Package 'jsonStrings'

April 5, 2022

Type Package

Title Manipulation of JSON Strings

Version 2.0.0

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Fast manipulation of JSON strings. Allows to extract or delete an element in a JSON string, merge two JSON strings, and more.

License GPL (>= 2)

Imports Rcpp (>= 1.0.0), methods, R6

LinkingTo Rcpp

RoxygenNote 7.1.2

Encoding UTF-8

SystemRequirements C++11

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

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

NeedsCompilation yes

Author Stéphane Laurent [aut, cre],
Niels Lohmann [cph] ('nlohmann/json' C++ library)

Repository CRAN

Date/Publication 2022-04-05 14:52:30 UTC

R topics documented:

jsonString *R6 class to represent a JSON string*

Description

R6 class to represent a JSON string.

Active bindings

`prettyPrint` get or set the value of `prettyPrint`

Methods

Public methods:

- `jsonString$new()`
- `jsonString$print()`
- `jsonString$asString()`
- `jsonString$at()`
- `jsonString$hasKey()`
- `jsonString$keys()`
- `jsonString$addProperty()`
- `jsonString$erase()`
- `jsonString$size()`
- `jsonString$update()`
- `jsonString$merge()`
- `jsonString$patch()`
- `jsonString$push()`
- `jsonString$is()`
- `jsonString$type()`
- `jsonString$flatten()`
- `jsonString$unflatten()`
- `jsonString$writeFile()`
- `jsonString$copy()`

Method `new()`**:** Creates a new `jsonString` object.

Usage:

```
jsonString$new(string)
```

Arguments:

`string` a string representing a JSON object, or the path to a JSON file

Returns: A `jsonString` object.

Examples:

```
jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
jstring$prettyPrint
jstring
jstring$prettyPrint <- FALSE
jstring
```

```
jstring <- "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
jsonString$new(jstring)
```

Method `print()`**:** Print a `jsonString` object.

Usage:

```
jsonString$print(...)
```

Arguments:

`...` ignored

Examples:

```
jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
jstring
jstring$prettyPrint <- FALSE
jstring
```

Method `asString()`**:** Converts a `jsonString` to a character string.

Usage:

```
jsonString$asString(pretty = FALSE)
```

Arguments:

`pretty` Boolean, whether to get a pretty string

Returns: A string.

Examples:

```
jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
cat(jstring$asString())
cat(jstring$asString(pretty = TRUE))
```

Method `at()`**:** Extract an element in a JSON string by giving a path of keys or indices.

Usage:

```
jsonString$at(...)
```

Arguments:

`...` the elements forming the path, integers or strings

Returns: A `jsonString` object.

Examples:

```
jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
jstring$at(1)
jstring$at(2, "x")
```

Method `hasKey()`**:** Checks whether a key exists in the reference JSON string.

Usage:

```
jsonString$hasKey(key)
```

Arguments:

`key` a string

Returns: A Boolean value.

Examples:

```
jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
jstring$hasKey("x")
jstring <- jsonString$new(
  "{\"x\": [2,3,4], \"y\": 42}"
)
jstring$hasKey("x")
```

Method `keys()`**:** Get the keys of the reference JSON string (if it represents an object).

Usage:

```
jsonString$keys()
```

Returns: A character vector.

Examples:

```
jstring <- jsonString$new(
  "{\"x\": [2,3,4], \"y\": 42}"
)
jstring$keys()
```

Method `addProperty()`**:** Add a new property to the reference JSON string (if it represents an object).

Usage:

```
jsonString$addProperty(key, value)
```

Arguments:

`key` a character string, the key of the new property

`value` a JSON string, either a `jsonString` object or a string

Returns: This updates the reference JSON string and this returns it invisibly.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
ppty <- jsonString$new("[9, 99]")
jstring$addProperty("c", ppty)
jstring
jstring$addProperty("d", "null")
jstring
```

Method `erase()`**:** Erase an object property or an array element from the reference JSON string.

Usage:

```
jsonString$erase(at)
```

Arguments:

`at` either a character string, the key of the property to be erased, or an integer, the index of the array element to be erased

Returns: This invisibly returns the updated reference JSON string.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring$erase("b")
jstring
jstring <- jsonString$new("[1, 2, 3, 4, 5]")
jstring$erase(2)
jstring
```

Method `size()`**:** Number of elements in the reference JSON string.

Usage:

```
jsonString$size()
```

Returns: An integer.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring$size()
```

Method `update()`**:** Update the reference JSON string (if it represents an object).

Usage:

```
jsonString$update(jstring)
```

Arguments:

`jstring` a JSON string representing an object, either a `jsonString` object or a string

Returns: This invisibly returns the updated reference JSON string.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring2 <- "{\"a\":[4,5,6],\"c\":\"goodbye\"}"
jstring$update(jstring2)
jstring
```

Method `merge()`**:** Merge the reference JSON string (if it represents an object).

Usage:

```
jsonString$merge(jstring)
```

Arguments:

`jstring` a JSON string, either a `jsonString` object or a string representing a JSON object

Returns: This invisibly returns the updated reference JSON string.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring2 <- "{\"a\":[4,5,6],\"c\":\"goodbye\"}"
jstring$merge(jstring2)
jstring
```

Method `patch()`**:** Apply a JSON patch to the reference JSON string (if it represents an array or an object).

Usage:

```
jsonString$patch(jspatch)
```

Arguments:

`jspatch` a JSON patch, a JSON string representing an array (see the link in details); it could be either a `jsonString` object or a string

Details: See jsonpatch.com.

Returns: A new `jsonString` object.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jspatch <- "[
  {\"op\": \"remove\", \"path\": \"/a\"},
  {\"op\": \"replace\", \"path\": \"/b\", \"value\": null}
]"
jstring$patch(jspatch)
```

Method `push()`**:** Append an element to the reference JSON string (if it represents an array).

Usage:

```
jsonString$push(jstring)
```

Arguments:

`jstring` a JSON string, either a `jsonString` object or a string representing a JSON object

Returns: This invisibly returns the updated reference JSON string.

Examples:

```
jstring <- jsonString$new("[1, 2, 3, 4, 5]")
jstring2 <- jsonString$new(
  "{\"a\":[4,5,6],\"c\":\"goodbye\"}"
 )
jstring$push(jstring2)
jstring
```

Method `is()`**:** Check the type of the reference JSON string.

Usage:

```
jsonString$is(type)
```

Arguments:

`type` the type to be checked, one of `"array"`, `"object"`, `"string"`, `"number"`, `"integer"`, `"float"`, `"null"`, `"boolean"`

Returns: A Boolean value.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring$is("object")
jstring$is("array")
jstring <- jsonString$new("999")
jstring$is("integer")
jstring$is("number")
jstring$is("float")
```

Method `type()`**:** Get the type of the reference JSON string.

Usage:

```
jsonString$type()
```

Returns: A character string indicating the type of the JSON string.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring$type()
jstring <- jsonString$new("999")
jstring$type()
```

Method `flatten()`**:** Flatten the reference JSON string.

Usage:

```
jsonString$flatten()
```

Returns: A new `jsonString` object.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":{\"x\":\"hello\",\"y\":\"hi\"}}"
)
jstring$flatten()
```

Method `unflatten()`**:** Unflatten the reference JSON string (if it is flattened).

Usage:

```
jsonString$unflatten()
```

Returns: A new `jsonString` object.

Examples:

```
folder <- system.file(package = "jsonStrings")
files <- list.files(folder, recursive = TRUE)
sizes <- file.size(file.path(folder, files))
files <- sprintf('"%s"', paste0("/", files))
string <- sprintf("{%s}", paste0(files, ":", sizes, collapse = ","))
jstring <- jsonString$new(string)
jstring$unflatten()
```

Method `writeFile()`**:** Write the reference JSON string to a file.

Usage:

```
jsonString$writeFile(filename)
```

Arguments:

`filename` name of the file

Returns: Nothing.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jsonfile <- tempfile(fileext = ".json")
jstring$writeFile(jsonfile)
cat(readLines(jsonfile), sep = "\n")
jsonString$new(jsonfile)
```

Method `copy()`**:** Copy the reference JSON string.

Usage:

```
jsonString$copy()
```

Returns: A new `jsonString` object.

Examples:

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
copy <- jstring$copy()
copy$erase("b")
```

```
jstring
naive_copy <- jstring
naive_copy$erase("b")
jstring
```

Examples

```
## ------------------------------------------------
## Method `jsonString$new`
## ------------------------------------------------

jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
jstring$prettyPrint
jstring
jstring$prettyPrint <- FALSE
jstring
jstring <- "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
jsonString$new(jstring)

## ------------------------------------------------
## Method `jsonString$print`
## ------------------------------------------------

jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
jstring
jstring$prettyPrint <- FALSE
jstring

## ------------------------------------------------
## Method `jsonString$asString`
## ------------------------------------------------

jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
cat(jstring$asString())
cat(jstring$asString(pretty = TRUE))

## ------------------------------------------------
## Method `jsonString$at`
## ------------------------------------------------

jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
jstring$at(1)
jstring$at(2, "x")
```

```
## ------------------------------------------------
## Method `jsonString$hasKey`
## ------------------------------------------------

jstring <- jsonString$new(
  "[1, [\"a\", 99], {\"x\": [2,3,4], \"y\": 42}]"
)
jstring$hasKey("x")
jstring <- jsonString$new(
  "{\"x\": [2,3,4], \"y\": 42}"
)
jstring$hasKey("x")

## ------------------------------------------------
## Method `jsonString$keys`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"x\": [2,3,4], \"y\": 42}"
)
jstring$keys()

## ------------------------------------------------
## Method `jsonString$addProperty`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
ppty <- jsonString$new("[9, 99]")
jstring$addProperty("c", ppty)
jstring
jstring$addProperty("d", "null")
jstring

## ------------------------------------------------
## Method `jsonString$erase`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring$erase("b")
jstring
jstring <- jsonString$new("[1, 2, 3, 4, 5]")
jstring$erase(2)
jstring

## ------------------------------------------------
## Method `jsonString$size`
## ------------------------------------------------
```

```
jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring$size()

## ------------------------------------------------
## Method `jsonString$update`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring2 <- "{\"a\":[4,5,6],\"c\":\"goodbye\"}"
jstring$update(jstring2)
jstring

## ------------------------------------------------
## Method `jsonString$merge`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring2 <- "{\"a\":[4,5,6],\"c\":\"goodbye\"}"
jstring$merge(jstring2)
jstring

## ------------------------------------------------
## Method `jsonString$patch`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jspatch <- "[
  {\"op\": \"remove\", \"path\": \"/a\"},
  {\"op\": \"replace\", \"path\": \"/b\", \"value\": null}
]"
jstring$patch(jspatch)

## ------------------------------------------------
## Method `jsonString$push`
## ------------------------------------------------

jstring <- jsonString$new("[1, 2, 3, 4, 5]")
jstring2 <- jsonString$new(
  "{\"a\":[4,5,6],\"c\":\"goodbye\"}"
 )
jstring$push(jstring2)
jstring

## ------------------------------------------------
## Method `jsonString$is`
```

```
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring$is("object")
jstring$is("array")
jstring <- jsonString$new("999")
jstring$is("integer")
jstring$is("number")
jstring$is("float")

## ------------------------------------------------
## Method `jsonString$type`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jstring$type()
jstring <- jsonString$new("999")
jstring$type()

## ------------------------------------------------
## Method `jsonString$flatten`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":{\"x\":\"hello\",\"y\":\"hi\"}}"
)
jstring$flatten()

## ------------------------------------------------
## Method `jsonString$unflatten`
## ------------------------------------------------

folder <- system.file(package = "jsonStrings")
files <- list.files(folder, recursive = TRUE)
sizes <- file.size(file.path(folder, files))
files <- sprintf('"%s"', paste0("/", files))
string <- sprintf("{%s}", paste0(files, ":", sizes, collapse = ","))
jstring <- jsonString$new(string)
jstring$unflatten()

## ------------------------------------------------
## Method `jsonString$writeFile`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
jsonfile <- tempfile(fileext = ".json")
jstring$writeFile(jsonfile)
```

```
cat(readLines(jsonfile), sep = "\n")
jsonString$new(jsonfile)

## ------------------------------------------------
## Method `jsonString$copy`
## ------------------------------------------------

jstring <- jsonString$new(
  "{\"a\":[1,2,3],\"b\":\"hello\"}"
)
copy <- jstring$copy()
copy$erase("b")
jstring
naive_copy <- jstring
naive_copy$erase("b")
jstring
```

Index