Package 'helper'

March 24, 2016

Title map-x helper functions.

Version 0.0.1

Date 2015-10-05

Description map-x helper functions

License GPL-3 | file LICENSE

URL https://github.com/fxi/map-x-shiny

BugReports https://github.com/fxi/map-x-shiny/issues

Imports leaflet,
RPostgreSQL

RoxygenNote 5.0.1

R topics documented:

`addPaletteFun` *Add palette to map-x style list*

Description

Update a style list with a palette, using the defined scale type : continuous or discrete.

Usage

```
addPaletteFun(sty, pal)
```

Arguments

`sty`	map-x style
`pal`	name of palette to use

`addVectorTiles` *Add vector tiles for a given PGRestAPI postgres endpoint.*

Description

Add vector tiles for a given PGRestAPI postgres endpoint.

Usage

```
addVectorTiles(map, protocol = "http", url = "localhost", port = 3030,
  table = NULL, dataColumns = NULL, geomColumn = "geom",
  idColumn = "gid", id = NULL, group = "default", debug = FALSE,
  zIndex = 100, onLoadFeedback = c("once", "never", "always"))
```

Arguments

`map`	Leaflet map object
`urlTemplate`	Url template for a given PGRestAPI endpoint.

dbAddGeoJSON *Add geojson list or file to db postgis*

Description

Add geojson list or file to db postgis

Usage

```
dbAddGeoJSON(geojsonList = NULL, geojsonPath = NULL, dbInfo = NULL,
  tableName = NULL, archiveIfExists = T, archivePrefix = "mx_archives")
```

Arguments

geojsonList	list containing the geojson data
geojsonPath	path the geojson
dbInfo	dbInfo object containgin pass,user, ...
tableName	Name of the postgis layer / table

dbGetColumnInfo *Get variable summary*

Description

Get variable summary

Usage

```
dbGetColumnInfo(dbInfo = NULL, table = NULL, column = NULL)
```

Arguments

dbInfo	Named list with dbName,host,port, user and password
table	Table/layer from which extract extent
column	Column/Variable on wich extract summary

dbGetFilterCenter *Get query extent, based on a pattern matching (character)*

Description

Search for a value in a column (character data type) and return the extent if something is found.

Usage

```
dbGetFilterCenter(dbInfo = NULL, table = NULL, column = NULL,
  value = NULL, geomColumn = "geom", operator = "=")
```

Arguments

dbInfo	Named list with dbName,host,port,user and password
table	Table/layer from which extract extent
geomColumn	set geometry column

Value

extent

dbGetGeoJSON *Geojson from postGIS base*

Description

Geojson from postGIS base

Usage

```
dbGetGeoJSON(dbInfo, query, fromSrid = "4326", toSrid = "4326",
  asList = FALSE)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password
query	PostGIS spatial sql querry.

Value

geojson list

dbGetLayerCentroid *Get layer center*

Description

Compute the union of all geometry in a given layer and return the coordinate of the centroid.

Usage

```
dbGetLayerCentroid(dbInfo = NULL, table = NULL, geomColumn = "geom")
```

Arguments

dbInfo	Named list with dbName,host,port,user and password
table	Table/layer from which extract extent
geomColumn	set geometry column

Value

extent

dbGetLayerExtent *Get layer extent*

Description

Get layer extent

Usage

```
dbGetLayerExtent(dbInfo = NULL, table = NULL, geomColumn = "geom")
```

Arguments

dbInfo	Named list with dbName,host,port,user and password
table	Table/layer from which extract extent
geomColumn	set geometry column

Value

extent

dbGetSp *Transfert postgis feature by sql query to sp object*

Description

Transfert postgis feature by sql query to sp object

Usage

```
dbGetSp(dbInfo, query)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password.
query	PostGIS spatial sql querry.

Value

spatial object.

dbWriteSpatial *Write spatial data frame to postgis*

Description

Convert spatial data.frame to postgis table. Taken from https://philipphunziker.wordpress.com/2014/07/20/transferring-vector-data-between-postgis-and-r/

Usage

```
dbWriteSpatial(con, spatial.df, schemaname = "public", tablename,
  overwrite = FALSE, keyCol = "gid", srid = 4326, geomCol = "geom")
```

Arguments

con	PostgreSQL connection
spatial.df	Spatial data frame object
schemaname	Target schema table
tablename	Target table name
overwrite	Overwrite if exists
keyCol	Set new primary key
srid	Set the epsg code / SRID
geomCol	Set the name of the geometry column

glAddLayer *gl add layer*

Description

gl add layer

Usage

```
glAddLayer(map, idGl, idBelowTo = NULL, style)
```

glAddSource *gl add source*

Description

gl add source

Usage

```
glAddSource(map, idGl, idSource, style)
```

glInit *gl layer new*

Description

gl layer new

Usage

```
glInit(map, idGl, style, token)
```

glMakeUrl *Create url for pgrestapi source*

Description

Create url for pgrestapi source

Usage

```
glMakeUrl(protocol = "http", host = "localhost", port, table,
  fieldVariables, fieldGeom)
```

Value

url

glRemoveLayer *gl remove layer*

Description

gl remove layer

Usage

```
glRemoveLayer(map, idGl, idLayer)
```

glRemoveSource *gl remove source*

Description

gl remove source

Usage

```
glRemoveSource(map, idGl, idSource)
```

glSetFilter *gl set filter for a layer*

Description

gl set filter for a layer

Usage

```
glSetFilter(map, idGl, idLayer, filter)
```

glSetPaintProperty *gl set paint property for a layer*

Description

gl set paint property for a layer

Usage

```
glSetPaintProperty(map, idGl, idLayer, name, value)
```

hot.to.df *hot.to.df*

Description

Converts the table data passed from the client-side into a data.frame

Usage

```
hot.to.df(b)
```

Arguments

b	The input$hotable_id value.

hotable *hotable*

Description

Creates a hotable (handsontable)

Usage

```
hotable(id, width = "100%", height = "100%")
```

Arguments

id	The id used to refer to the table input$id or output$id

leafletDrawDependencies *Add leaflet draw tools*

Description

Add leaflet draw tools

Usage

```
leafletDrawDependencies()
```

listToHtml *R list to html*

Description

R list to html

Usage

```
listToHtml(listInput, htL = "", h = 2, exclude = NULL)
```

Arguments

listInput	list in inptu
htL	List to append to
h	Value of the first level of html header
exclude	list named item to exclude

listToHtmlClass *R list to html list*

Description

Create a html list and apply a class for <ul> and

Usage

```
listToHtmlClass(listInput, exclude = NULL, c = 0, htL = "",
  classUl = "list-group", classLi = "list-group-item")
```

Arguments

listInput	list in inptu
exclude	list named item to exclude
htL	List to append to
h	Value of the first level of html header

Value

HTML list

loadUi *Load external ui file value in shiny app*

Description

Shortcut to load external shiny ui file

Usage

```
loadUi(path)
```

Arguments

path	Path to the file

mapxhelper *Map-x helper functions*

Description

Map-x core functions

mxAccordionGroup *Create a bootstrap accordion*

Description

Create a bootstrap accordion element, based on a named list.

Usage

```
mxAccordionGroup(id, style = NULL, show = NULL, itemList)
```

Arguments

id	Accordion group ID
style	Additional style.
show	Vector of item number. Collapse all item except those in this list. E.g. c(1,5) will open items 1 and 5 by default.
itemList	Nested named list of items, containing title and content items. E.g. list("foo"=list("title"="foo","conte

Examples

```
amAccordionGroup(id='superTest',
 itemList=list(
   'a'=list('title'='superTitle',content='acontent'),
   'b'=list('title'='bTitle',content='bContent'))
 )
```

mxActionButtonState *Toggle disabling of given button, based on its id.*

Description

Action or other button can be disabled using the attribute "disabled". This function can update a button state using this method.

Usage

```
mxActionButtonState(id, disable = FALSE, warning = FALSE,
  session = shiny:::getDefaultReactiveDomain())
```

Arguments

id	Id of the button.
disable	State of the button
session	Shiny session object.

mxAllow *Control ui access*

Description

Use mxConfig$roleVal list to check if the curent user's role name can access to the given numeric role.

Usage

```
mxAllow(logged, roleName, roleLowerLimit)
```

Arguments

logged	Boolean. Is the user logged in ?
roleName	Character. Role in numeric format
roleLowerLimit	Numeric. Minumum role requirement

mxAnalysisOverlaps *Overlaps analysis*

Description

Use a mask to get overlaps over a layer

Usage

```
mxAnalysisOverlaps(dbInfo, inputBaseLayer, inputMaskLayer, outName,
  dataOwner = "mapxw", sridOut = 4326, varToKeep = "gid")
```

mxCanReach *Test for internet connection. The idea is to reach google with a ping and determine if there is a full packet response without loss*

Description

Test for internet connection. The idea is to reach google with a ping and determine if there is a full packet response without loss

Usage

```
mxCanReach(server = "google.com", port = 80)
```

Arguments

host	String. Host name to ping

mxCatch *Catch errors*

Description

Catch errors and return alert panel in an existing div id.

Usage

```
mxCatch(title, expression, session = shiny:::getDefaultReactiveDomain(),
  debug = TRUE, logToJs = TRUE, panelId = "panelAlert", ...)
```

Arguments

title	Title of the alert
session	Shiny session object
debug	Boolean. Return also message as alert.
panelId	Id of the output element

mxCheckboxIcon *Set a checkbox button with custom icon.*

Description

Create a checkbox input with a select icon.

Usage

```
mxCheckboxIcon(id, idLabel, icon, display = TRUE)
```

Arguments

id	Id of the element
icon	Name of the fontawesome icon. E.g. cog, times, wrench

mxCreatePaletteList *Create a formated list of available palettes*

Description

Create a formated list of available palettes

Usage

```
mxCreatePaletteList(palettes)
```

mxCreateSecret *Create random secret*

Description

Get a random string of letters and hash it.

Usage

```
mxCreateSecret(n = 20)
```

Arguments

n	Number of input letter for the MD5 hash

mxDbAddData *Add data to db*

Description

Add data to db

Usage

```
mxDbAddData(dbInfo, data, table)
```

mxDbClearAll *Remove old results from db query*

Description

Remove old results from db query

Usage

```
mxDbClearAll(dbInfo)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password

mxDbExistsTable *Check if table exists in postgresql*

Description

Shortcut to create a connection, and check if table exists.

Usage

```
mxDbExistsTable(dbInfo, table)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password
table	Name of the table to check

mxDbGetQuery *Get query result from postgresql*

Description

Shortcut to create a connection, get the result of a query and close the connection, using a dbInfo list.

Usage

```
mxDbGetQuery(dbInfo, query, stringAsFactors = F)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password
SQL	query

mxDbListColumns *List existing column from postgresql table*

Description

Shortcut to create a connection, get the list of column and close the connection, using a dbInfo list.

Usage

```
mxDbListColumns(dbInfo, table)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password

mxDbListTable *List existing table from postgresql*

Description

Shortcut to create a connection, get the list of table and close the connection, using a dbInfo list.

Usage

```
mxDbListTable(dbInfo)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password

mxDebugMsg *Print debug message*

Description

Print a defaut debug message with date as prefix. NOTE: this function should take a global parameter "debug" and a log file.

Usage

```
mxDebugMsg(text = "")
```

Arguments

m	Message to be printed

mxDecode *decode base64 string*

Description

decode base64 string

Usage

```
mxDecode(base64text)
```

Arguments

base64text	base64string encoded

mxEitiGetCountryCenter *Create a formated list of country center from eiti countries table*

Description

Create a formated list of country center from eiti countries table

Usage

```
mxEitiGetCountryCenter(eitiCountryTable)
```

mxEitiGetCountrySelectizeList *Create a formated list for selectize input from eiti countries table*

Description

Create a formated list for selectize input from eiti countries table

Usage

```
mxEitiGetCountrySelectizeList(eitiCountryTable)
```

mxEncode *encode in base64*

Description

encode in base64

Usage

```
mxEncode(text)
```

Arguments

text	character string to encode

mxFileInput *Custom file input*

Description

Default shiny fileInput has no option for customisation. This function allows to fully customize file input using the label tag.

Usage

```
mxFileInput(inputId, label, fileAccept = NULL, multiple = FALSE)
```

Arguments

inputId	id of the file input
label	Label for the input
fileAccept	List of accepted file type. Could be extension.
multiple	Boolean. Allow multiple file to be choosen. Doesn't work on all client.

mxGetCookies *Get cookie from session HTTP request*

Description

Get cookie from session HTTP request

Usage

```
mxGetCookies(session = getDefaultReactiveDomain())
```

mxGetLayerMeta *Get layer meta stored in default layer table*

Description

Get layer meta stored in default layer table

Usage

```
mxGetLayerMeta(dbInfo, layer)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password
layer	Postgis layer stored in layer table. Should have a meta field.

mxGetViewData *Get view data as list*

Description

Get view data as list

Usage

```
mxGetViewData(dbInfo, viewId, select = NULL)
```

Arguments

dbInfo	Named list with dbName,host,port,user and password
viewId	Vector of view id(s) for which to retrieve data
select	Vector of columns to retrieve

mxGetViewsTable *Retrieve map views table*

Description

Get a list of available map-x views in given table, e.g. mx_views

Usage

```
mxGetViewsTable(dbInfo = NULL, table = "mx_views", validated = TRUE,
  archived = FALSE, country = "AFG")
```

Arguments

dbInfo	Named list with dbName,host,port, user and password
table	Table name containing views info
validated	Boolean filter validated dataset. Default = TRUE
archived	Boolean filter to get archived data. Default =FALSE
country	ISO 3 code to filter country.

mxGetWdiIndicators *Create WDI indicators list*

Description

Create WDI indicators list

Usage

```
mxGetWdiIndicators()
```

`mxMakeViewList` *extract views from the db and create a list*

Description

extract views from the db and create a list

Usage

```
mxMakeViewList(dbInfo, cntry)
```

Arguments

`dbInfo`	map-x db info list
`cntry`	Country iso3 code

Value

list of views data and style

`mxMakeViews` *Create html list of available views*

Description

get a list of views and return a HTML shiny checkbox input.

Usage

```
mxMakeViews(views)
```

Arguments

`views`	List of available views

mxPanel *Create a modal panel*

Description

Create a modal panel with some options as custom button, close button, html content.

Usage

```
mxPanel(id = "default", title = NULL, subtitle = NULL, html = NULL,
  listActionButton = NULL, background = TRUE, addCancelButton = FALSE,
  addOnClickClose = TRUE, defaultButtonText = "OK", style = NULL,
  class = NULL, hideCloseButton = FALSE, draggable = TRUE, fixed = TRUE)
```

Arguments

id	Panel id
title	Panel title
subtitle	Panel subtitle
html	HTML content of the panel, main text
listActionButton	If FALSE, hide buttons. If NULL, display default close panel button, with text given in defaultButtonText. If list of buttons, list of button.
defaultButtonText	Text of the default button if listActionButton is NULL and not FALSE
style	Additional CSS style for the panel
class	Additional class for the panel
hideCloseButton	Boolean. Hide the close panel button
draggable	Boolean. Set the panel as draggable

mxPanelAlert *Alert panel*

Description

Create an alert panel. This panel could be send to an output object from a reactive context.

Usage

```
mxPanelAlert(title = c("error", "warning", "message"), subtitle = NULL,
  message = NULL, listActionButton = NULL, ...)
```

Arguments

title	Title of the alert. Should be "error", "warning" or "message"
subtitle	Subtitle of the alert
message	html or text message for the alert
listActionButtons	List of action button for the panel

---

mxParseListFromText *Parse key value pair from text*

Description

Parse key value pair from text

Usage

```
mxParseListFromText(txt)
```

Arguments

txt	unformated text with key value pair. eg. myKey = myValue

Value

list of value

---

mxParseStory *Parse story map : markdown, R, view and video*

Description

Parse story map : markdown, R, view and video

Usage

```
mxParseStory(txtorig, knit = T, toc = F)
```

Arguments

test	Story map text

Value

parsed html

mxParseView *Parse view string*

Description

Parse view string

Usage

```
mxParseView(text)
```

Arguments

test	Story map text with @view_start(name ; id ; extent) ... @view_end tags

Value

parsed html

mxParseVimeo *Parse vimeo string*

Description

Parse vimeo string

Usage

```
mxParseVimeo(text)
```

Arguments

text	Story map text with @vimeo(id ; desc) tag

Value

html enabled version

mxRemoveEl *remove element by class or id*

Description

remove element by class or id

Usage

```
mxRemoveEl(session = getDefaultReactiveDomain(), class = NULL, id = NULL)
```

Arguments

session	default shiny session
class	class name to remove
id	id to remove

mxSelectInput *Custom select input*

Description

Custom select input without label.

Usage

```
mxSelectInput(inputId, choices = NULL, selected = NULL)
```

Arguments

inputId	Element id
choices	List of options
select	Value selected by default

mxSendJson *function to read json and save as an object*

Description

function to read json and save as an object

Usage

```
mxSendJson(pathToJson, objName, session = getDefaultReactiveDomain())
```

mxSetCookie *Save named list of value into cookie*

Description

Note : don't use this for storing sensitive data, unless you have a trusted network.

Usage

```
mxSetCookie(session = getDefaultReactiveDomain(), cookie = NULL,
  nDaysExpires = NULL, deleteAll = FALSE, read = TRUE)
```

Arguments

session	Shiny session object. By default: default reactive domain.
cookie	Named list holding paired cookie value. e.g. (list(whoAteTheCat="Alf"))
nDaysExpires	Integer of days for the cookie expiration
read	Boolean. Read written cookie

mxSetStyle *Apply map-x style to existing vector tiles*

Description

When leafletvt handle a vector tile source, a lealflet object is stored in leafletvtId, but no style is applied. Default is transparent. We add a style function after that the layer is fully loaded using this function. The style function is also stored alongside the leaflet object in leafletId under the name "vtStyle".

Usage

```
mxSetStyle(session = shiny:::getDefaultReactiveDomain(), style,
  mapId = "mapxMap")
```

Arguments

session	Shiny session object
style	map-x style

mxSliderOpacity *Set ioRange slider for opacity*

Description

Return a div than contain a slider input instantiated with ionRangeSlider for view opacity

Usage

```
mxSliderOpacity(id, opacity)
```

Arguments

id	Id of the slider
opacity	Default opacity

mxStyleReset *Reset all value in a reactiveValues object*

Description

Reset all value in a reactiveValues object

Usage

```
mxStyleReset(reactiveObj)
```

Arguments

reaciveObj	Reactive values object

mxTextValidation *Sting validation*

Description

Check if a string exists in a vector of string, if there is a duplicate, if contain at least n character, etc.. and update an existing div with a html summary. Return if the string is valid or not.

Usage

```
mxTextValidation(textToTest, existingTexts, idTextValidation, minChar = 5,
  testForDuplicate = TRUE, testForMinChar = TRUE,
  displayNameInValidation = TRUE, existsText = "taken",
  errorColor = "#FF0000")
```

Arguments

existingTexts	Vector of existing text
idTextValidation	Id of the ui element to update (id=example -> uiOutput("example"))
minChar	Minimum character length
testForDuplicate	Boolean test for duplicate.
testForMinChar	Boolean test for minimum number of character
displayNameInValidation	Boolean add text in validation text
textTotest	text to test against rules

Value

boolean : valid or not

---

mxTimeSlider *Set ioRange slider for time slider*

Description

Return a div than contain a slider input instantiated with ionRangeSlider for view time slider.

Usage

```
mxTimeSlider(id, min, max, lay)
```

Arguments

id	Id of the slider
min	Minimum js unix date in milisecond
max	Maxmimum js unix date in milisecond
lay	Layer name

---

mxTimeSliderRange *Set ioRange slider for time slider*

Description

Return a div than contain a slider input instantiated with ionRangeSlider for view time slider range.

Usage

```
mxTimeSliderRange(id, min, max, lay)
```

Arguments

id	Id of the slider
min	Minimum js unix date in milisecond
max	Maxmimum js unix date in milisecond
lay	Layer name

---

mxUiAccess *Control ui access UI manager based on login info*

Description

Control ui access

UI manager based on login info

Usage

```
mxUiAccess(logged, roleNum, roleLowerLimit, uiDefault, uiRestricted)
```

Arguments

logged	Boolean. Is the user logged in ?
roleNum	Numeric. Role in numeric format
roleLowerLimit	Numeric. Minumum role requirement
uiDefault	TagList. Default ui.
uiRestricted	TagList. Restricted ui.

---

mxUiEnable *Control visbility of elements*

Description

Display or hide element by id, without removing element AND without having element's space empty in UI. This function add or remove mx-hide class to the element.

Usage

```
mxUiEnable(session = shiny:::getDefaultReactiveDomain(), id = NULL,
  class = NULL, enable = TRUE, classToRemove = "mx-hide")
```

Arguments

session	Shiny session
id	Id of element to enable/disable
enable	Boolean. Enable or not.

mxUpdateChartRadar *Create a chartRadar in a canvas element.*

Description

Search the dom for an id a get drawing context, create a new chart object and config it with data.

Usage

```
mxUpdateChartRadar(session = shiny::getDefaultReactiveDomain(), main,
  compMain, id, idLegend, labels, values, compValues)
```

Arguments

session	Shiny reactive session
main	Main label
compMain	Comparative value label
id	Id of the canvas
idLegend	Id of the legend
labels	Labels for value and comparative values
compValues	Comparative values
value	Values

mxUpdatePanel *Update existing panel*

Description

Use output object to update the panel with a known id. E.g. for updating uiOutput("panelTest"), use mxUpdatePanel with panelId "panelTest"

Usage

```
mxUpdatePanel(panelId = NULL, session = shiny:::getDefaultReactiveDomain(),
  ...)
```

Arguments

panelId	Id of the existing panel
session	Shiny reactive object of the session
...	Other mxPanel options

mxUpdateText *Update text by id*

Description

Search for given id and update content.

Usage

```
mxUpdateText(id, text = NULL, ui = NULL, addId = FALSE,
  session = shiny:::getDefaultReactiveDomain())
```

Arguments

id	Id of the element
text	New text
session	Shiny session

mxUpdateValue *Update value by id*

Description

Search for given id and update value.

Usage

```
mxUpdateValue(id, value, session = shiny:::getDefaultReactiveDomain())
```

Arguments

id	Id of the element
value	New text value
session	Shiny session

`noDataCheck` *Check for no null, NA's, nchar of 0, lenght of 0 or "[NO DATA]" string in a vector.*

Description

Check for no null, NA's, nchar of 0, lenght of 0 or "[NO DATA]" string in a vector.

Usage

```
noDataCheck(val, useNoData = TRUE, noDataVal = "[ NO DATA ]")
```

Arguments

`val`	Vector to test for no data.

Value

TRUE if no data (nchar == 0 OR is.na OR is.null) found or if input is not a vector

`pwdInput` *Password input*

Description

Create a password input.

Usage

```
pwdInput(inputId, label)
```

Arguments

`inputId`	Input id
`label`	Label to display

randomName *Random name generator*

Description

Create a random name with optional prefix and suffix.

Usage

```
randomName(prefix = NULL, suffix = NULL, n = 20, sep = "_")
```

Arguments

prefix	Prefix. Default = NULL
suffix	Suffix. Default = NULL
n	Number of character to include in the random string

Value

Random string of letters, with prefix and suffix

remoteCmd *Send command on remote server through ssh*

Description

Allow sending command on a remote server, e.g. Vagrant machine, using ssh.

Usage

```
remoteCmd(host = NULL, user = NULL, port = NULL, cmd = NULL,
  vagrant = TRUE, sshConfig = "settings/sshConfig")
```

Arguments

host	Host
user	User
port	Port
cmd	Command to send
vagrant	Boolean. If TRUE, use ssh config file. E.g. vagrant ssh-config > sshConfig

renderHotable *renderHotable*

Description

Renders the hotable.

Usage

```
renderHotable(expr, env = parent.frame(), quoted = FALSE, options = NULL,
  readOnly = NULL, fixedCols = 1, stretched = c("all", "last", "none"))
```

Arguments

expr	The computation that leads to an output
env	The R environment in which to create the dataset
quoted	Pass to the exprToFunction
options	Pass to the exprToFunction
readOnly	A vector of TRUE/FALSE values to indicate which of the columns should be readonly. If numeric vector, select col number to set as readOnly.
fixedCols	A vector of integer of columns number to fix.

setVectorTilesVisibility
Remove vector tiles.

Description

Remove vector tiles.

Usage

```
setVectorTilesVisibility(map, group = "default", visible = TRUE)
```

Arguments

map	Leaflet map object
group	Group/id of the vector tiles layer

setZoomOptions *Set zoom button options*

Description

Set zoom button options

Usage

```
setZoomOptions(map, buttonOptions = list(), removeButton = FALSE)
```

Arguments

map	Leaflet map object
removeButton	Boolean. Remove the zoom button.
butonOptions	List of Leaflet options for zoom butons. E.g. list(position="topright")

subPunct *Substitute ponctiation and non-ascii character*

Description

Take a string and convert to ascii string with optional transliteration ponctuation convertion.

Usage

```
subPunct(str, sep = "_", rmTrailingSep = T, rmLeadingSep = T,
  rmDuplicateSep = T, useTransliteration = T)
```

Arguments

str	String to evaluate
sep	Replace separator
rmTrailingSep	Logical argument : no trailing separator returned
rmLeadingSep	Logical argument : no leading separator returned
rmDuplicateSep	Logical argument : no consecutive separator returned

usrInput *User name input*

Description

Create a username input

Usage

```
usrInput(inputId, label)
```

Arguments

inputId	Input id
label	Label to display

vtDataList *Get layer/table and available field/column combined in a list*

Description

Get layer/table and available field/column combined in a list

Usage

```
vtDataList(protocol = "http", url = "localhost", port = 3030)
```

Arguments

protocol	E.g. http
url	Server url (without http://), default = "localhost"
port	Server port number. default = 3000

vtGetColumns *Get available fields/columns from a layer/table*

Description

Get available fields/columns from a layer/table

Usage

```
vtGetColumns(protocol = "http", url = "localhost", port = 3030,
  table = NULL, exclude = NULL)
```

Arguments

protocol	E.g. http
url	Server url (without http://), default = "localhost"
port	Server port number, default = 3000
table	Table name.

---

vtGetLayers *Get vector tile layer (PostGIS table) from PGRestAPI*

Description

Get vector tile layer (PostGIS table) from PGRestAPI

Usage

```
vtGetLayers(protocol = "http", url = "localhost", port = 3030,
  grepExpr = "", nTry = 5)
```

Arguments

protocol	E.g. http
url	Server url (without http://), default = "localhost".
port	Server port number, default = 3000

Index