Package 'jsonify'

January 10, 2019

Type Package

Title Converts 'R' Objects to Javascript Object Notation (JSON)

Version 0.2.0

Date 2019-01-10

Description Converts 'R' objects into Javascript Object Notation (JSON) using the 'rapidjsonr' library <https://CRAN.R-project.org/package=rapidjsonr>.

License GPL (>= 2)

Imports Rcpp (>= 0.12.18)

LinkingTo BH, rapidjsonr, Rcpp

RoxygenNote 6.1.0

Suggests covr, microbenchmark, jsonlite, testthat, knitr, rmarkdown

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation yes

Author David Cooley [aut, cre]

Maintainer David Cooley <dcooley@symbolix.com.au>

Repository CRAN

Date/Publication 2019-01-10 13:10:03 UTC

R topics documented:

minify_json *Minify Json*

Description

Removes indentiation from a JSON string

Usage

```
minify_json(json, ...)
```

Arguments

json	string of JSON
...	other argments passed to to_json

Examples

```
df <- data.frame(id = 1:10, val = rnorm(10))
js <- to_json( df )
jsp <- pretty_json(js)
minify_json( jsp )
```

pretty_json *Pretty Json*

Description

Adds indentiation to a JSON string

Usage

```
pretty_json(json, ...)
```

Arguments

json	string of JSON
...	other argments passed to to_json

Examples

```
df <- data.frame(id = 1:10, val = rnorm(10))
js <- to_json( df )
pretty_json(js)

## can also use directly on an R object
pretty_json( df )
```

`to_json` *To JSON*

Description

Converts R objects to JSON

Usage

```
to_json(x, unbox = FALSE, digits = NULL, numeric_dates = TRUE,
  factors_as_string = TRUE, by = "row")
```

Arguments

`x`	object to convert to JSON
`unbox`	logical indicating if single-value arrays should be 'unboxed', that is, not contained inside an array.
`digits`	integer specifying the number of decimal places to round numerics. Default is `NULL` - no rounding
`numeric_dates`	logical indicating if dates should be treated as numerics. Defaults to TRUE for speed. If FALSE, the dates will be coerced to character in UTC time zone
`factors_as_string`	logical indicating if factors should be treated as strings. Defaults to TRUE.
`by`	either "row" or "column" indicating if data.frames and matrices should be processed row-wise or column-wise. Defaults to "row"

Examples

```
to_json(1:3)
to_json(letters[1:3])
to_json(data.frame(x = 1:3, y = letters[1:3]))
to_json(list(x = 1:3, y = list(z = letters[1:3])))
to_json(seq(as.Date("2018-01-01"), as.Date("2018-01-05"), length.out = 5))
to_json(seq(as.Date("2018-01-01"), as.Date("2018-01-05"), length.out = 5), numeric_dates = FALSE)

psx <- seq(
```

```
  as.POSIXct("2018-01-01", tz = "Australia/Melbourne"),
  as.POSIXct("2018-02-01", tz = "Australia/Melbourne"),
  length.out = 5
  )
to_json(psx)
to_json(psx, numeric_dates = FALSE)

## unbox single-value arrays
to_json(list(x = 1), unbox = TRUE)
to_json(list(x = 1, y = c("a"), z = list(x = 2, y = c("b"))), unbox = TRUE)

## rounding numbers using the digits argument
to_json(1.23456789, digits = 2)
df <- data.frame(x = 1L:3L, y = rnorm(3), z = letters[1:3])
to_json(df, digits = 0 )

## keeping factors
to_json(df, digits = 2, factors_as_string = FALSE )
```

`validate_json` *validate JSON*

Description

Validates JSON

Usage

```
validate_json(json)
```

Arguments

`json`	character or json object

Value

logical vector

Examples

```
validate_json('[]')
df <- data.frame(id = 1:5, val = letters[1:5])
validate_json( to_json(df) )

validate_json('{"x":1,"y":2,"z":"a"}')

validate_json( c('{"x":1,"y":2,"z":"a"}', to_json(df) ) )
```

```
validate_json( c('{"x":1,"y":2,"z":a}', to_json(df) ) )
```

Index