmitJob`, `launcherSubmitR`
launcherResourceLimit  

**Define a Launcher Resource Limit**

**Description**

Define a launcher resource limit, suitable for use with the `resourceLimits` argument to `launcherSubmitJob()`.

**Usage**

`launcherResourceLimit(type, value)`

**Arguments**

- **type**: The resource limit type. Must be one of `cpuCount`, `cpuFrequency`, `cpuSet`, `cpuTime`, `memory`, `memorySwap`. Different launcher plugins may support different subsets of these resource limit types; please consult the plugin documentation to learn which limits are supported.

- **value**: The formatted value of the requested limit.

**See Also**

Other job submission: `launcherConfig`, `launcherContainer`, `launcherHostMount`, `launcherNfsMount`, `launcherPlacementConstraint`, `launcherSubmitJob`, `launcherSubmitR`  

----------

launcherSubmitJob  

**Submit a Launcher Job**

**Description**

Submit a launcher job. See https://docs.rstudio.com/job-launcher/latest/index.html for more information.

**Usage**

`launcherSubmitJob(name, cluster = "Local", tags = NULL, command = NULL, exe = NULL, args = NULL, environment = NULL, stdin = NULL, stdoutFile = NULL, stderrFile = NULL, workingDirectory = NULL, host = NULL, container = NULL, exposedPorts = NULL, mounts = NULL, placementConstraints = NULL, resourceLimits = NULL, queues = NULL, config = NULL, user = Sys.getenv("USER"), applyConfigSettings = TRUE)"
**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A descriptive name to assign to the job.</td>
</tr>
<tr>
<td>cluster</td>
<td>The name of the cluster this job should be submitted to.</td>
</tr>
<tr>
<td>tags</td>
<td>A set of user-defined tags, used for searching and querying jobs.</td>
</tr>
<tr>
<td>command</td>
<td>The command to run within the job. This is executed via the system shell.</td>
</tr>
<tr>
<td>exe</td>
<td>The (fully pathed) executable to run within the job. Only one of command or</td>
</tr>
<tr>
<td></td>
<td>exe should be specified.</td>
</tr>
<tr>
<td>args</td>
<td>An array of arguments to pass to the command / executable.</td>
</tr>
<tr>
<td>environment</td>
<td>A list of environment variables to be set for processes launched with this</td>
</tr>
<tr>
<td></td>
<td>job.</td>
</tr>
<tr>
<td>stdin</td>
<td>Data to be written to stdin when the job process is launched.</td>
</tr>
<tr>
<td>stdoutFile</td>
<td>The file used for the job’s generated standard output. Not all launcher</td>
</tr>
<tr>
<td></td>
<td>plugins support this parameter.</td>
</tr>
<tr>
<td>stderrFile</td>
<td>The file used for the job’s generated standard error. Not all launcher</td>
</tr>
<tr>
<td></td>
<td>plugins support this parameter.</td>
</tr>
<tr>
<td>workingDirectory</td>
<td>The working directory to be used by the command / executable associated</td>
</tr>
<tr>
<td></td>
<td>with this job.</td>
</tr>
<tr>
<td>host</td>
<td>The host that the job is running on, or the desired host during job submission.</td>
</tr>
<tr>
<td>container</td>
<td>The container to be used for launched jobs.</td>
</tr>
<tr>
<td>exposedPorts</td>
<td>The ports that are exposed by services running on a container. Only applicable</td>
</tr>
<tr>
<td></td>
<td>to systems that support containers.</td>
</tr>
<tr>
<td>mounts</td>
<td>A list of mount points. See <code>launcherHostMount()</code> and <code>launcherNfsMount()</code></td>
</tr>
<tr>
<td></td>
<td>for more information.</td>
</tr>
<tr>
<td>placementConstraints</td>
<td>A list of placement constraints. See <code>launcherPlacementConstraint()</code> for</td>
</tr>
<tr>
<td></td>
<td>more information.</td>
</tr>
<tr>
<td>resourceLimits</td>
<td>A list of resource limits. See <code>launcherResourceLimit()</code> for more information.</td>
</tr>
<tr>
<td>queues</td>
<td>A list of available submission queues for the cluster. Only applicable to</td>
</tr>
<tr>
<td></td>
<td>batch systems like LSF.</td>
</tr>
<tr>
<td>config</td>
<td>A list of cluster-specific configuration options. See <code>launcherConfig()</code> for</td>
</tr>
<tr>
<td></td>
<td>more information.</td>
</tr>
<tr>
<td>user</td>
<td>The user-name of the job owner.</td>
</tr>
<tr>
<td>applyConfigSettings</td>
<td>Apply server-configured mounts, exposedPorts, and environment, in addition to</td>
</tr>
<tr>
<td></td>
<td>any specified in this call.</td>
</tr>
</tbody>
</table>

**See Also**

Other job submission: `launcherConfig`, `launcherContainer`, `launcherHostMount`, `launcherNfsMount`, `launcherPlacementConstraint`, `launcherResourceLimit`, `launcherSubmitR`
launcherSubmitR  Execute an R Script as a Launcher Job

Description
Convenience function for running an R script as a launcher job using whichever R is found on the path in the launcher cluster.

Usage
launcherSubmitR(script, cluster = "Local", container = NULL)

Arguments
- **script**  Fully qualified path of R script. Must be a path that is available in the job container (if using containerized job cluster such as Kubernetes).
- **cluster**  The name of the cluster this job should be submitted to.
- **container**  The container to be used for launched jobs.

Details
See `launcherSubmitJob()` for running jobs with full control over command, environment, and so forth.

See Also
Other job submission: `launcherConfig`, `launcherContainer`, `launcherHostMount`, `launcherNfsMount`, `launcherPlacementConstraint`, `launcherResourceLimit`, `launcherSubmitJob`

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. If the `line` and `column` arguments are both equal to `-1L` (the default), then the cursor position in the document that is opened will be preserved.

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.

---

### persistent-values

**Persistent Keys and Values**

**Description**

Store persistent keys and values. Storage is per-project, if there is no project currently active then a global store is used.

**Usage**

```r
setPersistentValue(name, value)
getPersistentValue(name)
```

**Arguments**

- `name` Key name
- `value` Key value

**Value**

The stored value as a character vector (NULL if no value of the specified name is available).

**Note**

The `setPersistentValue` and `getPersistentValue` functions were added in version 1.1.57 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**

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

## End(Not run)
```

---

**previewSql**

*Preview SQL statement*

**Description**

Makes use of 'DBI' and `dbGetQuery()` to preview a SQL statement for a given 'DBI' connection.

**Usage**

`previewSql(conn, statement, ...)`

**Arguments**

- `conn` The 'DBI' connection to be used to execute this statement.
- `statement` The SQL statement to execute. Either a path to a file containing a SQL statement or the SQL statement itself.
- `...` Additional arguments to be used in `dbGetQuery()`.

**Note**

The `previewSql` function was introduced in RStudio 1.2.600
primary_selection  

*Extract the Primary Selection*

**Description**

By default, functions returning a document context will return a list of selections, including both the 'primary' selection and also 'other' selections (e.g. to handle the case where a user might have multiple cursors active). Use `primary_selection()` to extract the primary selection.

**Usage**

```r
primary_selection(x, ...)
```

**Arguments**

- **x**  A document context, or a selection.
- **...**  Optional arguments (currently ignored).

**projects**  

*Open a Project in RStudio*

**Description**

Initialize and open RStudio projects.

**Usage**

```r
openProject(path = NULL, newSession = FALSE)
initializeProject(path = getwd())
```

**Arguments**

- **path**  Either the path to an existing .Rproj file, or a path to a directory in which a new project should be initialized and opened.
- **newSession**  Boolean; should the project be opened in a new session, or should the current RStudio session switch to that project? Note that TRUE values are only supported with RStudio Desktop and RStudio Server Pro.

**Details**

Calling `openProject()` without arguments effectively re-opens the currently open project in RStudio. When switching projects, users will be prompted to save any unsaved files; alternatively, you can explicitly save any open documents using `documentSaveAll()`.
Note

The `openProject` and `initializeProject` functions were added in version 1.1.287 of RStudio.

**readPreference**  
*Read Preference*

**Description**

Reads a user interface preference, useful to remember preferences across different R sessions for the same user.

**Usage**

`readPreference(name, default)`

**Arguments**

- `name`  
The name of the preference.
- `default`  
The default value to use when the preference is not available.

**Note**

The `readPreference` function was added in version 1.1.67 of RStudio.

**removeTheme**  
*Remove a custom theme from RStudio.*

**Description**

Remove a custom theme from RStudio.

**Usage**

`removeTheme(name)`

**Arguments**

- `name`  
The unique name of the theme to remove.

**Note**

The `removeTheme` function was introduced in RStudio 1.2.879.
restartSession  

*Restart the R Session*

**Description**

Restart the RStudio R session.

**Usage**

```r
restartSession(command = "")
```

**Arguments**

- `command`  
  An R command (as a string) to be run after restarting R.

**Note**

The `restartSession` function was added in version 1.1.281 of RStudio.

---

rstudio-documents  

*Interact with Documents open in RStudio*

**Description**

Use these functions to interact with documents open in RStudio.

- Creates a new document in RStudio
- Closes a document currently open in RStudio

**Usage**

```r
insertText(location, text, id = NULL)
modifyRange(location, text, id = NULL)
setDocumentContents(text, id = NULL)
setCursorPosition(position, id = NULL)
setSelectionRanges(ranges, id = NULL)
documentSave(id = NULL)
documentSaveAll()
documentNew(text, type = c("r", "rmarkdown", "sql"))
```
position = document_position(0, 0), execute = FALSE)

documentClose(id = NULL, save = TRUE)

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.
position  The cursor position, typically created through `document_position()`.
ranges  A list of one or more ranges, typically created through `document_range()`.
type  The type of document to be created.
execute  Should the code be executed after the document is created?
save  Whether to commit unsaved changes to the document before closing it.

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_positions, and modifyRange when working with document_ranges.

documentClose accepts an ID of an open document rather than a path. You can get the ID of an open document from the getSourceEditorContext function, among others.

Closing is always done non-interactively; that is, no prompts are given to the user. If the user has made changes to the document but not saved them, then the save parameter governs the behavior: when TRUE, unsaved changes are committed, and when FALSE they are discarded.

Note

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

The setCursorPosition and setSelectionRanges functions were added with version 0.99.1111 of RStudio.

The documentSave and documentSaveAll functions were added with version 1.1.287 of RStudio.

The documentNew function was introduced in RStudio 1.2.640

The documentClose function was introduced in RStudio 1.2.1255

rstudio-editors

Retrieve Information about an RStudio Editor

Description

Returns information about an RStudio editor.

Usage

getActiveDocumentContext()

getSourceEditorContext()

getConsoleEditorContext()

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, while the `getSourceEditorContext` and the `getConsoleEditorContext` functions were added with version 0.99.1111.

---

**savePlotAsImage**

**Save Active RStudio Plot as an Image**

**Description**

Save the currently active RStudio as an image file.

**Usage**

```r
savePlotAsImage(file, format = c("png", "jpeg", "bmp", "tiff", "emf", "svg", "eps"), width, height)
```

**Arguments**

- **file**: Target filename
- **format**: Image format ("png", "jpeg", "bmp", "tiff", "emf", "svg", or "eps")
- **width**: Image width in pixels
- **height**: Image height in pixels

**Note**

The `savePlotAsImage` function was introduced in RStudio 1.1.57
### sendToConsole

**Send Code to the R Console**

**Description**

Send code to the R console and optionally execute it.

**Usage**

```r
sendToConsole(code, execute = TRUE, echo = TRUE, focus = TRUE)
```

**Arguments**

- `code` Character vector containing code to be executed.
- `execute` Boolean; execute the code immediately or just enter the text into the console?
- `echo` Boolean; echo the R code in the console as it’s executed?
- `focus` Boolean; focus the R console after sending code?

**Note**

The `sendToConsole` function was added in version 0.99.787 of RStudio.

**Examples**

```r
## Not run:
rstudioapi::sendToConsole(".Platform", execute = TRUE)

## End(Not run)
```

### showDialog

**Show Dialog Box**

**Description**

Shows a dialog box with a given title and contents.

**Usage**

```r
showDialog(title, message, url = "")
```

**Arguments**

- `title` The title to display in the dialog box.
- `message` A character vector with the contents to display in the main dialog area. Contents can contain the following HTML tags: "p", "em", "strong", "b" and "i".
- `url` An optional url to display under the message.
showPrompt

Details

showDialog("A dialog", "Showing <b>bold</b> text in the message.")

Note

The showDialog function was added in version 1.1.67 of RStudio.

---

showPrompt  
Show Prompt Dialog Box

Description

Shows a dialog box with a prompt field.

Usage

showPrompt(title, message, default = NULL)

Arguments

title The title to display in the dialog box.
message A character vector with the contents to display in the main dialog area.
default An optional character vector that fills the prompt field with a default value.

Note

The showPrompt function was added in version 1.1.67 of RStudio.

---

showQuestion  
Show Question Dialog Box

Description

Shows a dialog box asking a question.

Usage

showQuestion(title, message, ok = NULL, cancel = NULL)

Arguments

title The title to display in the dialog box.
message A character vector with the contents to display in the main dialog area.
ok And optional character vector that overrides the caption for the OK button.
cancel An optional character vector that overrides the caption for the Cancel button.
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):

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Marker type (&quot;error&quot;, &quot;warning&quot;, &quot;info&quot;, &quot;style&quot;, or &quot;usage&quot;)</td>
</tr>
<tr>
<td>file</td>
<td>Path to source file</td>
</tr>
<tr>
<td>line</td>
<td>Line number within source file</td>
</tr>
<tr>
<td>column</td>
<td>Column number within line</td>
</tr>
<tr>
<td>message</td>
<td>Short descriptive message</td>
</tr>
</tbody>
</table>

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.
terminalActivate

Activate Terminal

Description

Ensure terminal is running and optionally bring to front in RStudio.

Usage

```r
terminalActivate(id = NULL, show = TRUE)
```

Arguments

- `id` The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`. If NULL, the terminal tab will be selected but no specific terminal will be chosen.
- `show` If TRUE, bring the terminal to front in RStudio.

Note

The `terminalActivate` function was added in version 1.1.350 of RStudio.

Examples

```r
## Not run:
# create a hidden terminal and run a lengthy command
termId = rstudioapi::terminalCreate(show = FALSE)
rstudioapi::terminalSend(termId, "sleep 5\n")

# wait until a busy terminal is finished
while (rstudioapi::terminalBusy(termId)) {
  Sys.sleep(0.1)
}
print("Terminal available")##'

rstudioapi::terminalActivate(termId)

## End(Not run)
```
### terminalBuffer

**Get Terminal Buffer**

**Description**

Returns contents of a terminal buffer.

**Usage**

```r
terminalBuffer(id, stripAnsi = TRUE)
```

**Arguments**

- `id`  
  The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`.

- `stripAnsi`  
  If FALSE, don’t strip out Ansi escape sequences before returning terminal buffer.

**Value**

The terminal contents, one line per row.

**Note**

The `terminalBuffer` function was added in version 1.1.350 of RStudio.

---

### terminalBusy

**Is Terminal Busy**

**Description**

Are terminals reporting that they are busy?

**Usage**

```r
terminalBusy(id)
```

**Arguments**

- `id`  
  The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`.

**Value**

a boolean
Note

The terminalBusy function was added in version 1.1.350 of RStudio.

Examples

```r
## Not run:
# create a hidden terminal and run a lengthy command
termId <- rstudioapi::terminalCreate(show = FALSE)
rstudioapi::terminalSend(termId, "sleep 5\n")

# wait until a busy terminal is finished
while (rstudioapi::terminalBusy(termId)) {
  Sys.sleep(0.1)
}
print("Terminal available")

## End(Not run)
```

---

**terminalClear**  
*Clear Terminal Buffer*

Description

Clears the buffer for specified terminal.

Usage

```r
terminalClear(id)
```

Arguments

- **id**  
The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`.

Note

The `terminalClear` function was added in version 1.1.350 of RStudio.

Examples

```r
## Not run:
termId <- rstudioapi::terminalCreate()
rstudioapi::terminalSend(termId, 'ls -l\n')
Sys.sleep(3)
rstudioapi::terminalClear(termId)

## End(Not run)
```
terminalContext Retrieve Information about RStudio Terminals

**Description**

Returns information about RStudio terminal instances.

**Usage**

`terminalContext(id)`

**Arguments**

- `id` The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`.

**Value**

A list with elements:

- `handle` the internal handle
- `caption` caption
- `title` title set by the shell
- `working_dir` working directory
- `shell` shell type
- `running` is terminal process executing
- `busy` is terminal running a program
- `exit_code` process exit code or NULL
- `connection` websockets or rpc
- `sequence` creation sequence
- `lines` lines of text in terminal buffer
- `cols` columns in terminal
- `rows` rows in terminal
- `pid` process id of terminal shell
- `full_screen` full screen program running

**Note**

The `terminalContext` function was added in version 1.1.350 of RStudio.

**Examples**

```r
## Not run:
termId <- rstudioapi::terminalCreate("example", show = FALSE)
View(rstudioapi::terminalContext(termId))
```
## terminalCreate

Create a Terminal

### Description
Create a new Terminal.

### Usage
```r
terminalCreate(caption = NULL, show = TRUE, shellType = NULL)
```

### Arguments
- **caption**
  The desired terminal caption. When NULL or blank, the terminal caption will be chosen by the system.
- **show**
  If FALSE, terminal won’t be brought to front.
- **shellType**
  Shell type for the terminal: NULL or "default" to use the shell selected in Global Options. For Microsoft Windows, alternatives are "win-cmd" for 64-bit Command Prompt, "win-ps" for 64-bit PowerShell, "win-git-bash" for Git Bash, or "win-wsl-bash" for Bash on Windows Subsystem for Linux. On Linux, Mac, and RStudio Server "custom" will use the custom terminal defined in Global Options. If the requested shell type is not available, the default shell will be used, instead.

### Value
The terminal identifier as a character vector (NULL if unable to create the terminal or the given terminal caption is already in use).

### Note
The `terminalCreate` function was added in version 1.1.350 of RStudio and the ability to specify `shellType` was added in version 1.2.696.

### Examples
```r
## Not run:
termId <- rstudioapi::terminalCreate('My Terminal')

## End(Not run)
```
terminalExecute 

Execute Command

Description

Execute a command, showing results in the terminal pane.

Usage

```r
terminalExecute(command, workingDir = NULL, env = character(), show = TRUE)
```

Arguments

- `command`: System command to be invoked, as a character string.
- `workingDir`: Working directory for command.
- `env`: Vector of name=value strings to set environment variables.
- `show`: If FALSE, terminal won’t be brought to front.

Value

The terminal identifier as a character vector (NULL if unable to create the terminal).

Note

The `terminalExecute` function was added in version 1.1.350 of RStudio.

Examples

```r
## Not run:
termId <- rstudioapi::terminalExecute(
  command = 'echo $HELLO && echo $WORLD',
  workingDir = '/usr/local',
  env = c('HELLO=WORD', 'WORLD=EARTH'),
  show = FALSE)

while (is.null(rstudioapi::terminalExitCode(termId))) {
  Sys.sleep(0.1)
}

result <- terminalBuffer(termId)
terminalKill(termId)
print(result)

## End(Not run)
```
### terminalExitCode

**Terminal Exit Code**

#### Description

Get exit code of terminal process, or NULL if still running.

#### Usage

```r
terminalExitCode(id)
```

#### Arguments

- **id**
  
  The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`.

#### Value

The exit code as an integer vector, or NULL if process still running.

#### Note

The `terminalExitCode` function was added in version 1.1.350 of RStudio.

### terminalKill

**Kill Terminal**

#### Description

Kill processes and close a terminal.

#### Usage

```r
terminalKill(id)
```

#### Arguments

- **id**
  
  The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`.

#### Note

The `terminalKill` function was added in version 1.1.350 of RStudio.
terminalList \hspace{1cm} Get All Terminal Ids

Description

Return a character vector containing all the current terminal identifiers.

Usage

terminalList()

Value

The terminal identifiers as a character vector.

Note

The `terminalList` function was added in version 1.1.350 of RStudio.

terminalRunning \hspace{1cm} Is Terminal Running

Description

Does a terminal have a process associated with it? If the R session is restarted after a terminal has been created, the terminal will not restart its shell until it is displayed either via the user interface, or via `terminalActivate()`.

Usage

terminalRunning(id)

Arguments

id \hspace{1cm} The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`.

Value

a boolean

Note

The `terminalRunning` function was added in version 1.1.350 of RStudio.
terminalSend

Examples

```r
## Not run:
# termId has a handle to a previously created terminal
# make sure it is still running before we send it a command
if (!rstudioapi::terminalRunning(termId)) {
  rstudioapi::terminalActivate(termId)
}

# wait for it to start
while (!rstudioapi::terminalRunning(termId)) {
  Sys.sleep(0.1)
}

terminalSend(termId, "echo Hello\n")
```

## End(Not run)

---

**Send Text to a Terminal**

**Description**

Send text to an existing terminal.

**Usage**

```r
terminalSend(id, text)
```

**Arguments**

- `id` : The terminal id. The id is obtained from `terminalList()`, `terminalVisible()`, `terminalCreate()`, or `terminalExecute()`.
- `text` : Character vector containing text to be inserted.

**Note**

The `terminalSend` function was added in version 1.1.350 of RStudio.

**Examples**

```r
## Not run:
termId <- rstudioapi::terminalCreate()
rstudioapi::terminalSend(termId, 'ls -l\n')
```

## End(Not run)
terminalvisible  

**Get Visible Terminal**

**Description**
Get Visible Terminal

**Usage**
terminalvisible()

**Value**
Terminal identifier selected in the client, if any.

**Note**
The terminalvisible function was added in version 1.1.350 of RStudio.

translateLocalUrl  

**Translate Local URL**

**Description**
Translates a local URL into an externally accessible URL on RStudio Server.

**Usage**
translateLocalUrl(url, absolute = FALSE)

**Arguments**
- **url**: The fully qualified URL to translate; for example, http://localhost:1234/service/page.html.
- **absolute**: Whether to return a relative path URL (the default) or an absolute URL.

**Details**
On RStudio Server, URLs which refer to the local host network address (such as http://localhost:1234/ and http://127.0.0.1:5678/) must be translated in order to be externally accessible from a browser. This method performs the required translation, and returns the translated URL, which RStudio Server uses to proxy HTTP requests.

Returns an unmodified URL on RStudio Desktop, and when the URL does not refer to a local address.

**Value**
The translated URL.
updateDialog  

*Updates a Dialog Box*

**Description**

Updates specific properties from the current dialog box.

**Usage**

```r
updateDialog(...)```

**Arguments**

...  

Named parameters and values to update a dialog box.

**Details**

Currently, the only dialog with support for this action is the New Connection dialog in which the code preview can be updated through this API.

```r
updateDialog(code = "con <- NULL")```

**Note**

The `updateDialog` function was added in version 1.1.67 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**

```r
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 `packageVersion`

- `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**

```r
## 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

```r
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)
writePreference

# 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)

---

**writePreference**  
*Write Preference*

**Description**

Writers a user interface preference, useful to remember preferences across different R sessions for the same user.

**Usage**

writePreference(name, value)

**Arguments**

- **name**  
The name of the preference.
- **value**  
The value of the preference.

**Note**

The `writePreference` function was added in version 1.1.67 of RStudio.
Index

addTheme, 3
applyTheme, 4
asNdocument_position
  (document_position), 9
asNdocument_range (document_range), 10
askForPassword, 4
askForSecret, 5
build-tools, 5
buildToolsCheck (build-tools), 5
buildToolsExec (build-tools), 5
buildToolsInstall (build-tools), 5
callFun, 6
convertTheme, 7
createProjectTemplate, 8
dictionaries, 8
dictionariesPath (dictionaries), 8
document_position, 9, 10, 33, 34
document_range, 10, 33, 34
documentClose (rstudio-documents), 32
documentNew (rstudio-documents), 32
documentSave (rstudio-documents), 32
documentSaveAll, 30
documentSaveAll (rstudio-documents), 32
generateCommand, 10
exists, 15
filedialogs, 11
findFun (hasFun), 15
get, 15
getActiveDocumentContext, 33
getActiveDocumentContext
  (rstudio-editors), 34
getActiveProject, 12
getConsoleEditorContext
  (rstudio-editors), 34
generateValue (persistent-values), 28
generateSourceEditorContext
  (rstudio-editors), 34
getThemeInfo, 12
getThemes, 13
getVersion, 13
hasColorConsole, 14
hasFun, 6, 15
initializeProject (projects), 30
insertText (rstudio-documents), 32
isNdocument_position
  (document_position), 9
isNdocument_range (document_range), 10
isAvailable, 15
jobAdd, 16, 17–20
jobAddOutput, 17, 17, 18–20
jobAddProgress, 17, 17, 18–20
jobRemove, 17, 18, 19, 20
jobRunScript, 17, 18, 18, 19, 20
jobSetProgress, 17–19, 19, 20
jobSetState, 17–19, 19, 20
jobSetStatus, 17–20, 20
launcher, 21
launcherAvailable (launcher), 21
launcherConfig, 21, 22–27
launcherConfig(), 26
launcherContainer, 22, 22, 23–27
launcherControlJob, 22
launcherGetInfo (launcher), 21
launcherGetJob, 23
launcherGetJobs (launcher), 21
launcherHostMount, 22, 23, 24–27
launcherHostMount(), 26
launcherNfsMount, 22–24, 24, 25–27
launcherNfsMount(), 26
launcherPlacementConstraint, 22–24, 24–27
launcherPlacementConstraint(), 26
launcherResourceLimit, 22–24, 25, 26, 27
launcherResourceLimit(), 26
launcherSubmitJob, 22–25, 25–27
launcherSubmitJob(), 21–25, 27
launcherSubmitR, 22–26, 27
modifyRange (rstudio-documents), 32
navigateToFile, 27
numeric_version, 13
OpenCPU, 50
openProject (projects), 30
persistent-values, 28
previewRd, 29
previewSql, 29
primary_selection, 30
projects, 30
readPreference, 31
removeTheme, 31
restartSession, 32
Rook, 50
rstudio-documents, 32
rstudio-editors, 34
savePlotAsImage, 35
selectDirectory (file-dialogs), 11
selectFile (file-dialogs), 11
sendToConsole, 36
setCursorPosition (rstudio-documents), 32
setDocumentContents (rstudio-documents), 32
setPersistentValue (persistent-values), 28
setSelectionRanges (rstudio-documents), 32
Shiny, 50
showDialog, 36
showPrompt, 37
showQuestion, 37
sourceMarkers, 38
tempfile, 50
terminalActivate, 39, 46
terminalBuffer, 40
terminalBusy, 40
terminalClear, 41
terminalContext, 42
terminalCreate, 39–42, 43, 45–47
terminalExecute, 39–42, 44, 45–47
terminalExitCode, 45
terminalKill, 45
terminalList, 39–42, 45, 46, 47
terminalRunning, 46
terminalSend, 47
terminalVisible, 39–42, 45–47, 48
translateLocalUrl, 48
updateDialog, 49
userDictionariesPath (dictionaries), 8
verifyAvailable (isAvailable), 15
versionInfo, 49
View, 28
viewer, 50
writePreference, 52