Package ‘webdeveloper’

February 5, 2021

Type Package
Title Functions for Web Development
Version 0.1.0
Author Timothy Conwell
Maintainer Timothy Conwell <timconwell@gmail.com>
Description Organizational framework for web development in R including functions to serve static and dynamic content via HTTP methods, includes the html5 package to create HTML pages, and offers other utility functions for common tasks related to web development.
License GPL (>= 2)
Encoding UTF-8
LazyData true
Depends httpuv, html5
Imports stringi
Suggests dbWebForms, pgTools
RoxygenNote 7.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2021-02-05 09:40:05 UTC

R topics documented:

  castDateString .................................................. 2
  castNumeric ..................................................... 2
  doubleQuoteText ................................................ 3
  endServer ....................................................... 3
  fromInput ....................................................... 4
  jsonStr .......................................................... 5
  paramList ....................................................... 5
  quoteText ........................................................ 6
  sampleStr ....................................................... 6
castDateString

Format a date string as " from a SQL database to a format compatible with a HTML date input value.

Usage

castDateString(x)

Arguments

x

A string.

Value

A string, formatted YYYY-MM-DD.

Examples

castDateString(Sys.time())

castNumeric

Convert strings to numeric if possible, otherwise remains as is.

Description

Convert strings to numeric if possible, otherwise remains as is.

Usage

castNumeric(x)

Arguments

x

A string.
**doubleQuoteText**

**Value**

A string, converted to numeric if possible.

**Examples**

```r
castNumeric("100")
```

---

**endServer**

Stop HTTP server(s) by calling `httpuv::stopServer()` or `httpuv::stopAllServers()`.

**Description**

Stop HTTP server(s) by calling `httpuv::stopServer()` or `httpuv::stopAllServers()`.

**Usage**

```r
doubleQuoteText(x = NULL, all = FALSE)
```
fromInput

Arguments

  x  A server object that was previously returned from serveHTTP.
  all TRUE/FALSE, if TRUE, calls httpuv::stopAllServers.

Value

Nothing.

Examples

endServer(all = TRUE)

fromInput

Prepare values collected from HTML forms to save to a SQL database by calling quoteText. If x is "", returns "NULL".

Description

Prepare values collected from HTML forms to save to a SQL database by calling quoteText. If x is "", returns "NULL".

Usage

fromInput(x)

Arguments

  x  A vector of length 1.

Value

A string, if x is "", returns "NULL".

Examples

fromInput("Test")
fromInput("100")
fromInput(100)
fromInput(""")
### jsonStr

Format data as a JSON object (like this: “x”: “120”).

**Description**

Format data as a JSON object (like this: “x”: “120”).

**Usage**

```r
jsonStr(name, val)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A string, the name of the JSON entry</td>
</tr>
<tr>
<td>val</td>
<td>A string, the value to associate with the JSON entry.</td>
</tr>
</tbody>
</table>

**Value**

A string, data formatted as a JSON object.

**Examples**

```r
jsonStr(name = "var1", val = "Blue")
```

---

### paramList

Parse HTTP parameter strings.

**Description**

Parse HTTP parameter strings.

**Usage**

```r
paramList(
  x,
  split = ",",
  custom_decode = list(pattern = c("+"), replacement = c(" "))
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A parameter string, likely accessed from req[&quot;rook.input&quot;]$read_lines().</td>
</tr>
<tr>
<td>split</td>
<td>The character to use to split the parameter string into constituent parameters.</td>
</tr>
<tr>
<td>custom_decode</td>
<td>A named list, must consist of list(pattern = c(...), replacement = c(...)) where pattern contains characters to decode that are not included in utils::URLEncode and replacement contains the character to replace the character passed in the same indexed position in pattern.</td>
</tr>
</tbody>
</table>
Value

A list, with names being parameter names and values being parameter values.

Examples

```
paramList("?param1=Test&param2=1234&param3=Example")
```

quoteText

Add single quotes to strings, useful for converting R strings into SQL formatted strings.

Description

Add single quotes to strings, useful for converting R strings into SQL formatted strings.

Usage

```
quoteText(x, char_only = TRUE)
```

Arguments

- **x**  A string.
- **char_only**  TRUE/FALSE, if TRUE, adds quotes only if `is.character(x)` is TRUE.

Value

A string, with single quotes added to match PostgreSQL string formatting.

Examples

```
quoteText("Sample quotes.")
```

sampleStr

Generates (pseudo)random strings of the specified char length.

Description

Generates (pseudo)random strings of the specified char length.

Usage

```
sampleStr(char)
```

Arguments

- **char**  A integer, the number of chars to include in the output string.
Value

A string.

Examples

    sampleStr(10)

---

serveHTTP

Conveniently create HTTP server using httpuv::startServer() or httpuv::runServer().

Description

Conveniently create HTTP server using httpuv::startServer() or httpuv::runServer().

Usage

    serveHTTP(
        host = "127.0.0.1",
        port = 5001,
        persistent = FALSE,
        static = list(),
        dynamic = list()
    )

Arguments

host  A string that is a valid IPv4 or IPv6 address that is owned by this server, which
      the application will listen on. "0.0.0.0" represents all IPv4 addresses and ":/:0"
      represents all IPv6 addresses. Refer to host parameter of httpuv::startServer() for
      more details.

port  The port number to listen on. Refer to port parameter of httpuv::startServer() for
      more details.

persistent  TRUE/FALSE. If FALSE, calls httpuv::startServer(), which returns back to the
            R session (and would therefore not work with launching a persistent server
            through a system service as the R session would continue and likely exit/end). If
            TRUE, calls httpuv::runServer(), which does not return to the R session unless
            an error or interruption occurs and is suitable for use with system services to
            start or stop a server.

static  A named list, names should be URL paths, values should be paths to the files to
        be served statically (such as a HTML file saved somewhere).

dynamic  A named list, names should be URL paths, values should be named vectors
         with vector names equaling a HTTP method (such as "GET" or "POST") and
         the values being expressions that when evaluated return a named list with valid
         entries for status, headers, and body as specified by httpuv::startServer(). Re-
         fer to httpuv::startServer() for more details on what can be returned as the re-
         sponse. ex. list("/" = c("GET" = expression(get_function(req)), "POST" = ex-
                     pression(post_function(req))))
serveHTTP is a convenient way to start a HTTP server that works for both static and dynamically created pages. It offers a simplified and organized interface to httpuv::startServer()/httpuv::runServer() that makes serving static and dynamic pages easier. For dynamic pages, the expression evaluated when a browser requests a dynamically served path should likely be an expression wrapping a function that has "req" as a parameter. Per the Rook specification implemented by httpuv, "req" is the R environment in which browser request information is collected. Therefore, to access HTTP request headers, inputs, etc. in a function served by a dynamic path, "req" should be a parameter of that function. For the dynamic parameter of serveHTTP, list("/" = c("GET" = expression(get_homepage(req)))) would be a suitable way to call the function get_homepage(req) when the root path of a website is requested with the GET method. The req environment has the following variables: request_method = req$REQUEST_METHOD, script_name = req$SCRIPT_NAME, path_info = req$PATH_INFO, query_string = req$QUERY_STRING, server_name = req$SERVER_NAME, server_port = req$SERVER_PORT, headers = req$HEADERS, rook_input = req["rook.input"]$read_lines(), rook_version = req["rook.version"]$read_lines(), rook_url_scheme = req["rook.url_scheme"]$read_lines(), rook_error_stream = req["rook.errors"]$read_lines()

Value

A HTTP web server on the specified host and port.

Examples

```r
# Run both functions and go to http://127.0.0.1:5001/ in a web browser
get_example <- function(req){
  html <- html_doc(
    head(),
    body(
      h1("Hello"),
      p("Here is a list of some of the variables included in the req environment that were associated with this request:"),
      ul(
        li(paste0("req$REQUEST_METHOD = ", req$REQUEST_METHOD)),
        li(paste0("req$SCRIPT_NAME = ", req$SCRIPT_NAME)),
        li(paste0("req$PATH_INFO = ", req$PATH_INFO)),
        li(paste0("req$QUERY_STRING = ", req$QUERY_STRING)),
        li(paste0("req$SERVER_NAME = ", req$SERVER_NAME)),
        li(paste0("req$SERVER_PORT = ", req$SERVER_PORT))
      ),
      p("You can use paramList() to deal with inputs passed through query strings as well as passed through the input stream.
      
      params <- paramList(req[["rook.input"]])$read_lines() will give you a named list of parameters.
    )
  )
  return(
    list(
      status = 200L,
      headers = list('Content-Type' = 'text/html'),
      body = html
    )
  )
```

smart_options

)  
)  
)

serveHTTP(
host = "127.0.0.1",
port = 5001,
persistent = FALSE,
static = list(),
dynamic = list(
"/" = c(
"GET" = expression(get_example(req))
)
)
)

---

smart_options  Creates HTML option tags for each position of a list of values and labels by calling html5::option(), returning a string of HTML to pass to a select tag through html5::select().

Description

Creates HTML option tags for each position of a list of values and labels by calling html5::option(), returning a string of HTML to pass to a select tag through html5::select().

Usage

smart_options(x, value, label, selected_value, add_blank = FALSE)

Arguments

x  A named list, one name should refer to a vector of values, one name should refer to a vector of labels equal in length to the values.
value  The name of the position in x to use as the value attribute for each option tag.
label  The name of the position in x to use as the displayed content for each option tag.
selected_value  A value in the vector passed as value to mark as the initially selected option in the select tag.
add_blank  TRUE/FALSE, if TRUE, adds a blank ("") option tag.

Value

A string, with an option tag each row of x.
Examples

```r
smart_options(
    x = list(col1 = c("1", "2", "3"), col2 = c("New York", "Los Angeles", "Chicago")),
    value = "col1",
    label = "col2",
    selected_value = "3",
    add_blank = TRUE
)
```

<table>
<thead>
<tr>
<th>toInput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace NA values with &quot;, useful for passing values to HTML tags.</td>
</tr>
</tbody>
</table>

Description

Replace NA values with ", useful for passing values to HTML tags.

Usage

```r
toInput(x)
```

Arguments

- **x** A vector of length 1.

Value

A string, if x is NA, returns "."

Examples

```r
toInput(NA)
```
Index

castDateString, 2
castNumeric, 2
doubleQuoteText, 3
delServer, 3
fromInput, 4
jsonStr, 5
paramList, 5
quoteText, 6
sampleStr, 6
serveHTTP, 7
smart_options, 9
toInput, 10