Package ‘leaflet.extras2’

October 13, 2022

Type Package
Title Extra Functionality for 'leaflet' Package
Version 1.2.0
Description Several 'leaflet' plugins are integrated, which are available as extension to the 'leaflet' package.
License GPL-3 | file LICENSE
Encoding UTF-8
LazyData true
Depends R (>= 3.1.0), leaflet (>= 2.0.0)
Imports htmltools, magrittr, utils
Suggests jsonlite, shiny, sf, geojsonsf, sp, testthat (>= 2.1.0), fontawesome, htmlwidgets, covr
BugReports https://github.com/trafficonese/leaflet.extras2/issues
RoxygenNote 7.1.1
NeedsCompilation no
Author Gatscha Sebastian [aut, cre], Ricardo Rodrigo Basa [ctb], Jeffrey O Hanson [ctb] (<https://orcid.org/0000-0002-4716-6134>)
Maintainer Gatscha Sebastian <sebastian_gatscha@gmx.at>
Repository CRAN
Date/Publication 2022-05-10 19:40:02 UTC

R topics documented:

addAntpath ................................................. 4
addArrowhead ............................................ 6
addContextmenu ........................................... 8
R topics documented:

addEasyprint ...................................................... 9
addGIBS .......................................................... 10
addHeightgraph ................................................... 11
addHexbin .......................................................... 13
addHistory .......................................................... 15
addItemContextmenu ............................................... 16
addLabelgun ........................................................ 16
addLeafletsync .................................................... 17
addLeafletsyncDependency ....................................... 18
addMapkeyMarkers ............................................... 19
addMovingMarker .................................................. 21
addOpenweatherCurrent .......................................... 22
addOpenweatherTiles ............................................. 24
addPlayback ....................................................... 25
addReachability ................................................... 28
addSidebar .......................................................... 29
addSidebyside ..................................................... 30
addSpinner .......................................................... 31
addTangram ........................................................ 32
addTimeslider ...................................................... 33
addVelocity ........................................................ 35
addWMS ............................................................. 36
antpathOptions ...................................................... 38
arrowheadOptions ................................................ 39
clearAntpath ........................................................ 40
clearArrowhead .................................................... 41
clearFuture .......................................................... 41
clearHexbin .......................................................... 42
clearHistory ........................................................ 43
closeSidebar .......................................................... 43
context_mapmenuItems .......................................... 44
context_markernmenuItems ....................................... 44
context_menuItem .................................................. 45
easyprintMap ....................................................... 46
easyprintOptions ................................................. 47
gibs_layers .......................................................... 48
goBackHistory ..................................................... 49
goForwardHistory ................................................ 49
heightgraphOptions .............................................. 50
hexbinOptions ...................................................... 51
hideContextmenu .................................................. 52
hideHexbin .......................................................... 53
historyOptions ..................................................... 53
insertItemContextmenu .......................................... 55
isSynced ............................................................. 56
leafletextras2 ...................................................... 56
leafletsyncOptions ............................................... 57
makeMapkeyIcon .................................................... 57
R topics documented:

mapkeyIconList ........................................ 59
mapkeyIcons ........................................... 60
mapmenuItems ........................................... 61
markermenuItems ....................................... 62
menuItem .................................................. 62
movingMarkerOptions ................................... 63
openSidebar .............................................. 64
openweatherCurrentOptions ......................... 64
openweatherOptions .................................... 65
playbackOptions ........................................ 66
reachabilityOptions .................................... 67
removeAllItemsContextmenu ......................... 68
removeAntpath .......................................... 69
removeArrowhead ....................................... 69
removeEasyprint ........................................ 70
removeItemContextmenu ................................ 70
removePlayback ......................................... 71
removeReachability .................................... 71
removeSidebar .......................................... 72
removeSidebyside ...................................... 72
removeTimeslider ....................................... 73
removeVelocity ......................................... 73
setDate .................................................... 74
setDisabledContextmenu ............................... 74
setOptionsVelocity ..................................... 75
setTransparent ......................................... 76
showContextmenu ....................................... 76
showHexbin .............................................. 77
sidebar_pane ........................................... 77
sidebar_tabs ............................................. 78
startMoving ............................................. 79
timesliderOptions ....................................... 81
to_jsonformat ........................................... 82
to_ms ...................................................... 83
unsync .................................................... 83
updateHexbin ........................................... 84
velocityOptions ........................................ 84
[.leaflet_mapkey_icon_set .............................. 85

Index ................................................... 87
addAntpath  

Add Antpath Lines

Description

Can be used almost exactly like addPolylines but instead of pathOptions you can use antpathOptions to adapt the Antpath behaviour. See leaflet-ant-path for further details.

Usage

addAntpath(
  map,
  lng = NULL,
  lat = NULL,
  layerId = NULL,
  group = NULL,
  stroke = TRUE,
  color = "#03F",
  weight = 5,
  opacity = 0.5,
  fill = FALSE,
  fillColor = color,
  fillOpacity = 0.2,
  dashArray = NULL,
  smoothFactor = 1,
  noClip = FALSE,
  popup = NULL,
  popupOptions = NULL,
  label = NULL,
  labelOptions = NULL,
  options = antpathOptions(),
  highlightOptions = NULL,
  data = getMapData(map)
)

Arguments

map  
a map widget object created from leaflet()

lng  
a numeric vector of longitudes, or a one-sided formula of the form ~x where x is a variable in data; by default (if not explicitly provided), it will be automatically inferred from data by looking for a column named lng, long, or longitude (case-insensitively)

lat  
a vector of latitudes or a formula (similar to the lng argument; the names lat and latitude are used when guessing the latitude column from data)

layerId  
the layer id
The `addAntpath` function adds animated polylines to a leaflet map. It accepts several options, including:

- **group**: the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g., markers and polygons) can share the same group name.

- **stroke**: whether to draw stroke along the path (e.g., the borders of polygons or circles)
- **color**: stroke color
- **weight**: stroke width in pixels
- **opacity**: stroke opacity (or layer opacity for tile layers)
- **fill**: whether to fill the path with color (e.g., filling on polygons or circles)
- **fillColor**: fill color
- **fillOpacity**: fill opacity
- **dashArray**: a string that defines the stroke dash pattern
- **smoothFactor**: how much to simplify the polyline on each zoom level (more means better performance and less accurate representation)
- **noClip**: whether to disable polyline clipping
- **popup**: a character vector of the HTML content for the popups (you are recommended to escape the text using `htmlEscape()` for security reasons)
- **popupOptions**: a Vector of `popupOptions` to provide popups
- **label**: a character vector of the HTML content for the labels
- **labelOptions**: a Vector of `labelOptions` to provide label options for each label. Default NULL
- **options**: A named list of options. See `antpathOptions`
- **highlightOptions**: Options for highlighting the shape on mouse over.
- **data**: the data object from which the argument values are derived; by default, it is the data object provided to `leaflet()` initially, but can be overridden

**Value**

A modified leaflet map, with an 'ant-path' animated polyline

**References**

https://github.com/rubenspgcavalcante/leaflet-ant-path

**See Also**

Other Antpath Functions: `antpathOptions()`, `clearAntpath()`, `removeAntpath()`

**Examples**

```r
library(leaflet)
leaflet() %>%
  addAntpath(data = atlStorms2005)
```
**Description**

Can be used almost exactly like `addPolylines` but instead of `pathOptions` you can use `arrowheadOptions`. See [leaflet-arrowheads](https://github.com/rstudio/leaflet-arrowheads) for further details.

**Usage**

```r
addArrowhead(
  map,
  lng = NULL,
  lat = NULL,
  layerId = NULL,
  group = NULL,
  stroke = TRUE,
  color = "#03F",
  weight = 5,
  opacity = 0.5,
  fill = FALSE,
  fillColor = color,
  fillOpacity = 0.2,
  dashArray = NULL,
  smoothFactor = 1,
  noClip = FALSE,
  popup = NULL,
  popupOptions = NULL,
  label = NULL,
  labelOptions = NULL,
  options = arrowheadOptions(),
  highlightOptions = NULL,
  data = getMapData(map)
)
```

**Arguments**

- `map` a map widget object created from `leaflet()`
- `lng` a numeric vector of longitudes, or a one-sided formula of the form `~x` where `x` is a variable in `data`; by default (if not explicitly provided), it will be automatically inferred from `data` by looking for a column named `lng`, `long`, or `longitude` (case-insensitively)
- `lat` a vector of latitudes or a formula (similar to the `lng` argument; the names `lat` and `latitude` are used when guessing the latitude column from `data`)
- `layerId` the layer id
The `addArrowhead` function allows adding arrowheads to polylines in a Leaflet map. The function takes several arguments:

- `group`: the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g. markers and polygons) can share the same group name.
  - `stroke`: whether to draw stroke along the path (e.g. the borders of polygons or circles)
  - `color`: stroke color
  - `weight`: stroke width in pixels
  - `opacity`: stroke opacity (or layer opacity for tile layers)
  - `fill`: whether to fill the path with color (e.g. filling on polygons or circles)
  - `fillColor`: fill color
  - `fillOpacity`: fill opacity
  - `dashArray`: a string that defines the stroke dash pattern
  - `smoothFactor`: how much to simplify the polyline on each zoom level (more means better performance and less accurate representation)
  - `noClip`: whether to disable polyline clipping
  - `popup`: a character vector of the HTML content for the popups (you are recommended to escape the text using `htmlEscape()` for security reasons)
  - `popupOptions`: a Vector of `popupOptions` to provide popups
  - `label`: a character vector of the HTML content for the labels
  - `labelOptions`: a Vector of `labelOptions` to provide label options for each label. Default NULL
  - `options`: A named list of options. See `arrowheadOptions`
  - `highlightOptions`: Options for highlighting the shape on mouse over.
  - `data`: the data object from which the argument values are derived; by default, it is the data object provided to `leaflet()` initially, but can be overridden

The `Value` section describes the modified Leaflet map with a polyline with arrowheads.

References

- https://github.com/slutske22/leaflet-arrowheads

See Also

Other Arrowhead Functions: `arrowheadOptions()`, `clearArrowhead()`, `removeArrowhead()`

Examples

```r
library(leaflet)
leaflet() %>%
  addArrowhead(data = atlStorms2005)
```
Description
Add a contextmenu to the map or markers/vector layers.

Usage
addContextmenu(map)

Arguments
map

a map widget object created from leaflet

Details
This function is only used to include the required JavaScript and CSS bindings and to set up some Shiny event handlers.

Contextmenu initialization: The contextmenu for
• the map must be defined in leafletOptions.
• the markers/vector layers must be defined in markerOptions or pathOptions.

Contextmenu selection: When a contextmenu is selected, a Shiny input with the ID "MAPID_contextmenu_select" is set ('MAPID' refers to the map's id). If the selected contextmenu item is triggered from:
• the map, the returned list contains the text of the item.
• the markers, the returned list also contains the layerId, group, lat, lng and label.
• the vector layers, the returned list also contains the layerId, group and label.

Value
A leaflet map object

References
https://github.com/aratcliffe/Leaflet.contextmenu

See Also
Other Contextmenu Functions: addItemContextmenu(), context_mapmenuItems(), context_markermenuItems(), context_menuItem(), hideContextmenu(), insertItemContextmenu(), mapmenuItems(), markermenuItems(), menuItem(), removeItemContextmenu(), removeAllItemsContextmenu(), setDisabledContextmenu(), showContextmenu()
Examples

```r
library(leaflet)
leaflet(options = leafletOptions(
  contextmenu = TRUE,
  contextmenuWidth = 200,
  contextmenuItems =
    context_mapmenuItems(
      context_menuItem("Zoom Out", "function(e) {this.zoomOut()}", disabled=FALSE),
      ":",
      context_menuItem("Zoom In", "function(e) {this.zoomIn()}"))) )
addTiles(group = "base") %>%
addContextmenu() %>%
addMarkers(data = breweries91, label = ~brewery,
    layerId = ~founded, group = "marker",
    options = markerOptions(
      contextmenu = TRUE,
      contextmenuWidth = 200,
      contextmenuItems =
        context_markermenuItems(
          context_menuItem(text = "Show Marker Coords",
            callback = "function(e) {alert(e.latlng);}",
            index = 1)
        )
    )
))
```

---

**addEasyprint**  
*Add easyPrint Plugin*

**Description**

Add a control, which allows to print or export a map as .PNG.

**Usage**

```r
addEasyprint(map, options = easyprintOptions())
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>a map widget object created from <em>leaflet</em></td>
</tr>
<tr>
<td>options</td>
<td>A named list of options. See <em>easyprintOptions</em></td>
</tr>
</tbody>
</table>

**Value**

A leaflet map object

**References**

[https://github.com/rowanwins/leaflet-easyPrint](https://github.com/rowanwins/leaflet-easyPrint)
See Also
Other EasyPrint Functions: easyprintMap(), easyprintOptions(), removeEasyprint()

Examples
library(leaflet)
leaflet() %>%
  addTiles() %>%
  addEasyprint(options = easyprintOptions(
    title = 'Print map',
    position = 'bottomleft',
    exportOnly = TRUE))

addGIBS       Add GIBS Layers

Description
A leaflet plugin for NASA EOSDIS GIBS imagery integration. 154 products are available. The date
can be set dynamically for multi-temporal products. No-data pixels of MODIS Multiband Imagery
can be made transparent.

Usage
addGIBS(
  map,
  layers = NULL,
  group = NULL,
  dates = NULL,
  opacity = 0.5,
  transparent = TRUE
)

Arguments
map         a map widget object created from leaflet()
layers      A character vector of GIBS-layers. See gibs_layers
group       the name of the group the newly created layers should belong to (for clearGroup
            and addLayersControl purposes). Human-friendly group names are permitted--
            they need not be short, identifier-style names. Any number of layers and even
            different types of layers (e.g. markers and polygons) can share the same group
            name.
dates       Date object. If multiple layers are added, you can add a Date vector of the
            same length
opacity      Numeric value determining the opacity. If multiple layers are added, you can
            add a numeric vector of the same length
transparent  Should the layer be transparent. If multiple layers are added, you can add a
            boolean vector of the same length
Value

the new map object

References

https://github.com/aparshin/leaflet-GIBS

See Also

Other GIBS Functions: setDate(), setTransparent()

Examples

library(leaflet)
library(leaflet.extras2)

layers <- gibs_layers$title[c(35, 128, 185)]

leaflet() %>%
  addTiles() %>%
  setView(9, 50, 4) %>%
  addGIBS(layers = layers,
    dates = Sys.Date() - 1,
    group = layers) %>%
  addLayersControl(overlayGroups = layers)

addHeightgraph

Add a Heightgraph layer

Description

Visualize height information and road attributes of linestring segments. The linestrings must be a Simple Feature LINESTRING Z and are transformed to GeoJSON. The function therefore inherits arguments from addGeoJSON.

Usage

addHeightgraph(
  map,
  data = NULL,
  columns = NULL,
  layerId = NULL,
  group = NULL,
  color = "#03F",
  weight = 5,
  opacity = 0.5,
  dashArray = NULL,
  smoothFactor = 1,
addHeightgraph

```r
noClip = FALSE,
pathOpts = leaflet::pathOptions(),
options = heightgraphOptions()
)
```

### Arguments

- **map**
  - A map widget object created from `leaflet()`
- **data**
  - A Simple Feature LINESTRING with Z dimension.
- **columns**
  - A character vector of the columns you want to include in the heightgraph control
- **layerId**
  - the layer id
- **group**
  - the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g. markers and polygons) can share the same group name.
- **color**
  - stroke color
- **weight**
  - stroke width in pixels
- **opacity**
  - stroke opacity (or layer opacity for tile layers)
- **dashArray**
  - a string that defines the stroke **dash pattern**
- **smoothFactor**
  - how much to simplify the polyline on each zoom level (more means better performance and less accurate representation)
- **noClip**
  - whether to disable polyline clipping
- **pathOpts**
  - List of further options for the path. See `pathOptions`
- **options**
  - List of further plugin options. See `heightgraphOptions`

### Value

the new map object

### Note

When used in Shiny, 3 events update a certain Shiny Input:

1. A click updates `input$MAPID_heightgraph_click`
2. A mouseover updates `input$MAPID_heightgraph_mouseover`
3. A mouseout updates `input$MAPID_heightgraph_mouseout`

If you want to explicitly remove the Heightgraph control, please use `removeControl` with the `layerId = "hg_control"`.

### References

See Also

Other Heightgraph Functions: `heightgraphOptions()`

Examples

```r
## Not run:
library(leaflet)
library(leaflet.extras2)
library(sf)

data <- st_cast(st_as_sf(leaflet::atlStorms2005[4,]), "LINESTRING")
data <- st_transform(data, 4326)
data <- data.frame(st_coordinates(data))
data$elev <- runif(nrow(data), 10, 500)
data$L1 <- NULL
L1 <- round(seq.int(1, 4, length.out = nrow(data)))
data <- st_as_sf(st_sfc(lapply(split(data, L1), sfg_linestring)))
data$steepness <- 1:nrow(data)
data$suitability <- nrow(data):1
data$popup <- apply(data, 1, function(x) {
  sprintf("Steepness: %s<br>Suitability: %s", x$steepness, x$suitability)
})

leaflet() %>%
  addTiles(group = "base") %>%
  addHeightgraph(color = "red", columns = c("steepness", "suitability"),
                 opacity = 1, data = data, group = "heightgraph",
                 options = heightgraphOptions(width = 400))

## End(Not run)
```

---

**addHexbin**  
*Add a Hexbin layer*

**Description**

Create dynamic hexbin-based heatmaps on Leaflet maps. This plugin leverages the data-binding power of d3 to allow you to dynamically update the data and visualize the transitions.

**Usage**

```r
addHexbin(
  map,
  lng = NULL,
  lat = NULL,
  radius = 20,
  ...)```
```r
layerId = NULL,
group = NULL,
opacity = 0.5,
options = hexbinOptions(),
data = getMapData(map)
)
```

**Arguments**

- `map` a map widget object created from `leaflet()`
- `lng` a numeric vector of longitudes, or a one-sided formula of the form `~x` where `x` is a variable in `data`; by default (if not explicitly provided), it will be automatically inferred from `data` by looking for a column named `lng`, `long`, or `longitude` (case-insensitively)
- `lat` a vector of latitudes or a formula (similar to the `lng` argument; the names `lat` and `latitude` are used when guessing the latitude column from `data`)
- `radius` Radius of the hexbin layer
- `layerId` the layer id
- `group` the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g. markers and polygons) can share the same group name.
- `opacity` Opacity of the hexbin layer
- `options` List of further options. See `hexbinOptions`.
- `data` the data object from which the argument values are derived; by default, it is the data object provided to `leaflet()` initially, but can be overridden

**Value**

the new `map` object

**Note**

Currently doesn’t respect `layerId` nor `group`.

**References**

[https://github.com/Asymmetrik/leaflet-d3#hexbins-api](https://github.com/Asymmetrik/leaflet-d3#hexbins-api)

**See Also**

Other Hexbin-D3 Functions: `clearHexbin()`, `hexbinOptions()`, `hideHexbin()`, `showHexbin()`, `updateHexbin()`
addHistory

Examples

library(leaflet)
library(leaflet.extras2)

n <- 1000
df <- data.frame(lat = rnorm(n, 42.0285, .01),
                 lng = rnorm(n, -93.65, .01))

leaflet() %>%
  addTiles() %>%
  addHexbin(lng = df$lng, lat = df$lat,
            options = hexbinOptions(
              colorRange = c("red", "yellow", "blue"),
              radiusRange = c(10, 20)
            ))

addHistory  Add History Plugin

Description

The plugin enables tracking of map movements in a history similar to a web browser. By default, it is a simple pair of buttons – back and forward.

Usage

addHistory(map, layerId = NULL, options = historyOptions())

Arguments

map  
a map widget object created from leaflet
layerId  
the control id
options  
A named list of options. See historyOptions

Value

the new map object

References

https://github.com/cscott530/leaflet-history

See Also

Other History Functions: clearFuture(), clearHistory(), goBackHistory(), goForwardHistory(), historyOptions()
Examples

```r
library(leaflet)
leaflet() %>%
  addTiles() %>%
  addHistory()
```

Description

Add a new contextmenu menu item

Usage

```r
addItemContextMenu(map, option)
```

Arguments

- `map` a map widget object created from `leaflet`
- `option` new menu item to add

Value

A leaflet map object

See Also

Other Contextmenu Functions: `addContextmenu()`, `context_mapmenuItems()`, `context_markermenuItems()`, `context_menuItem()`, `hideContextMenu()`, `insertItemContextMenu()`, `mapmenuItems()`, `markermenuItems()`, `menuItem()`, `removeItemContextMenu()`, `removeAllItemsContextMenu()`, `setDisabledContextMenu()`, `showContextMenu()`

---

addLabelgun

### Add addLabelgun Plugin

Description

The plugin allows to avoid cluttering in marker labels and gives priority to labels of your choice (with higher weight).

Usage

```r
addLabelgun(map, group = NULL, weight = NULL, entries = NULL)
```
addLeafletsync

Arguments

map A map widget object created from leaflet

group The group name of the layer/s for which label collisions are to be avoided. To see
the effects of this plugin the labelOptions of the markers must be configured
with either permanent = TRUE or noHide = TRUE.

weight An optional weight for markers. If a vector is given, the length should match
the number of all markers in the corresponding groups. If a numeric value is
specified, it is used for each marker and thus no prioritization of the labels takes
place. In all other cases a random integer is calculated.

extries A numeric value, a higher value relates to faster insertion and slower search, and
vice versa. The default is 10

Value

A leaflet map object

Note

It is important to invoke the function after the markers have been added to the map. Otherwise
nothing will happen.

References

https://github.com/Geovation/labelgun

addLeafletsync Synchronize multiple Leaflet map

Description

The plugin allows you to synchronize and unsynchronize multiple leaflet maps in a Shiny ap-
plication. You can pass additional options to leafletsyncOptions. For more information see
Leaflet.Sync

Usage

addLeafletsync(  
  map,
  ids = NULL,
  synclist = "all",
  options = leafletsyncOptions()
)
addLeafletsyncDependency

Arguments

map the map

ids the map ids to be synced. If you use a synclist, you may leave it NULL. The unique names and values of synclist will be used.

synclist The synchronization list. The default is 'all', which creates a list of all possible combinations of ids. For a more detailed control, a named list can be passed in this form `list(m1 = c("m2", "m3"), m2 = c("m1", "m3"), m3 = c("m1", "m2"))`, where the names and values represent map-ids. The names of the lists serve as a basis and the list values are the maps to be kept in sync with the basemap.

options A named list of options. See leafletsyncOptions. If you want to add different options to multiple maps, you can wrap the options in a named list, with the names being the map-ids. See the example in ./inst/examples/offset_continuous.R

Value

A modified leaflet map

Note

If you synchronize multiple maps, a map may not yet be initialized and therefore cannot be used. Make sure to use addLeafletsync after all maps have been rendered.

References

https://github.com/jieter/Leaflet.Sync

See Also

Other leafletsync Functions: addLeafletsyncDependency(), isSynced(), leafletsyncOptions(), unsync()
addMapkeyMarkers

Value

A modified leaflet map

See Also

Other leafletsync Functions: addLeafletsync(), isSynced(), leafletsyncOptions(), unsync()

---

Add Mapkey Markers

Description

Add Mapkey Markers

Usage

addMapkeyMarkers(
  map,
  lng = NULL,
  lat = NULL,
  layerId = NULL,
  group = NULL,
  icon = NULL,
  popup = NULL,
  popupOptions = NULL,
  label = NULL,
  labelOptions = NULL,
  options = leaflet::markerOptions(),
  clusterOptions = NULL,
  clusterId = NULL,
  data = leaflet::getMapData(map)
)

Arguments

map the map to add mapkey Markers to.

lng a numeric vector of longitudes, or a one-sided formula of the form ~x where x is a variable in data; by default (if not explicitly provided), it will be automatically inferred from data by looking for a column named lng, long, or longitude (case-insensitively)

lat a vector of latitudes or a formula (similar to the lng argument; the names lat and latitude are used when guessing the latitude column from data)

layerId the layer id
addMapkeyMarkers

- **group**: the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g. markers and polygons) can share the same group name.

- **icon**: the icon(s) for markers;

- **popup**: a character vector of the HTML content for the popups (you are recommended to escape the text using `htmlEscape()` for security reasons);

- **popupOptions**: A Vector of `popupOptions` to provide popups;

- **label**: a character vector of the HTML content for the labels;

- **labelOptions**: A Vector of `labelOptions` to provide label options for each label. Default NULL;

- **options**: a list of extra options for markers. See `markerOptions`;

- **clusterOptions**: if not NULL, markers will be clustered using `Leaflet.markercluster`; you can use `markerClusterOptions()` to specify marker cluster options;

- **clusterId**: the id for the marker cluster layer;

- **data**: the data object from which the argument values are derived; by default, it is the data object provided to `leaflet()` initially, but can be overridden.

**Value**

the new `map` object

**References**

https://github.com/mapshakers/leaflet-mapkey-icon

**See Also**

Other Mapkey Functions: `.[.leaflet_mapkey_icon_set()`, `makeMapkeyIcon()`, `mapkeyIconList()`, `mapkeyIcons()`

**Examples**

```r
library(leaflet)

leaflet() %>%
  addTiles() %>%
  addMapkeyMarkers(data = breweries91,
                   icon = makeMapkeyIcon(icon = "mapkey",
                                          iconSize = 30,
                                          boxShadow = FALSE,
                                          background = "transparent"),
                   group = "mapkey",
                   label = ~state, popup = ~village)
```
addMovingMarker

Add Moving Markers

Description

The function expects either line or point data as spatial data or as Simple Feature. Alternatively, coordinates can also be passed as numeric vectors.

Usage

```r
addMovingMarker(
  map,
  lng = NULL,
  lat = NULL,
  layerId = NULL,
  group = NULL,
  duration = 2000,
  icon = NULL,
  popup = NULL,
  popupOptions = NULL,
  label = NULL,
  labelOptions = NULL,
  movingOptions = movingMarkerOptions(),
  options = leaflet::markerOptions(),
  data = leaflet::getMapData(map)
)
```

Arguments

- `map` the map to add moving markers
- `lng` a numeric vector of longitudes, or a one-sided formula of the form `~x` where `x` is a variable in data; by default (if not explicitly provided), it will be automatically inferred from data by looking for a column named `lng`, `long`, or `longitude` (case-insensitively)
- `lat` a vector of latitudes or a formula (similar to the `lng` argument; the names `lat` and `latitude` are used when guessing the latitude column from data)
- `layerId` In order to be able to address the moving markings individually, a layerId is required. If none is specified, one is created that is derived from the current timestamp.
- `group` the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g. markers and polygons) can share the same group name.
- `duration` Duration in milliseconds per line segment between 2 points. Can be a vector or a single number. Default is 1000
icon the icon(s) for markers;

popup a character vector of the HTML content for the popups (you are recommended to escape the text using \texttt{htmlEscape()} for security reasons)

popupOptions A Vector of \texttt{popupOptions} to provide popups

label a character vector of the HTML content for the labels

labelOptions A Vector of \texttt{labelOptions} to provide label options for each label. Default NULL

movingOptions a list of extra options for moving markers. See \texttt{movingMarkerOptions}

options a list of extra options for markers. See \texttt{markerOptions}

data the data object from which the argument values are derived; by default, it is the data object provided to \texttt{leaflet()} initially, but can be overridden

Value
the new map object

References
https://github.com/ewoken/Leaflet.MovingMarker

See Also
Other MovingMarker Functions: \texttt{movingMarkerOptions()}, \texttt{startMoving()}

Examples

library(sf)
library(leaflet)
library(leaflet.extras2)

\begin{verbatim}
df <- sf::st_as_sf(atlStorms2005)[1,]
leaflet() %>%
  addTiles() %>%
  addPolylines(data = df) %>%
  addMovingMarker(data = df,
                 movingOptions = movingMarkerOptions(autostart = TRUE, loop = TRUE),
                 label="I am a pirate!",
                 popup="Arrr")
\end{verbatim}
addOpenweatherCurrent

Usage

addOpenweatherCurrent(
  map,  
  apikey = NULL,  
  group = NULL,  
  layerId = NULL,  
  options = openweatherCurrentOptions()
)

Arguments

map    a map widget object created from leaflet()
apikey  a valid Openweathermap-API key. Get one from here.
group the name of the group the newly created layers should belong to (for clearGroup and addLayersControl purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g. markers and polygons) can share the same group name.
layerId the layer id
options List of further options. See openweatherCurrentOptions

Value

the new map object

Note

The current weather icons will appear beginning with zoom level 9 and if used in Shiny, a click on an icon will update a Shiny input at input$MAPID_owm_click.

References

https://github.com/trafficonese/leaflet-openweathermap

See Also

Other Openweathermap Functions: addOpenweatherTiles(), openweatherCurrentOptions(), openweatherOptions()

Examples

## Not run:
library(leaflet)
library(leaflet.extras2)
Sys.setenv("OPENWEATHERMAP" = 'Your_API_Key')

leaflet() %>%
  addTiles() %>% setView(9, 50, 9) %>%

addOpenweatherCurrent(options = openweatherCurrentOptions(
  lang = "en", popup = TRUE))

## End(Not run)

---

addOpenweatherTiles  
*Add OpenWeatherMap Tiles*

---

**Description**

Add OpenWeatherMap Tiles

**Usage**

```r
addOpenweatherTiles(
  map,
  apikey = NULL,
  layers = NULL,
  group = NULL,
  layerId = NULL,
  opacity = 0.5,
  options = openweatherOptions()
)
```

**Arguments**

- `map` a map widget object created from `leaflet()`
- `apikey` a valid OpenWeatherMap-API key. Get one from [here](#).
- `layers` character vector of layers you wish to add to the map. The following layers are currently possible: `c("clouds","cloudsClassic","precipitation","precipitationClassic","rain","rainClassic","snow","pressure","pressureContour","temperature","wind")`.
- `group` the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g. markers and polygons) can share the same group name.
- `layerId` the layer id
- `opacity` opacity of the layer
- `options` List of further options. See `openweatherOptions`

**Value**

the new map object
Note

Out of the box a legend image is only available for Pressure, Precipitation Classic, Clouds Classic, Rain Classic, Snow, Temperature and Wind Speed. Please add your own images if you need some more.

References

https://github.com/trafficonese/leaflet-openweathermap

See Also

Other Openweathermap Functions: addOpenweatherCurrent(), openweatherCurrentOptions(), openweatherOptions()

Examples

```r
## Not run:
library(leaflet)
library(leaflet.extras2)
Sys.setenv("OPENWEATHERMAP" = 'Your_API_Key')

leaflet() %>%
  addTiles() %>% setView(9, 50, 6) %>%
  addOpenweatherTiles(layers = "wind")

## End(Not run)
```

---

**addPlayback**

*Add Playback to Leaflet*

**Description**

The LeafletPlayback plugin provides the ability to replay GPS Points in the form of POINT Simple Features. Rather than simply animating a marker along a polyline, the speed of the animation is synchronized to a clock. The playback functionality is similar to a video player; you can start and stop playback or change the playback speed.

**Usage**

```r
addPlayback(
  map,
  data,
  time = "time",
  icon = NULL,
  pathOpts = pathOptions(),
  popup = NULL,
  label = NULL,
```
addPlayback

```r
popupOptions = NULL,
labelOptions = NULL,
options = playbackOptions()
```

**Arguments**

- `map` a map widget
- `data` data must be a POINT Simple Feature or a list of POINT Simple Feature’s with a time column.
- `time` The column name of the time column. Default is "time".
- `icon` an icon which can be created with `makeIcon`
- `pathOpts` style the CircleMarkers with `pathOptions`
- `popup` A formula with the column names for the popup content
- `label` A formula with the column names for the label content
- `popupOptions` A Vector of `popupOptions` to provide popups
- `labelOptions` A Vector of `labelOptions` to provide label options for each label. Default NULL
- `options` List of additional options. See `playbackOptions`

**Value**

the new map object

**Note**

If used in Shiny, you can listen to 2 events

- ‘map-ID’+”_pb_mouseover”
- ‘map-ID’+”_pb_click”

**References**

[https://github.com/hallahan/LeafletPlayback](https://github.com/hallahan/LeafletPlayback)

**See Also**

Other Playback Functions: `playbackOptions()`, `removePlayback()`

**Examples**

```r
## Not run:
library(leaflet)
library(leaflet.extras2)
library(sf)

## Single Elements
data <- sf::st_as_sf(leaflet::atlStorms2005[,1])
data <- st_cast(data, "POINT")
```
```r
# Data Time
data$time = as.POSIXct(
  seq.POSIXt(Sys.time() - 1000, Sys.time(), length.out = nrow(data)))
data$label <- as.character(data$time)

leaflet() %>%
  addTiles() %>%
  addPlayback(data = data, label = ~label,
              popup = ~sprintf("I am a popup for \textit{<b>%s</b> and <b>%s</b>},
                          Name, label),
              popupOptions = popupOptions(offset = c(0, -35)),
              options = playbackOptions(radius = 3,
                                         ticklen = 36000,
                                         speed = 50,
                                         maxInterpolationTime = 1000),
              pathOpts = pathOptions(weight = 5))

## Multiple Elements
data <- sf::st_as_sf(leaflet::atlStorms2005[1:5,])
data$Name <- as.character(data$Name)
data <- st_cast(data, "POINT")
data$time <- unlist(lapply(rle(data$Name)$lengths, function(x) {
  seq.POSIXt(as.POSIXct(Sys.Date()-2), as.POSIXct(Sys.Date()), length.out = x)
}))
data$time <- as.POSIXct(data$time, origin="1970-01-01")
data$label <- paste0("Time: ", data$time)
data$popup = sprintf("<h3>Customized Popup</h3><b>Name</b>: %s<br><b>Time</b>: %s",
                      data$Name, data$time)
data <- split(data, f = data$Name)

leaflet() %>%
  addTiles() %>%
  addPlayback(data = data,
              popup = ~popup,
              label = ~label,
              popupOptions = popupOptions(offset=c(0,-35)),
              labelOptions = labelOptions(noHide = TRUE),
              options = playbackOptions(radius = 3,
                                         ticklen = 1000000,
                                         speed = 5000,
                                         maxInterpolationTime = 10000,
                                         transitionpopup = FALSE,
                                         transitionlabel = FALSE,
                                         playCommand = "Let's go",
                                         stopCommand = "Stop it!",
                                         color = c("red","green","blue",
                                                   "orange","yellow")),
              pathOpts = pathOptions(weight = 5))

## End(Not run)
```
addReachability  

Add Isochrones to Leaflet

Description

A leaflet plugin which shows areas of reachability based on time or distance for different modes of travel using the openrouteservice isochrones API. Based on the leaflet.reachability plugin

Usage

addReachability(map, apikey = NULL, options = reachabilityOptions())

Arguments

- **map**: a map widget
- **apikey**: a valid Openrouteservice API-key. Can be obtained from Openrouteservice
- **options**: A list of further options. See reachabilityOptions

Value

the new map object

Note

When used in Shiny, 3 events update a certain shiny Input:

1. reachability:displayed updates input$MAPID_reachability_displayed
2. reachability:delete updates input$MAPID_reachability_delete
3. reachability:error updates input$MAPID_reachability_error

References

https://github.com/traffordDataLab/leaflet.reachability

See Also

Other Reachability Functions: reachabilityOptions(), removeReachability()

Examples

```r
## Not run:
library(leaflet)
library(leaflet.extras2)

Sys.setenv("OPRS" = 'Your_API_Key')

leaflet() %>%
  addTiles() %>%
  addReachability()
```
addSidebar

addSidebar

Add a Sidebar Leaflet Control

Description

The sidebar plugin only works in a reactive environment (e.g. Shiny), as the HTML must be created with `sidebar_tabs` and `sidebar_pane` before `leafletOutput` is called.

Usage

```r
addSidebar(map, id = "sidebar", options = list(position = "left"))
```

Arguments

- `map` A leaflet map widget
- `id` Id of the sidebar-div. Must match with the id of `sidebar_tabs`
- `options` A named list with the only option `position`, which should be either `left` or `right`.

Value

the new map object

References

https://github.com/Turbo87/sidebar-v2

See Also

Other Sidebar Functions: `closeSidebar()`, `openSidebar()`, `removeSidebar()`, `sidebar_pane()`, `sidebar_tabs()`

Examples

```r
## Not run:
library(shiny)

# run example app showing a single sidebar
runApp(paste0(system.file("examples", package = "leaflet.extras2"), 
           
# run example app showing two sidebars
runApp(paste0(system.file("examples", package = "leaflet.extras2"),
```
addSidebyside

"/multi_sidebar_app.R")

## End(Not run)

---

**addSidebyside**

**Add Side by Side View**

**Description**

A Leaflet control to add a split screen to compare two map overlays. The plugin works with Panes, see the example.

**Usage**

```r
addSidebyside(
  map,
  layerId = NULL,
  leftId = NULL,
  rightId = NULL,
  options = list(thumbSize = 42, padding = 0)
)
```

**Arguments**

- **map**: a map widget
- **layerId**: the layer id, needed for removeSidebyside
- **leftId**: the layerId of the Tile layer that should be visible on the left side
- **rightId**: the layerId of the Tile layer that should be visible on the right side
- **options**: A list of options. Currently only thumbSize and padding can be changed.

**Value**

the new map object

**Note**

It is currently not working correctly if the baseGroups are defined in addLayersControl.

**References**

[https://github.com/digidem/leaflet-side-by-side](https://github.com/digidem/leaflet-side-by-side)

**See Also**

Other Sidebyside Functions: removeSidebyside()
addSpinner

**Examples**

```r
library(leaflet)
library(leaflet.extras2)

leaflet(quakes) %>%
  addMapPane("left", zIndex = 0) %>%
  addMapPane("right", zIndex = 0) %>%
  addTiles(group = "base", layerId = "baseid",
           options = pathOptions(pane = "right")) %>%
  addProviderTiles(providers$CartoDB.DarkMatter, group="carto", layerId = "cartoid",
                   options = pathOptions(pane = "left")) %>%
  addCircleMarkers(data = breweries91[1:15,], color = "blue", group = "blue",
                   options = pathOptions(pane = "left")) %>%
  addCircleMarkers(data = breweries91[15:20,], color = "yellow", group = "yellow") %>%
  addCircleMarkers(data = breweries91[15:30,], color = "red", group = "red",
                   options = pathOptions(pane = "right")) %>%
  addLayersControl(overlayGroups = c("blue","red","yellow")) %>%
  addSidebyside(layerId = "sidecontrols",
                rightId = "baseid",
                leftId = "cartoid")
```

---

**Description**

Adds an animated loading spinning over the map.

**Usage**

```r
addSpinner(map)
startSpinner(map, options = NULL)
stopSpinner(map)
```

**Arguments**

- `map`: A map widget object created from `leaflet`

**Value**

A leaflet map object

**References**

https://github.com/makinacorpus/Leaflet.Spin
https://github.com/fgnass/spin.js
Examples

library(leaflet)
library(leaflet.extras2)

leaflet(data = quakes) %>%
  addTiles() %>%
  addSpinner() %>%
  startSpinner(options = list("lines" = 7, "length" = 20)) %>%
  addMarkers(~long, ~lat, popup = ~as.character(mag), label = ~as.character(mag)) %>%
  stopSpinner()

addTangram

**addTangram** adds a Tangram layer to a Leaflet map in a Shiny App.

Description

Adds a Tangram layer to a Leaflet map in a Shiny App.

Usage

addTangram(map, scene = NULL, layerId = NULL, group = NULL, options = NULL)

Arguments

- **map**: A leaflet map widget
- **scene**: Path to a required .yaml or .zip file. If the file is within the /www folder of a Shiny-App, only the filename must be given, otherwise the full path is needed. See the Tangram repository or the Tangram docs for further information on how to edit such a .yaml file.
- **layerId**: A layer ID
- **group**: The name of the group the newly created layer should belong to (for clearGroup and addLayersControl purposes).
- **options**: A list of further options. See the app in the examples/tangram folder or the docs for further information.

Value

the new map object

Note

Only works correctly in a Shiny-App environment.

References

https://github.com/tangrams/tangram
Examples

```r
## Not run:
library(shiny)
library(leaflet)
library(leaflet.extras2)

## In the /www folder of a ShinyApp. Must contain the Nextzen API-key
scene <- "scene.yaml"

ui <- fluidPage(leafletOutput("map"))

server <- function(input, output, session) {
  output$map <- renderLeaflet({
    leaflet() %>%
      addTiles(group = "base") %>%
      addTangram(scene = scene, group = "tangram") %>%
      addCircleMarkers(data = breweries91, group = "brews") %>%
      setView(11, 49.4, 14) %>%
      addLayersControl(baseGroups = c("tangram", "base"),
          overlayGroups = c("brews"))
  })
}

shinyApp(ui, server)

## End(Not run)
```

addTimeslider

Add Time Slider to Leaflet

Description

The LeafletSlider plugin enables you to dynamically add and remove Markers/Lines on a map by using a JQuery UI slider.

Usage

```r
addTimeslider(
  map,        
  data,       
  radius = 10,
  stroke = TRUE,
  color = "#03F",
  weight = 5,
  opacity = 0.5,
  fill = TRUE,
  fillColor = color,
  fillOpacity = 0.2,
```


dashArray = NULL,
popup = NULL,
popupOptions = NULL,
label = NULL,
labelOptions = NULL,
ordertime = TRUE,
options = timesliderOptions()
)

Arguments

map  
a map widget

data  
data must be a Simple Feature collection of type POINT or LINESTRING with a column of class Date or POSIXct.

radius  
a numeric vector of radii for the circles; it can also be a one-sided formula, in which case the radius values are derived from the data (units in meters for circles, and pixels for circle markers)

stroke  
whether to draw stroke along the path (e.g. the borders of polygons or circles)

color  
stroke color

weight  
stroke width in pixels

opacity  
stroke opacity (or layer opacity for tile layers)

fill  
whether to fill the path with color (e.g. filling on polygons or circles)

fillColor  
fill color

fillOpacity  
fill opacity

dashArray  
a string that defines the stroke dash pattern

popup  
a character vector of the HTML content for the popups (you are recommended to escape the text using htmlEscape() for security reasons)

popupOptions  
a Vector of popupOptions to provide popups

label  
a character vector of the HTML content for the labels

labelOptions  
a Vector of labelOptions to provide label options for each label. Default NULL

ordertime  
boolean value indicating whether to order the data by the time column. The slider will adopt the order of the timestamps. The default is TRUE.

options  
List of additional options. See timesliderOptions

Value

the new map object

References

https://github.com/dwilhelm89/LeafletSlider

See Also

Other Timeslider Functions: removeTimeslider(), timesliderOptions()
Examples

```r
## Not run:
library(leaflet)
library(leaflet.extras2)
library(sf)
library(geojsonsf)

data <- sf::st_as_sf(leaflet::atlStorms2005[1,])
data <- st_cast(data, "POINT")
data$time = as.POSIXct(
  seq.POSIXt(Sys.time() - 1000, Sys.time(), length.out = nrow(data))
)

leaflet() %>%
  addTiles() %>%
  addTimeslider(data = data,
                options = timesliderOptions(
                  position = "topright",
                  timeAttribute = "time",
                  range = TRUE)) %>%
  setView(-72, 22, 4)

## End(Not run)
```

---

### addVelocity

**Add Velocity Animation**

**Description**

Add velocity animated data to leaflet. Based on the `leaflet-velocity` plugin

**Usage**

```r
addVelocity(
  map, 
  layerId = NULL, 
  group = NULL, 
  content = NULL, 
  options = velocityOptions()
)
```

**Arguments**

- `map`
  - a map widget object created from `leaflet()`
- `layerId`
  - the layer id
- `group`
  - the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g. markers and polygons) can share the same group name.
addWMS

Add Queryable WMS Layer

Description

A Leaflet plugin for working with Web Map services, providing: single-tile/untiled/nontiled layers, shared WMS sources, and GetFeatureInfo-powered identify.

Usage

addWMS(
  map,
  baseUrl,
  layerId = NULL,
  group = NULL,
  options = WMSTileOptions(),
  attribution = NULL,
  content)

content the path or URL to a JSON file representing the velocity data or a data.frame which can be transformed to such a JSON file. Please see the demo files for some example data.

options List of further options. See velocityOptions

Value

the new map object

References

https://github.com/onaci/leaflet-velocity

See Also

Other Velocity Functions: removeVelocity(), setOptionsVelocity(), velocityOptions()
addWMS

```r
layers = NULL,
popupOptions = NULL,
checkempty = FALSE,
data = getMapData(map)
```

Arguments

- **map**
  - a map widget object created from `leaflet()`
- **baseUrl**
  - a base URL of the WMS service
- **layerId**
  - the layer id
- **group**
  - the name of the group the newly created layers should belong to (for `clearGroup` and `addLayersControl` purposes). Human-friendly group names are permitted—they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g., markers and polygons) can share the same group name.
- **options**
  - a list of extra options for tile layers, popups, paths (circles, rectangles, polygons, ...), or other map elements
- **attribution**
  - the attribution text of the tile layer (HTML)
- **layers**
  - comma-separated list of WMS layers to show
- **popupOptions**
  - List of popup options. See `popupOptions`. Default is NULL.
- **checkempty**
  - Should the returned HTML-content be checked for emptiness? If the HTML-body is empty no popup is opened. Default is FALSE
- **data**
  - the data object from which the argument values are derived; by default, it is the data object provided to `leaflet()` initially, but can be overridden

Value

- the new map object

References

- [https://github.com/heigeo/leaflet.wms](https://github.com/heigeo/leaflet.wms)

Examples

```r
library(leaflet)
library(leaflet.extras2)

leaflet() %>%
  addTiles(group = "base") %>%
  setView(9, 50, 5) %>%
  addWMS(baseUrl = "https://maps.dwd.de/geoserver/dwd/wms",
          layers = "dwd:BRD_1km_winndaten_10m",
          popupOptions = popupOptions(maxWidth = 600),
          checkempty = TRUE,
          options = WMSTileOptions(
            transparent = TRUE,
            ...)```
antpathOptions

Description

Additional list of options for 'ant-path' animated polylines.

Usage

antpathOptions(
  delay = 400,
  paused = FALSE,
  reverse = FALSE,
  hardwareAccelerated = FALSE,
  dashArray = c(10, 20),
  pulseColor = "#ffffff",
  lineCap = NULL,
  lineJoin = NULL,
  interactive = TRUE,
  pointerEvents = NULL,
  className = ""
)

Arguments

delay Add a delay to the animation flux. Default is 400
paused Should the animation be paused. Default is FALSE
reverse Defines if the flow follows the path order or not. Default is FALSE
hardwareAccelerated Makes the animation run with hardware acceleration. Default is FALSE
dashArray The size of the animated dashes. Default is c(10, 20)
pulseColor Adds a color to the dashed flux. Default is #ffffff
lineCap a string that defines shape to be used at the end of the stroke
lineJoin a string that defines shape to be used at the corners of the stroke
interactive whether the element emits mouse events
pointerEvents sets the pointer-events attribute on the path if SVG backend is used
className a CSS class name set on an element

Value

A list of options for addAntpath animated polylines
See Also

Other Antpath Functions: \texttt{addAntpath()}, \texttt{clearAntpath()}, \texttt{removeAntpath()}

\begin{table}
\begin{tabular}{ll}
\textbf{arrowheadOptions} & \textit{Arrowhead Options} \\
\end{tabular}
\end{table}

Description

Additional list of options for polylines with arrowheads. You can also pass options inherited from \texttt{L.Path}

Usage

\begin{verbatim}
arrowheadOptions(
    yawn = 60,
    size = "15%",
    frequency = "allvertices",
    proportionalToTotal = FALSE,
    offsets = NULL,
    perArrowheadOptions = NULL,
    ...
)
\end{verbatim}

Arguments

\begin{description}
\item[yawn] Defines the width of the opening of the arrowhead, given in degrees. The larger the angle, the wider the arrowhead.
\item[size] Determines the size of the arrowhead. Accepts three types of values:
\begin{itemize}
\item A string with the suffix 'm', i.e. '500m' will set the size of the arrowhead to that number of meters.
\item A string with the suffix '%', i.e. '15%' will render arrows whose size is that percentage of the size of the parent polyline. If the polyline has multiple segments, it will take the percent of the average size of the segments.
\item A string the suffix 'px', i.e. '20px' will render an arrowhead whose size stays at a constant pixel value, regardless of zoom level. Will look strange at low zoom levels or for smaller parent vectors. Ideal for larger parent vectors and at higher zoom levels.
\end{itemize}
\item[frequency] How many arrowheads are rendered on a polyline.
\begin{itemize}
\item 'allvertices' renders an arrowhead on each vertex.
\item 'endonly' renders only one at the end.
\item A numeric value renders that number of arrowheads evenly spaced along the polyline.
\item A string with suffix 'm', i.e. '100m' will render arrowheads spaced evenly along the polyline with roughly that many meters between each one.
\end{itemize}
\end{description}
- A string with suffix ‘px’, i.e. ‘30px’ will render arrowheads spaced evenly with roughly that many pixels between each, regardless of zoom level.

**proportionalToTotal**

Only relevant when size is given as a percent. Useful when frequency is set to ‘endOnly’. Will render the arrowheads with a size proportional to the entire length of the multi-segmented polyline, rather than proportional to the average length of all the segments.

**offsets**

Enables the developer to have the arrowheads start or end at some offset from the start and/or end of the polyline. This option can be a list with ‘start’ and ‘end’ names. The values must be strings defining the size of the offset in either meters or pixels, i.e. `list('start' = '100m', 'end' = '15px')`.

**perArrowheadOptions**

Enables the developer to customize arrowheads on a one-by-one basis. Must be in the form of a function of i, which is the index of the arrowhead as it is rendered in the loop through all arrowheads. Must return an options object. Cannot account for frequency or proportionalToTotal from within the perArrowheadOptions callback. See the example for details.

... Additional options for arrowheads, inherited from L.Path

**Value**

A list of options for addArrowhead polylines

**References**

https://github.com/slutske22/leaflet-arrowheads#options

**See Also**

Other Arrowhead Functions: `addArrowhead()`, `clearArrowhead()`, `removeArrowhead()`

---

**clearAntpath**

**Description**

Clear all Antpaths

**Usage**

`clearAntpath(map)`

**Arguments**

- `map` a map widget object, possibly created from `leaflet()` but more likely from `leafletProxy()`
**clearArrowhead**

**Description**

Remove arrowheads from Lines by group

**Usage**

clearArrowhead(map, group)

**Arguments**

- **map**: the map
- **group**: A group name

**Value**

A modified leaflet map

**See Also**

Other Antpath Functions: addAntpath(), antpathOptions(), removeAntpath()

---

**clearFuture**

**Description**

Resets the stack of future items.

**Usage**

clearFuture(map)

**Arguments**

- **map**: a map widget object created from `leafletProxy`
clearHexbin

Value
the new map object

References
https://github.com/cscott530/leaflet-history

See Also
Other History Functions: addHistory(), clearHistory(), goBackHistory(), goForwardHistory(), historyOptions()
clearHistory

Description
Resets the stack of history items.

Usage
clearHistory(map)

Arguments
map a map widget object created from leafletProxy

Value
the new map object

References
https://github.com/cscott530/leaflet-history

See Also
Other History Functions: addHistory(), clearFuture(), goBackHistory(), goForwardHistory(), historyOptions()

closeSidebar

Description
Close the Sidebar

Usage
closeSidebar(map, sidebar_id = NULL)

Arguments
map A leaflet map widget
sidebar_id The id of the sidebar (per sidebar_tabs). Defaults to NULL such that the first sidebar is opened.
Value
the new map object

See Also
Other Sidebar Functions: addSidebar(), openSidebar(), removeSidebar(), sidebarPane(), sidebarTabs()

context_mapmenuItems context_mapmenuItems

Description
context_mapmenuItems

Usage
context_mapmenuItems(...)

Arguments
... contextmenu item/s

Value
A list of context_menuItem for the map

See Also
Other Contextmenu Functions: addItemContextmenu(), context_mapmenuItems(), contextMenuItem(), hideContextmenu(), insertItemContextmenu(), menuItems(), removeItemContextmenu(), removeAllItemsContextmenu(), setDisabledContextmenu(), showContextmenu()

context_markermenuItems context_markermenuItems

Description
context_markermenuItems

Usage
context_markermenuItems(...)

Arguments
... contextmenu item/s

Value
A list of context_menuItem for the map

See Also
Other Contextmenu Functions: addItemContextmenu(), context_mapmenuItems(), contextMenuItem(), hideContextmenu(), insertItemContextmenu(), menuItems(), removeItemContextmenu(), removeAllItemsContextmenu(), setDisabledContextmenu(), showContextmenu()
context_menuItem

Arguments

... context_menuItem/s

Value

A list of context_menuItem for markers

See Also

Other Contextmenu Functions: addItemContextmenu(), context_menuItem(), context_mapmenuItems(), context_markermenuItems(), hideContextmenu(), insertItemContextmenu(), mapmenuItems(), markermenuItems(), menuItem(), removeItemContextmenu(), removeAllItemsContextmenu(), setDisabledContextmenu(), showContextmenu()

context_menuItem context_menuItem

Description

context_menuItem

Usage

context_menuItem(text, callback = NULL, ...)

Arguments

text The label to use for the menu item

callback A callback function to be invoked when the menu item is clicked. The callback is passed an object with properties identifying the location the menu was opened at: latlng, layerPoint and containerPoint. The callback-function must be valid JavaScript and will be wrapped in JS.

... For further options please visit https://github.com/aratcliffe/Leaflet.

context_menuItem

Value

A contextmenu item list

See Also

Other Contextmenu Functions: addItemContextmenu(), context_menuItem(), context_mapmenuItems(), context.markermenuItems(), hideContextmenu(), insertItemContextmenu(), mapmenuItems(), markermenuItems(), menuItem(), removeItemContextmenu(), removeAllItemsContextmenu(), setDisabledContextmenu(), showContextmenu()
Description

Print or export a map programmatically (e.g. in a Shiny environment).

Usage

easyprintMap(map, sizeModes = "A4Portrait", filename = "map")

Arguments

map the map widget

sizeModes Must match one of the given sizeMode names in easyprintOptions. The options are: CurrentSize, A4Portrait or A4Landscape. If you want to print the map with a Custom sizeMode you need to pass the Custom className. Default is A4Portrait

filename Name of the file if exportOnly option is TRUE.

Value

A leaflet map object

See Also

Other EasyPrint Functions: addEasyprint(), easyprintOptions(), removeEasyprint()

Examples

## Only run examples in interactive R sessions
if (interactive()) {
library(shiny)
library(leaflet)
library(leaflet.extras2)

ui <- fluidPage(
  leafletOutput("map"),
  selectInput("scene", "Select Scene", choices = c("CurrentSize", "A4Landscape", "A4Portrait")),
  actionButton("print", "Print Map")
)

server <- function(input, output, session) {
  output$map <- renderLeaflet({
    input$print
    leaflet() %>%
    addTiles() %>%
    setView(10, 50, 9) %>%
  })
}
Description

Create a list of further options for the easyprint plugin.

Usage

easyprintOptions(
  title = "Print map",
  position = "topleft",
  sizeModes = list("A4Portrait", "A4Landscape", "CurrentSize"),
  defaultSizeTitles = NULL,
  exportOnly = FALSE,
  tileLayer = NULL,
  tileWait = 500,
  filename = "map",
  hidden = FALSE,
  hideControlContainer = TRUE,
  hideClasses = NULL,
  customWindowTitle = NULL,
  spinnerBgColor = "#0DC5C1",
  customSpinnerClass = "epLoader"
)

Arguments

title Sets the text which appears as the tooltip of the print/export button
position Positions the print button
sizeModes Either a character vector with one of the following options: CurrentSize, A4Portrait, A4Landscape. If you want to include a custom size mode you need to pass a named list, with width, height, name and className and assign a background-image in CSS. See the example in ./inst/examples/easyprint_app.R.
defaultSizeTitles  Button tooltips for the default page sizes
eexportOnly  If set to TRUE the map is exported to a .png file
tileLayer  The group name of one tile layer that you can wait for to draw (helpful when resizing)
tileWait  How long to wait for the tiles to draw (helpful when resizing)
filename  Name of the file if exportOnly option is TRUE
hidden  Set to TRUE if you don’t want to display the toolbar. Instead you can create your own buttons or fire print events programmatically.
hideControlContainer  Hides the leaflet controls like the zoom buttons and the attribution on the print out
hideClasses  Use a character vector or list of CSS-classes to hide on the output image.
customWindowTitle  A title for the print window which will get added to the printed paper
spinnerBgColor  A valid css colour for the spinner background color
customSpinnerClass  A class for a custom css spinner to use while waiting for the print.

 Value
A list of options for the ‘easyprint’ control

 References
https://github.com/rowanwins/leaflet-easyPrint

 See Also
Other EasyPrint Functions: addEasyprint(), easyprintMap(), removeEasyprint()

<table>
<thead>
<tr>
<th>gibs_layers</th>
<th>The available GIBS layers with attributes</th>
</tr>
</thead>
</table>

 Description
The available GIBS layers with attributes

 Usage
gibs_layers

 Format
An object of class data.frame with 276 rows and 4 columns.
Description
If possible, will go to previous map extent. Pushes current extent to the "future" stack.

Usage
goBackHistory(map)

Arguments
map a map widget object created from leafletProxy

Value
the new map object

References
https://github.com/cscott530/leaflet-history

See Also
Other History Functions: addHistory(), clearFuture(), clearHistory(), goForwardHistory(), historyOptions()
References

https://github.com/cscott530/leaflet-history

See Also

Other History Functions: addHistory(), clearFuture(), clearHistory(), goBackHistory(), historyOptions()

heightgraphOptions

Description

Customize the heightgraph with the following additional options.

Usage

heightgraphOptions(
  position = c("bottomright", "topleft", "topright", "bottomleft"),
  width = 800,
  height = 200,
  margins = list(top = 10, right = 30, bottom = 55, left = 50),
  expand = TRUE,
  expandCallback = NULL,
  mappings = NULL,
  highlightStyle = list(color = "red"),
  translation = NULL,
  xTicks = 3,
  yTicks = 3
)

Arguments

position  position of control: "topleft" , "topright" , "bottomleft" , or "bottomright" . Default is bottomright.
width     The width of the expanded heightgraph display in pixels. Default is 800.
height    The height of the expanded heightgraph display in pixels. Default is 200.
margins   The margins define the distance between the border of the heightgraph and the actual graph inside. You are able to specify margins for top, right, bottom and left in pixels. Default is list(top = 10, right = 30, bottom = 55, left = 50).
expand    Boolean value that defines if the heightgraph should be expanded on creation. Default is 200.
expandCallback Function to be called if the heightgraph is expanded or reduced. The state of the heightgraph is passed as an argument. It is TRUE when expanded and FALSE when reduced. Default is NULL.
hexbinOptions

mappings  You may add a mappings object to customize the colors and labels in the height graph. Without adding custom mappings the segments and labels within the graph will be displayed in random colors. Each key of the object must correspond to the summary key in properties within the FeatureCollection. Default is NULL.

highlightStyle You can customize the highlight style when using the horizontal line to find parts of the route above an elevation value. Use any Leaflet Path options as value of the highlightStyle parameter. Default is list(color = "red").

translation You can change the labels of the heightgraph info field by passing translations for distance, elevation, segment_length, type and legend. Default is NULL.

xTicks Specify the tick frequency in the x axis of the graph. Corresponds approximately to 2 to the power of value ticks. Default is 3.

yTicks Specify the tick frequency in the y axis of the graph. Corresponds approximately to 2 to the power of value ticks. Default is 3.

Value

A list of further options for addHeightgraph

See Also

Other Heightgraph Functions: addHeightgraph()

Description

A list of options for customizing the appearance/behavior of the hexbin layer.

Usage

```r
hexbinOptions(
  duration = 200,
  colorScaleExtent = NULL,
  radiusScaleExtent = NULL,
  colorRange = c("#f7fbff", "#08306b"),
  radiusRange = c(5, 15),
  pointerEvents = "all",
  resizetoCount = FALSE,
  tooltip = "Count "
)
```
Arguments

- **duration**: Transition duration for the hexbin layer
- **colorScaleExtent**: Extent of the color scale for the hexbin layer. This is used to override the derived extent of the color values and is specified as a vector of the form `c(min= numeric, max= numeric)`. Can be a numeric vector or a custom JS array, like `(JS("[40, undefined]"))`
- **radiusScaleExtent**: This is the same exact configuration option as colorScaleExtent, only applied to the radius extent.
- **colorRange**: Sets the range of the color scale used to fill the hexbins on the layer.
- **radiusRange**: Sets the range of the radius scale used to size the hexbins on the layer.
- **pointerEvents**: This value is passed directly to an element-level css style for pointer-events. You should only modify this config option if you want to change the mouse event behavior on hexbins. This will modify when the events are propagated based on the visibility state and/or part of the hexbin being hovered.
- **resizetoCount**: Resizes the hexbin to the count. Default is FALSE. If set to TRUE it will resize based on the amount of underlying elements. You can also pass a custom JS function.
- **tooltip**: Should tooltips be displayed? If set to TRUE, it will show the amount of underlying elements. If a string is given, it will append the string before the count. To disable tooltips, please pass NULL or FALSE. You can also pass a custom JS function.

Value

A list of hexbin-specific options

See Also

Other Hexbin-D3 Functions: `addHexbin()`, `clearHexbin()`, `hideHexbin()`, `showHexbin()`, `updateHexbin()`
hideHexbin

Value
A leaflet map object

See Also
Other Contextmenu Functions: addContextmenu(), addItemContextmenu(), context_mapmenuItems(), context_markermenuItems(), context_menuItem(), insertItemContextmenu(), mapmenuItems(), markermenuItems(), menuItem(), removeItemContextmenu(), removeAllItemsContextmenu(), setDisabledContextmenu(), showContextmenu()

hideHexbin  hideHexbin

Description
Hide the hexbinLayer.

Usage
hideHexbin(map)

Arguments
map The map widget

Value
the new map object

See Also
Other Hexbin-D3 Functions: addHexbin(), clearHexbin(), hexbinOptions(), showHexbin(), updateHexbin()

historyOptions History Options

Description
History Options
Usage

```r
historyOptions(
  position = c("topright", "topleft", "bottomleft", "bottomright"),
  maxMovesToSave = 10,
  backImage = "fa fa-caret-left",
  forwardImage = "fa fa-caret-right",
  backText = "",
  forwardText = "",
  backTooltip = "Go to Previous Extent",
  forwardTooltip = "Go to Next Extent",
  backImageBeforeText = TRUE,
  forwardImageBeforeText = FALSE,
  orientation = c("horizontal", "vertical"),
  shouldSaveMoveInHistory = NULL
)
```

Arguments

- **position**: Set the position of the History control. Default is topright.
- **maxMovesToSave**: Number of moves in the history to save before clearing out the oldest. Default value is 10, use 0 or a negative number to make unlimited.
- **backImage**: The class for the 'back' button icon. Default is "fa fa-caret-left".
- **forwardImage**: The class for the 'forward' button icon. Default is "fa fa-caret-right".
- **backText**: The text in the buttons. Default is "".
- **forwardText**: The text in the buttons. Default is "".
- **backTooltip**: Tooltip content. Default is "Go to Previous Extent".
- **forwardTooltip**: Tooltip content. Default is "Go to Next Extent".
- **backImageBeforeText**: When both text and image are present, whether to show the image first or the text first (left to right). Default is TRUE
- **forwardImageBeforeText**: When both text and image are present, whether to show the image first or the text first (left to right). Default is FALSE
- **orientation**: Whether to position the buttons on top of one another or side-by-side. Default is horizontal
- **shouldSaveMoveInHistory**: A JS callback you can provide that gets called with every move. return false to not save a move.

Value

A list of further options for addHistory

References

https://github.com/cscott530/leaflet-history
**insertItemContextmenu**

**See Also**

Other History Functions: `addHistory()`, `clearFuture()`, `clearHistory()`, `goBackHistory()`, `goForwardHistory()`

**Examples**

```r
library(leaflet)
leaflet() %>%
  addTiles() %>%
  addHistory(options = historyOptions(position = "bottomright",
    maxMovesToSave = 20,
    backText = "Go back",
    forwardText = "Go forward",
    orientation = "vertical"
  ))
```

**Description**

Insert a new contextmenu menu item at a specific index

**Usage**

```
insertItemContextmenu(map, option, index)
```

**Arguments**

- **map**: a map widget object created from `leaflet`
- **option**: new menu item to add
- **index**: Index of the contextmenu. (NOTE: Since the index is passed to JavaScript, it is zero-based)

**Value**

A leaflet map object

**See Also**

Other Contextmenu Functions: `addContextmenu()`, `addItemContextmenu()`, `context_mapmenuItems()`, `context_markermenuItems()`, `context_menuItem()`, `hideContextmenu()`, `mapmenuItems()`, `markermenuItems()`, `menuItem()`, `removeItemContextmenu()`, `removeallItemsContextmenu()`, `setDisabledContextmenu()`, `showContextmenu()`
isSynced is a map synchronized?

Description
Is a map synchronized with any or a specific map? Invoking this method sets a Shiny input that returns TRUE when the map is synchronized with another map. If syncwith is set, TRUE is returned if the map is synchronized exactly with that other map.

Usage
isSynced(map, id = NULL, syncwith = NULL)

Arguments
map the map
id The map id
syncwith Is the map synchronized with one of these maps?

Details
The Shiny input name is combined of the map-id and "_synced". For a map with id map1 the input can be retrieved with input$map1_synced.

Value
A map

See Also
Other leafletsync Functions: addLeafletsyncDependency(), addLeafletsync(), leafletsyncOptions(), unsync()
Description

Additional list of options.

Usage

```r
leafletsyncOptions(
  noInitialSync = FALSE,
  syncCursor = TRUE,
  offsetFn = JS("function (center, zoom, refMap, tgtMap) { return center; }")
)
```

Arguments

- `noInitialSync` Setting to TRUE disables initial synchronization of the maps. The default is FALSE.
- `syncCursor` The default TRUE adds a circle marker on the synced map.
- `offsetFn` A JavaScript-function to compute an offset for the center.

Value

A list of options for `addLeafletsync`

See Also

Other leafletsync Functions: `addLeafletsyncDependency()`, `addLeafletsync()`, `isSynced()`, `unsync()`

Description

Make Mapkey Icon
makeMapkeyIcon

Usage

makeMapkeyIcon(
  icon = "mapkey",
  color = "#ff0000",
  iconSize = 12,
  background = "#1F7499",
  borderRadius = "100%",
  hoverScale = 1.4,
  hoverEffect = TRUE,
  additionalCSS = NULL,
  hoverCSS = NULL,
  htmlCode = NULL,
  boxShadow = TRUE
)

Arguments

- **icon** ID of the mapkey Icon you want to use. See [mapkeyicons.com](https://mapkeyicons.com) for a full list.
- **color** Any CSS color (e.g. 'red','rgba(20,160,90,0.5)', '#686868', ...)
- **iconSize** Size of Icon in Pixels. Default is 12
- **background** Any CSS color or false for no background
- **borderRadius** Any number (for circle size/2, for square 0.001)
- **hoverScale** Any real number (best result in range 1 - 2, use 1 for no effect)
- **hoverEffect** Switch on/off effect on hover
- **additionalCSS** CSS code (e.g. "border:4px solid #aa3838;")
- **hoverCSS** CSS code (e.g. "background-color:#992b00 !important; color:#99defc !important;")
- **htmlCode** e.g. '&57347;&#xe003;'. See [mapkeyicons.com](https://mapkeyicons.com) for further information
- **boxShadow** Should a shadow be visible

Value

A list of mapkey-icon data that can be passed to the argument icon

References

[https://github.com/mapshakers/leaflet-mapkey-icon](https://github.com/mapshakers/leaflet-mapkey-icon)

See Also

Other Mapkey Functions: [.leaflet_mapkey_icon_set()](https), addMapkeyMarkers() , mapkeyIconList() , mapkeyIcons()
mapkeyIconList

Examples

makeMapkeyIcon(icon = "traffic_signal",
               color = "#0000ff",
               iconSize = 12,
               boxShadow = FALSE,
               background="transparent")

mapkeyIconList Make Mapkey-icon set

Description

Make Mapkey-icon set

Usage

mapkeyIconList(...)

Arguments

... icons created from makeMapkeyIcon()

Value

A list of class "leaflet_mapkey_icon_set"

References

https://github.com/mapshakers/leaflet-mapkey-icon

See Also

Other Mapkey Functions: [.leaflet_mapkey_icon_set(), addMapkeyMarkers(), makeMapkeyIcon(),
mapkeyIcons()]

Examples

iconSet = mapkeyIconList(
    red = makeMapkeyIcon(color = "#ff0000"),
    blue = makeMapkeyIcon(color = "#0000ff")
)
iconSet[c("red", "blue")]

mapkeyIcons

Create a list of Mapkey icon data

Description

An icon can be represented as a list of the form `list(color, iconSize,...)`. This function is vectorized over its arguments to create a list of icon data. Shorter argument values will be re-cycled. NULL values for these arguments will be ignored.

Usage

```r
mapkeyIcons(
    icon = "mapkey",
    color = "#ff0000",
    iconSize = 12,
    background = "#1F7499",
    borderRadius = "100%",
    hoverScale = 1.4,
    hoverEffect = TRUE,
    hoverCSS = NULL,
    additionalCSS = NULL,
    htmlCode = NULL,
    boxShadow = TRUE
)
```

Arguments

- **icon**: ID of the mapkey Icon you want to use. See mapkeyicons.com for a full list.
- **color**: Any CSS color (e.g. 'red', 'rgba(20,160,90,0.5)', '#686868', ...)
- **iconSize**: Size of Icon in Pixels. Default is 12
- **background**: Any CSS color or false for no background
- **borderRadius**: Any number (for circle size/2, for square 0.001)
- **hoverScale**: Any real number (best result in range 1 - 2, use 1 for no effect)
- **hoverEffect**: Switch on/off effect on hover
- **hoverCSS**: CSS code (e.g. "background-color:#992b00 !important; color:#99defc !important;")
- **additionalCSS**: CSS code (e.g. "border:4px solid #aa3838;")
- **htmlCode**: e.g. '&#57347;&#xe003;'. See mapkeyicons.com for further information
- **boxShadow**: Should a shadow be visible

Value

A list of mapkey-icon data that can be passed to the argument `icon`
mapmenuItems

References

https://github.com/mapshakers/leaflet-mapkey-icon

See Also

Other Mapkey Functions: \texttt{.leaflet_mapkey_icon_set()}, \texttt{addMapkeyMarkers()}, \texttt{makeMapkeyIcon()}, \texttt{mapkeyIconList()}

Examples

```r
library(leaflet)

leaflet() %>%
  addMapkeyMarkers(data = breweries91,
  icon = mapkeyIcons(
    color = "red",
    borderRadius = 0,
    iconSize = 25))
```

Description

mapmenuItems

Usage

mapmenuItems(...) 

Arguments

... contextmenu item/s

Value

A list of menuItem for the map

See Also

Other Contextmenu Functions: \texttt{addContextMenu()}, \texttt{addItemContextMenu()}, \texttt{context_mapmenuItems()}, \texttt{context_markerMenuItems()}, \texttt{context_menuItem()}, \texttt{hideContextMenu()}, \texttt{insertItemContextMenu()}, \texttt{markerMenuItems()}, \texttt{menuItem()}, \texttt{removeItemContextMenu()}, \texttt{removeAllItemsContextMenu()}, \texttt{setDisabledContextMenu()}, \texttt{showContextMenu()}

menuItem

menuItem

Description

menuItem

Usage

menuItem(text, callback = NULL, ...)

Arguments

text

callback

... For further options please visit https://github.com/aratcliffe/Leaflet.

contextmenu
movingMarkerOptions

Value

A contextmenu item list

See Also

Other Contextmenu Functions: addContextmenu(), addItemContextmenu(), context_mapmenuItems(), context_markermenuItems(), context_menuItem(), hideContextmenu(), insertItemContextmenu(), mapmenuItems(), markermenuItems(), removeItemContextmenu(), removeallItemsContextmenu(), setDisabledContextmenu(), showContextmenu()
openSidebar | Open the Sidebar by ID

Description

Open the Sidebar by ID

Usage

openSidebar(map, id, sidebar_id = NULL)

Arguments

- **map**: A leaflet map widget
- **id**: The id of the `sidebar_pane` to open.
- **sidebar_id**: The id of the sidebar (per `sidebar_tabs`). Defaults to NULL such that the first sidebar is opened.

Value

the new map object

See Also

Other Sidebar Functions: `addSidebar()`, `closeSidebar()`, `removeSidebar()`, `sidebar_pane()`, `sidebar_tabs()`

---

openweatherCurrentOptions

openweatherCurrentOptions

Description

openweatherCurrentOptions

Usage

openweatherCurrentOptions(lang = "en", minZoom = 7, interval = 10, ...)

---
openweatherOptions

Arguments

- **lang**: 'en', 'de', 'ru', 'fr', 'es', 'ca'. Language of popup texts. Note: not every translation is finished yet.
- **minZoom**: Number (7). Minimal zoom level for fetching city data. Use smaller values only at your own risk.
- **interval**: Number (0). Time in minutes to reload city data. Please do not use less than 10 minutes.
- ... Further options passed to \L.OWM.current. See the full list of options

Value

A list of options for addOpenweatherCurrent

See Also

Other Openweathermap Functions: addOpenweatherCurrent(), addOpenweatherTiles(), openweatherOptions()

openweatherOptions  
*OpenWeatherMap Options*

Description

OpenWeatherMap Options

Usage

openweatherOptions(
  showLegend = TRUE,
  legendImagePath = NULL,
  legendPosition = c("bottomleft", "bottomright", "topleft", "topright")
)

Arguments

- **showLegend**: If TRUE and option legendImagePath is set there will be a legend image on the map
- **legendImagePath**: A URL (is set to a default image for some layers, null for others, see below). URL or relative path to an image which is a legend to this layer
- **legendPosition**: Position of the legend images on the map. Must be one of 'bottomleft', 'bottomright', 'topleft', 'topright'

Value

A list of options for addOpenweatherTiles
See Also

Other Openweathermap Functions: addOpenweatherCurrent(), addOpenweatherTiles(), openweatherCurrentOptions()
reachabilityOptions

staleTime  Set time before a track is considered stale and faded out. Default is 60*60*1000 (1 hour)
transitionpopup  Should the position of the popup move smoothly, like the marker icon? Default: TRUE
transitionlabel  Should the position of the label move smoothly, like the marker icon? Default: TRUE
...  Further arguments passed to 'L.Playback'

Value
A list of options for addPlayback

References
https://github.com/hallahan/LeafletPlayback

See Also
Other Playback Functions: addPlayback(), removePlayback()
Value

A list of options for `addReachability`

References

https://github.com/traffordDataLab/leaflet.reachability

See Also

Other Reachability Functions: `addReachability()`, `removeReachability()`

---

```r
removeallItemsContextmenu
```

Description

Remove all contextmenu items from the map.

Usage

```r
removeallItemsContextmenu(map)
```

Arguments

- `map`: a map widget object created from `leaflet`

Value

A leaflet map object

See Also

Other Contextmenu Functions: `addContextmenu()`, `addItemContextmenu()`, `context_mapmenuItems()`, `context_markermenuItems()`, `context_menuItem()`, `hideContextmenu()`, `insertItemContextmenu()`, `mapmenuItems()`, `markermenuItems()`, `menuItem()`, `removeItemContextmenu()`, `setDisabledContextmenu()`, `showContextmenu()`
removeAntpath

Description
Remove one or more Antpaths from a map, identified by layerId.

Usage
removeAntpath(map, layerId = NULL)

Arguments
- map: a map widget object, possibly created from leaflet() but more likely from leafletProxy()
- layerId: character vector; the layer id(s) of the item to remove

Value
the new map object

See Also
Other Antpath Functions: addAntpath(), antpathOptions(), clearAntpath()

removeArrowhead

Description
Remove arrowheads from Lines by layerId

Usage
removeArrowhead(map, layerId)

Arguments
- map: the map
- layerId: A single layerId or a vector of layerId’s

Value
A modified leaflet map

See Also
Other Arrowhead Functions: addArrowhead(), arrowheadOptions(), clearArrowhead()
### removeEasyprint

**Description**

Removes the easyprint control from the map.

**Usage**

```javascript
removeEasyprint(map)
```

**Arguments**

- **map**
  - the map widget

**Value**

A leaflet map object

**See Also**

Other EasyPrint Functions: `addEasyprint()`, `easyprintMap()`, `easyprintOptions()`

---

### removeItemContextmenu

**Description**

Remove a contextmenu item by index.

**Usage**

```javascript
removeItemContextmenu(map, index)
```

**Arguments**

- **map**
  - a map widget object created from `leaflet`
- **index**
  - Index of the contextmenu. (NOTE: Since the index is passed to JavaScript, it is zero-based)

**Value**

A leaflet map object
**removePlayback**

**See Also**

Other Contextmenu Functions: `addContextmenu()`, `addItemContextMenu()`, `context_mapmenuItems()`, `context_markermenuItems()`, `context_menuItem()`, `hideContextMenu()`, `insertItemContextMenu()`, `mapmenuItems()`, `markermenuItems()`, `menuItem()`, `removeAllItemsContextMenu()`, `setDisabledContextMenu()`, `showContextMenu()`

---

**Description**

Remove the Playback controls and markers.

**Usage**

```python
removePlayback(map)
```

**Arguments**

- `map` the map widget

**Value**

the new map object

**See Also**

Other Playback Functions: `addPlayback()`, `playbackOptions()`

---

**removeReachability**

**Description**

Remove the reachability controls.

**Usage**

```python
removeReachability(map)
```

**Arguments**

- `map` the map widget.

**Value**

the new map object
See Also

Other Reachability Functions: `addReachability()`, `reachabilityOptions()`

---

**removeSidebar**

*Remove the Sidebar*

---

**Description**

Remove the Sidebar

**Usage**

```r
removeSidebar(map, sidebar_id = NULL)
```

**Arguments**

- `map`: A leaflet map widget
- `sidebar_id`: The id of the sidebar (per `sidebar_tabs`). Defaults to NULL such that the first sidebar is opened.

**Value**

the new map object

**See Also**

Other Sidebar Functions: `addSidebar()`, `closeSidebar()`, `openSidebar()`, `sidebar_pane()`, `sidebar_tabs()`

---

**removeSidebyside**

**Description**

removeSidebyside

**Usage**

```r
removeSidebyside(map, layerId = NULL)
```

**Arguments**

- `map`: a map widget
- `layerId`: the layer id of the `addSidebyside` layer
removeTimeslider

Value
the new map object

See Also
Other Sidebyside Functions: addSidebyside()

Description
Remove the Timeslider controls and markers.

Usage
removeTimeslider(map)

Arguments
map the map widget

Value
the new map object

See Also
Other Timeslider Functions: addTimeslider(), timesliderOptions()

removeVelocity

Description
removeVelocity

Usage
removeVelocity(map, group)

Arguments
map the map widget
group the group to remove
**setDisabledContextmenu**

Value

the new map object

See Also

Other Velocity Functions: `addVelocity()`, `setOptionsVelocity()`, `velocityOptions()`

---

**setDate**

*Set Date for GIBS Layers*

Description

Set a new date for multi-temporal layers.

Usage

```
setDate(map, layers = NULL, dates = NULL)
```

Arguments

- `map` a map widget object created from `leaflet()`
- `layers` A character vector of GIBS-layers. See `gibs_layers`
- `dates` Date object. If multiple layers are added, you can add a Date vector of the same length

Value

the new map object

See Also

Other GIBS Functions: `addGIBS()`, `setTransparent()`

---

**setDisabledContextmenu**

*setDisabledContextmenu*

Description

Enable/Disable a contextmenu item by index.

Usage

```
setDisabledContextmenu(map, index, disabled = TRUE)
```
setOptionsVelocity

Arguments

- map: a map widget object created from `leaflet`
- index: Index of the contextmenu. (NOTE: Since the index is passed to JavaScript, it is zero-based)
- disabled: Set to `TRUE` to disable the element and `FALSE` to enable it. Default is `TRUE`

Value

A leaflet map object

See Also

Other Contextmenu Functions: `addContextmenu()`, `addItemContextmenu()`, `context_mapmenuItems()`, `context_markermenuItems()`, `context_menuItem()`, `hideContextmenu()`, `insertItemContextmenu()`, `mapmenuItems()`, `markermenuItems()`, `menuItem()`, `removeItemContextmenu()`, `removeallItemsContextmenu()`, `showContextmenu()`

---

setOptionsVelocity setOptionsVelocity

Description

setOptionsVelocity

Usage

`setOptionsVelocity(map, layerId, options)`

Arguments

- map: the map widget
- layerId: the layer id
- options: see `velocityOptions`

Value

the new map object

See Also

Other Velocity Functions: `addVelocity()`, `removeVelocity()`, `velocityOptions()`
setTransparent  

*Set Transparency for GIBS Layers*

**Description**

Change the transparency for no-data pixels.

**Usage**

```
setTransparent(map, layers = NULL, transparent = TRUE)
```

**Arguments**

- `map`: a map widget object created from `leaflet()`.
- `layers`: A character vector of GIBS-layers. See `gibs_layers`.
- `transparent`: Should the layer be transparent. If multiple layers are added, you can add a boolean vector of the same length.

**Value**

the new map object

**See Also**

Other GIBS Functions: `addGIBS()`, `setDate()`

---

showContextmenu  

**showContextmenu**

**Description**

Open the contextmenu at certain lat/lng-coordinates.

**Usage**

```
showContextmenu(map, lat = NULL, lng = NULL, data = leaflet::getMapData(map))
```

**Arguments**

- `map`: a map widget object created from `leaflet()`.
- `lat`: a vector of latitudes or a formula (similar to the lng argument; the names lat and latitude are used when guessing the latitude column from data).
- `lng`: a numeric vector of longitudes, or a one-sided formula of the form `~x` where `x` is a variable in data; by default (if not explicitly provided), it will be automatically inferred from data by looking for a column named lng, long, or longitude (case-insensitively).
- `data`: the data object from which the argument values are derived; by default, it is the data object provided to `leaflet()` initially, but can be overridden.
**showHexbin**

**Value**

A leaflet map object

**See Also**

Other Contextmenu Functions: `addContextmenu()`, `addItemContextmenu()`, `context_mapmenuItems()`, `context_markermenuItems()`, `context_menuItem()`, `hideContextmenu()`, `insertItemContextmenu()`, `mapmenuItems()`, `markermenuItems()`, `menuItem()`, `removeItemContextmenu()`, `removeallItemsContextmenu()`, `setDisabledContextmenu()`

---

**sidebar_pane**

Create a Sidebar Pane

**Description**

Create a Sidebar Pane
Usage

`sidebar_pane(
    title = "Sidebar Title",
    id = NULL,
    icon = icon("caret-right"),
    ...
)
``

Arguments

- **title**: A title for the sidebar panel
- **id**: An id for the sidebar panel
- **icon**: An icon for the sidebar panel
- **...**: List of elements to include in the panel

Value

A `shiny.tag` with sidebar-specific HTML classes

References

- [https://github.com/Turbo87/sidebar-v2](https://github.com/Turbo87/sidebar-v2)

See Also

Other Sidebar Functions: `addSidebar()`, `closeSidebar()`, `openSidebar()`, `removeSidebar()`, `sidebar_tabs()`

Examples

```r
## Not run:
library(shiny)
sidebar_pane(id = "id", icon = icon("cars"), tags$div())
## End(Not run)
```

---

**sidebar_tabs**  
*Create a Sidebar*

Description

Create a Sidebar

Usage

`sidebar_tabs(id = "sidebar", iconList = NULL, ...)"
Arguments

- **id**: The id of the sidebar, which must match the id of `addSidebar`. Default is "sidebar".
- **iconList**: A list of icons to be shown, when the sidebar is collapsed. The list is required and must match the amount of `sidebar_pane`.
- **...**: The individual `sidebar_pane`'s.

Value

A `shiny.tag` with individual sidebar panes

References


See Also

Other Sidebar Functions: `addSidebar()`, `closeSidebar()`, `openSidebar()`, `removeSidebar()`, `sidebar_pane()`

Examples

```r
## Not run:
library(shiny)

# run example app showing a single sidebar
runApp(paste0(system.file("examples", package = "leafletextras2"),
"/sidebar_app.R"))

# run example app showing two sidebars
runApp(paste0(system.file("examples", package = "leafletextras2"),
"/multi_sidebar_app.R"))

## End(Not run)
```

---

**startMoving**  
*Interact with the moving markers*

Description

The marker begins its path or resumes if it is paused.
Usage

\begin{verbatim}
startMoving(map, layerId = NULL)
stopMoving(map, layerId = NULL)
pauseMoving(map, layerId = NULL)
resumeMoving(map, layerId = NULL)
addLatLngMoving(map, layerId = NULL, latlng, duration)
moveToMoving(map, layerId = NULL, latlng, duration)
addStationMoving(map, layerId = NULL, pointIndex, duration)
\end{verbatim}

Arguments

- **map**: The leafletProxy object
- **layerId**: You can pass a string or a vector of strings for the moving markers that you want to address. If none is specified, the action will be applied to all moving markers.
- **latlng**: Coordinates as list (e.g.: list(33, -67) or list(lng=-65, lat=33))
- **duration**: Duration in milliseconds
- **pointIndex**: Index of a certain point

Value

the new map object

Functions

- **stopMoving**: Manually stops the marker, if you call start after, the marker starts again the polyline at the beginning.
- **pauseMoving**: Pauses the marker
- **resumeMoving**: The marker resumes its animation
- **addLatLngMoving**: Adds a point to the polyline. Useful, if we have to set the path one by one.
- **moveToMoving**: Stop the current animation and make the marker move to latlng in duration ms.
- **addStationMoving**: The marker will stop at the pointIndex point of the polyline for duration milliseconds. You can’t add a station at the first or last point of the polyline.

References

https://github.com/ewoken/Leaflet.MovingMarker

See Also

Other MovingMarker Functions: addMovingMarker(), movingMarkerOptions()
**Description**

A list of options for `addTimeslider`.

**Usage**

```r
timesliderOptions(
  position = c("topright", "bottomleft", "bottomright", "topleft"),
  timeAttribute = "time",
  isEpoch = FALSE,
  startTimeIdx = 0,
  timeStrLength = 19,
  maxValue = -1,
  minValue = 0,
  showAllOnStart = FALSE,
  range = FALSE,
  follow = FALSE,
  alwaysShowDate = FALSE,
  rezoom = NULL,
  sameDate = FALSE
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>position of control: &quot;topleft&quot;, &quot;topright&quot;, &quot;bottomleft&quot;, or &quot;bottomright&quot;. Default is <code>topright</code>.</td>
</tr>
<tr>
<td>timeAttribute</td>
<td>The column name of the time property. Default is &quot;time&quot;</td>
</tr>
<tr>
<td>isEpoch</td>
<td>whether the time attribute is seconds elapsed from epoch. Default is <code>FALSE</code></td>
</tr>
<tr>
<td>startTimeIdx</td>
<td>where to start looking for a timestring. Default is <code>0</code></td>
</tr>
<tr>
<td>timeStrLength</td>
<td>the size of <code>yyyy-mm-dd hh:mm:ss</code> - if milliseconds are present this will be larger. Default is <code>19</code></td>
</tr>
<tr>
<td>maxValue</td>
<td>Set the maximum value of the slider. Default is <code>-1</code></td>
</tr>
<tr>
<td>minValue</td>
<td>Set the minimum value of the slider. Default is <code>0</code></td>
</tr>
<tr>
<td>showAllOnStart</td>
<td>Specify whether all markers should be initially visible. Default is <code>FALSE</code></td>
</tr>
<tr>
<td>range</td>
<td>To use a range-slider, set to <code>TRUE</code>. Default is <code>FALSE</code>. Default is <code>FALSE</code></td>
</tr>
<tr>
<td>follow</td>
<td>To display only the markers at the specific timestamp specified by the slider. Specify a value of <code>1</code> (or <code>true</code>) to display only a single data point at a time, and a value of <code>null</code> (or <code>false</code>) to display the current marker and all previous markers. The range property overrides the follow property. Default is <code>FALSE</code></td>
</tr>
<tr>
<td>alwaysShowDate</td>
<td>Should the Date always be visible. Default is <code>FALSE</code></td>
</tr>
<tr>
<td>rezoom</td>
<td>Use the rezoom property to ensure the markers being displayed remain in view. Default is <code>NULL</code></td>
</tr>
<tr>
<td>sameDate</td>
<td>Show only data with the current selected time. Default is <code>FALSE</code></td>
</tr>
</tbody>
</table>
to_jsonformat

Value

A list of options for addTimeslider

References

https://github.com/dwilhelm89/LeafletSlider

See Also

Other Timeslider Functions: addTimeslider(), removeTimeslider()

Description

to_jsonformat Transform object to JSON expected format

Usage

to_jsonformat(data, time, popup = NULL, label = NULL)

Arguments

data          The data

time          Name of the time column.

popup         Name of the popup column.

label         Name of the label column.

Value

A list that is transformed to the expected JSON format
to_ms

Description

to_ms Change POSIX or Date to milliseconds

Usage

to_ms(data, time)

Arguments

data The data
time Name of the time column.

Value

A data.frame with the time column in milliseconds

unsync

Removes synchronization.

Description

Removes the synchronization of multiple maps from a specific map.

Usage

unsync(map, id = NULL, unsyncids = NULL)

Arguments

map the map
id The map id from which to unsynchronize the maps in unsyncids
unsyncids Unsynchronize the maps with the following IDs

Value

A map

See Also

Other leafletsync Functions: addLeafletsyncDependency(), addLeafletsync(), isSynced(), leafletsyncOptions()
updateHexbin

Description
Dynamically change the data and/or the colorRange.

Usage
updateHexbin(map, data = NULL, lng = NULL, lat = NULL, colorRange = NULL)

Arguments
- **map**: a map widget object created from `leaflet()`
- **data**: the data object from which the argument values are derived; by default, it is the data object provided to `leaflet()` initially, but can be overridden
- **lng**: a numeric vector of longitudes, or a one-sided formula of the form ~x where x is a variable in data; by default (if not explicitly provided), it will be automatically inferred from data by looking for a column named lng, long, or longitude (case-insensitively)
- **lat**: a vector of latitudes or a formula (similar to the lng argument; the names lat and latitude are used when guessing the latitude column from data)
- **colorRange**: The range of the color scale used to fill the hexbins

Value
the new map object

See Also
Other Hexbin-D3 Functions: `addHexbin()`, `clearHexbin()`, `hexbinOptions()`, `hideHexbin()`, `showHexbin()`

velocityOptions

Description
Define further options for the velocity layer.
Usage

velocityOptions(
  speedUnit = c("m/s", "k/h", "kt"),
  minVelocity = 0,
  maxVelocity = 10,
  velocityScale = 0.005,
  colorScale = NULL,
  ...
)

Arguments

  speedUnit Could be 'm/s' for meter per second, 'k/h' for kilometer per hour or 'kt' for knots
  minVelocity velocity at which particle intensity is minimum
  maxVelocity velocity at which particle intensity is maximum
  velocityScale scale for wind velocity
  colorScale A vector of hex colors or an RGB matrix
  ... Further arguments passed to the Velocity layer and Windy.js. For more information, please visit leaflet-velocity plugin

Value

A list of further options for addVelocity

See Also

Other Velocity Functions: addVelocity(), removeVelocity(), setOptionsVelocity()
See Also

Other Mapkey Functions: `addMapkeyMarkers()`, `makeMapkeyIcon()`, `mapkeyIconList()`, `mapkeyIcons()`
# Index

* **Antpath Functions**
  - addAntpath, 4
  - antpathOptions, 38
  - clearAntpath, 40
  - removeAntpath, 69

* **Arrowhead Functions**
  - addArrowhead, 6
  - arrowheadOptions, 39
  - clearArrowhead, 41
  - removeArrowhead, 69

* **Contextmenu Functions**
  - addContextmenu, 8
  - addItemContextmenu, 16
  - context_mapmenuItems, 44
  - context_markermenuItems, 44
  - context_menuItem, 45
  - hideContextmenu, 52
  - insertItemContextmenu, 55
  - mapmenuItems, 61
  - markermenuItems, 62
  - menuItem, 62
  - removeallItemsContextmenu, 68
  - removeItemContextmenu, 70
  - setDisabledContextmenu, 74
  - showContextmenu, 76

* **EasyPrint Functions**
  - addEasyprint, 9
  - easyprintMap, 46
  - easyprintOptions, 47
  - removeEasyprint, 70

* **GIBS Functions**
  - addGIBS, 10
  - setDate, 74
  - setTransparent, 76

* **Heightgraph Functions**
  - addHeightgraph, 11
  - heightgraphOptions, 50

* **Hexbin-D3 Functions**
  - addHexbin, 13
  - clearHexbin, 42
  - hexbinOptions, 51
  - hideHexbin, 53
  - showHexbin, 77
  - updateHexbin, 84

* **History Functions**
  - addHistory, 15
  - clearFuture, 41
  - clearHistory, 43
  - goBackHistory, 49
  - goForwardHistory, 49
  - historyOptions, 53

* **Mapkey Functions**
  - [.leaflet_mapkey_icon_set, 85
  - addMapkeyMarkers, 19
  - makeMapkeyIcon, 57
  - mapkeyIconList, 59
  - mapkeyIcons, 60

* **MovingMarker Functions**
  - addMovingMarker, 21
  - movingMarkerOptions, 63
  - startMoving, 79

* **Openweathermap Functions**
  - addOpenweatherCurrent, 22
  - addOpenweatherTiles, 24
  - openweatherCurrentOptions, 64
  - openweatherOptions, 65

* **Playback Functions**
  - addPlayback, 25
  - playbackOptions, 66
  - removePlayback, 71

* **Reachability Functions**
  - addReachability, 28
  - reachabilityOptions, 67
  - removeReachability, 71

* **Sidebar Functions**
  - addSidebar, 29
  - closeSidebar, 43
  - openSidebar, 64
addAntpath, 4, 39, 41, 69
addArrowhead, 6, 40, 41, 69
addContextmenu, 8, 16, 44, 45, 53, 55, 61–63, 68, 71, 75, 77
addEasyprint, 9, 46, 48, 70
addGeoJSON, 11
addGIBS, 10, 74, 76
addHeightgraph, 11, 51
addHexbin, 13, 42, 52, 53, 77, 84
addHistory, 15, 42, 43, 49, 50, 55
addItemContextmenu, 8, 16, 44, 45, 53, 55, 61–63, 68, 71, 75, 77
addLabelgun, 16
addLatLngMoving (startMoving), 79
addLayerscontrol, 5, 7, 10, 12, 14, 20, 21, 23, 24, 30, 32, 35, 37
addLeafletsync, 17, 19, 56, 57, 83
addLeafletsyncDependency, 18, 18, 56, 57, 83
addMapkeyMarkers, 19, 58, 59, 61, 86
addMovingMarker, 21, 63, 80
addOpenweatherCurrent, 22, 25, 65, 66
addOpenweatherTiles, 23, 24, 65, 66
addPlayback, 25, 66, 67, 71
addReachability, 28, 68, 72
addSidebar, 29, 44, 64, 72, 76, 79
addSidebyside, 30, 72, 73
addSpinner, 31
addStationMoving (startMoving), 79
addTangram, 32
addTimeslider, 33, 73, 81, 82
addVelocity, 35, 74, 75, 83
addWMS, 36
antpathOptions, 4, 5, 38, 41, 69
arrowheadOptions, 6, 7, 39, 41, 69
clearAntpath, 5, 39, 40, 69
clearArrowhead, 7, 40, 41, 69
clearFuture, 15, 41, 43, 49, 50, 55
clearGroup, 5, 7, 10, 12, 14, 20, 21, 23, 24, 32, 35, 37
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
clearHexbin, 14, 42, 52, 53, 77, 84
clearHistory, 15, 42, 43, 49, 50, 55
closeSidebar, 29, 43, 64, 72, 78, 79
context_mapmenuItems, 8, 16, 44, 45, 53, 55, 61–63, 68, 71, 75, 77
context_markermenuItems, 8, 16, 44, 45, 53, 55, 61–63, 68, 71, 75, 77
easyprintMap, 10, 46, 48, 70
easyprintOptions, 9, 10, 46, 47, 70
gibs_layers, 10, 48, 74, 76
goBackHistory, 15, 42, 43, 49, 50, 55
goForwardHistory, 15, 42, 43, 49, 50, 55
heightgraphOptions, 12, 13, 50
hexbinOptions, 14, 42, 51, 53, 77, 84
hideContextmenu, 8, 16, 44, 45, 52, 55, 61–63, 68, 71, 75, 77
hideHexbin, 14, 42, 52, 53, 77, 84
historyOptions, 15, 42, 43, 49, 50, 53
htmlEscape, 5, 7, 20, 22, 34
insertItemContextmenu, 8, 16, 44, 45, 53, 55, 61–63, 68, 71, 75, 77
INDEX

isSynced, 18, 19, 56, 57, 83
JS, 45, 52, 62

labelOptions, 5, 7, 20, 22, 26, 34
leaflet, 4, 6, 8–10, 12, 14–17, 23, 24, 31, 35, 37, 40, 52, 55, 68–70, 74–76, 84
leaflet.extras2, 56
leafletOptions, 8
leafletOutput, 29
leafletProxy, 40, 41, 43, 49, 69
leafletsyncOptions, 17–19, 56, 57, 83

makeIcon, 26
makeMapkeyIcon, 20, 57, 59, 61, 86
mapkeyIconList, 20, 58, 59, 61, 86
mapkeyIcons, 20, 58, 59, 60, 86
mapmenuItems, 8, 16, 44, 45, 53, 55, 61, 62, 63, 68, 71, 75, 77
markerClusterOptions, 20
markermenuItems, 8, 16, 44, 45, 53, 55, 61, 62, 63, 68, 71, 75, 77
markerOptions, 8, 20, 22
menuItem, 8, 16, 44, 45, 53, 55, 61, 62, 63, 68, 71, 75, 77
moveToMoving (startMoving), 79
movingMarkerOptions, 22, 63, 80

openSidebar, 29, 44, 64, 72, 78, 79
openweatherCurrentOptions, 23, 25, 64, 66
openweatherOptions, 23–25, 65, 66

pathOptions, 8, 12, 26
pauseMoving (startMoving), 79
playbackOptions, 26, 66, 71
popupOptions, 5, 7, 20, 22, 26, 34, 37

reachabilityOptions, 28, 67, 72
removeAllItemsContextmenu, 8, 16, 44, 45, 53, 55, 61–63, 68, 71, 75, 77
removeAntpath, 5, 39, 41, 69
removeArrowhead, 7, 40, 41, 69
removeControl, 12
removeEasyprint, 10, 46, 48, 70
removeItemContextmenu, 8, 16, 44, 45, 53, 55, 61–63, 68, 70, 75, 77
removePlayback, 26, 67, 71
removeReachability, 28, 68, 71
removeSidebar, 29, 44, 64, 72, 78, 79
removeSidebyside, 30, 72

removeTimeslider, 34, 73, 82
removeVelocity, 36, 73, 75, 85
resumeMoving (startMoving), 79

setDate, 11, 74, 76
setDisabledContextmenu, 8, 16, 44, 45, 53, 55, 61–63, 68, 71, 74, 77
setOptionsVelocity, 36, 74, 75, 85
setTransparent, 11, 74, 76
showContextmenu, 8, 16, 44, 45, 53, 55, 61–63, 68, 71, 75, 76
showHexbin, 14, 42, 52, 53, 77, 84
sidebarPane, 29, 44, 64, 72, 77, 79
sidebar_tabs, 29, 43, 44, 64, 72, 78, 79
startMoving, 22, 63, 79
startSpinner (addSpinner), 31
stopMoving (startMoving), 79
stopSpinner (addSpinner), 31
timesliderOptions, 34, 73, 81
to_jsonformat, 82
to_ms, 83

unsubscribe, 18, 19, 56, 57, 83
updateHexbin, 14, 42, 52, 53, 77, 84

velocityOptions, 36, 74, 75, 84