

# Package ‘leaflet.extras2’

August 21, 2023

**Type** Package

**Title** Extra Functionality for 'leaflet' Package

**Version** 1.2.2

**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, curl

**URL** <https://trafficonese.github.io/leaflet.extras2/>,  
<https://github.com/trafficonese/leaflet.extras2>

**BugReports** <https://github.com/trafficonese/leaflet.extras2/issues>

**RoxygenNote** 7.2.0

**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](mailto:sebastian_gatscha@gmx.at)>

**Repository** CRAN

**Date/Publication** 2023-08-21 09:22:43 UTC

## R topics documented:

addAntpath . . . . .	4
addArrowhead . . . . .	6
addContextmenu . . . . .	8

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 . . . . .	42
clearHexbin . . . . .	42
clearHistory . . . . .	43
closeSidebar . . . . .	43
context_mapmenuItems . . . . .	44
context_markermenuItems . . . . .	45
context_menuItem . . . . .	45
disableContextmenu . . . . .	46
easyprintMap . . . . .	47
easyprintOptions . . . . .	48
enableContextmenu . . . . .	49
gibs_layers . . . . .	50
goBackHistory . . . . .	50
goForwardHistory . . . . .	51
heightgraphOptions . . . . .	51
hexbinOptions . . . . .	53
hideContextmenu . . . . .	54
hideHexbin . . . . .	55
historyOptions . . . . .	55
insertItemContextmenu . . . . .	57
isSynced . . . . .	57
leaflet.extras2 . . . . .	58

leafletSyncOptions . . . . .	58
makeMapkeyIcon . . . . .	59
mapkeyIconList . . . . .	60
mapkeyIcons . . . . .	61
mapmenuItem . . . . .	62
markermenuItem . . . . .	63
menuItem . . . . .	64
movingMarkerOptions . . . . .	64
openSidebar . . . . .	65
openweatherCurrentOptions . . . . .	66
openweatherOptions . . . . .	66
playbackOptions . . . . .	67
reachabilityOptions . . . . .	69
removeallItemsContextmenu . . . . .	69
removeAntpath . . . . .	70
removeArrowhead . . . . .	71
removeEasyprint . . . . .	71
removeItemContextmenu . . . . .	72
removePlayback . . . . .	72
removeReachability . . . . .	73
removeSidebar . . . . .	73
removeSidebyside . . . . .	74
removeTimeslider . . . . .	74
removeVelocity . . . . .	75
setDate . . . . .	75
setDisabledContextmenu . . . . .	76
setOptionsVelocity . . . . .	77
setTransparent . . . . .	77
showContextmenu . . . . .	78
showHexbin . . . . .	79
sidebar_pane . . . . .	79
sidebar_tabs . . . . .	80
startMoving . . . . .	81
timesliderOptions . . . . .	82
to_jsonformat . . . . .	84
to_ms . . . . .	84
unsync . . . . .	85
updateHexbin . . . . .	85
velocityOptions . . . . .	86
[.leaflet_mapkey_icon_set . . . . .	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

<code>map</code>	a map widget object created from <code>leaflet()</code>
<code>lng</code>	a numeric vector of longitudes, or a one-sided formula of the form <code>~x</code> where <code>x</code> is a variable in <code>data</code> ; by default (if not explicitly provided), it will be automatically inferred from <code>data</code> by looking for a column named <code>lng</code> , <code>long</code> , or <code>longitude</code> (case-insensitively)
<code>lat</code>	a vector of latitudes or a formula (similar to the <code>lng</code> argument; the names <code>lat</code> and <code>latitude</code> are used when guessing the latitude column from <code>data</code> )
<code>layerId</code>	the layer id

group	the name of the group the newly created layers should belong to (for <code>clearGroup</code> and <code>addLayersControl</code> 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 <b>dash pattern</b>
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 <code>htmlEscape()</code> for security reasons)
popupOptions	A Vector of <code>popupOptions</code> to provide popups
label	a character vector of the HTML content for the labels
labelOptions	A Vector of <code>labelOptions</code> to provide label options for each label. Default NULL
options	A named list of options. See <code>antpathOptions</code>
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 <code>leaflet()</code> 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**

```
library(leaflet)
leaflet() %>%
  addAntpath(data = atlStorms2005)
```

---

`addArrowhead`*Add Lines with an arrowhead*

---

### Description

Can be used almost exactly like `addPolyLines` but instead of `pathOptions` you can use `arrowheadOptions`. See [leaflet-arrowheads](#) for further details.

### Usage

```
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

<code>map</code>	a map widget object created from <code>leaflet()</code>
<code>lng</code>	a numeric vector of longitudes, or a one-sided formula of the form <code>~x</code> where <code>x</code> is a variable in <code>data</code> ; by default (if not explicitly provided), it will be automatically inferred from <code>data</code> by looking for a column named <code>lng</code> , <code>long</code> , or <code>longitude</code> (case-insensitively)
<code>lat</code>	a vector of latitudes or a formula (similar to the <code>lng</code> argument; the names <code>lat</code> and <code>latitude</code> are used when guessing the latitude column from <code>data</code> )
<code>layerId</code>	the layer id

group	the name of the group the newly created layers should belong to (for <code>clearGroup</code> and <code>addLayersControl</code> 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 <b>dash pattern</b>
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 <code>htmlEscape()</code> for security reasons)
popupOptions	A Vector of <code>popupOptions</code> to provide popups
label	a character vector of the HTML content for the labels
labelOptions	A Vector of <code>labelOptions</code> to provide label options for each label. Default NULL
options	A named list of options. See <code>arrowheadOptions</code>
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 <code>leaflet()</code> initially, but can be overridden

**Value**

A modified leaflet map with a polyline with arrowheads

**References**

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

**See Also**

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

**Examples**

```
library(leaflet)
leaflet() %>%
  addArrowhead(data = at1Storms2005)
```

---

addContextmenu	<i>Add contextmenu Plugin</i>
----------------	-------------------------------

---

## 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\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)



**Examples**

```

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**

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

**Arguments**

map	a map widget object created from <a href="#">leaflet</a>
options	A named list of options. See <a href="#">easyprintOptions</a>

**Value**

A leaflet map object

**References**

<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 <a href="#">leaflet()</a>
layers	A character vector of GIBS-layers. See <a href="#">gibs_layers</a>
group	the name of the group the newly created layers should belong to (for <a href="#">clearGroup</a> and <a href="#">addLayersControl</a> 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,
```

```

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

```

### Arguments

map	a map widget object created from <a href="#">leaflet()</a>
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 <a href="#">clearGroup</a> and <a href="#">addLayersControl</a> 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 <b>dash pattern</b>
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 <a href="#">pathOptions</a>
options	List of further plugin options. See <a href="#">heightgraphOptions</a>

### 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

<https://github.com/GIScience/Leaflet.Heightgraph>

**See Also**

Other Heightgraph Functions: [heightgraphOptions\(\)](#)

**Examples**

```
## 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 <- st_as_sf(st_sfc(lapply(split(data, L1), function(x) {
  st_linestring(as.matrix(x))
})))
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**

```
addHexbin(
  map,
  lng = NULL,
  lat = NULL,
  radius = 20,
```

```

    layerId = NULL,
    group = NULL,
    opacity = 0.5,
    options = hexbinOptions(),
    data = getMapData(map)
  )

```

### Arguments

map	a map widget object created from <code>leaflet()</code>
lng	a numeric vector of longitudes, or a one-sided formula of the form <code>~x</code> where <code>x</code> is a variable in <code>data</code> ; by default (if not explicitly provided), it will be automatically inferred from <code>data</code> by looking for a column named <code>lng</code> , <code>long</code> , or <code>longitude</code> (case-insensitively)
lat	a vector of latitudes or a formula (similar to the <code>lng</code> argument; the names <code>lat</code> and <code>latitude</code> are used when guessing the latitude column from <code>data</code> )
radius	Radius of the hexbin layer
layerId	the layer id
group	the name of the group the newly created layers should belong to (for <code>clearGroup</code> and <code>addLayersControl</code> 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 <code>hexbinOptions</code>
data	the data object from which the argument values are derived; by default, it is the data object provided to <code>leaflet()</code> initially, but can be overridden

### Value

the new map object

### Note

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

### References

<https://github.com/bluehalo/leaflet-d3#hexbins-api>

### See Also

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

## 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 <a href="#">leaflet</a>
layerId	the control id
options	A named list of options. See <a href="#">historyOptions</a>

## Value

the new map object

## References

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

## See Also

Other History Functions: [clearFuture\(\)](#), [clearHistory\(\)](#), [goBackHistory\(\)](#), [goForwardHistory\(\)](#), [historyOptions\(\)](#)

**Examples**

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

---

addItemContextmenu      *addItemContextmenu*

---

**Description**

Add a new contextmenu menu item

**Usage**

```
addItemContextmenu(map, option)
```

**Arguments**

map	a map widget object created from <a href="#">leaflet</a>
option	new menu item to add

**Value**

A leaflet map object

**See Also**

Other Contextmenu Functions: [addContextmenu\(\)](#), [context\\_mapmenuItems\(\)](#), [context\\_markermenuItems\(\)](#), [context\\_menuItem\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [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**

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



**Arguments**

map	A map widget object created from <a href="#">leaflet</a>
group	The group name of the layer/s for which label collisions are to be avoided. To see the effects of this plugin the <code>labelOptions</code> of the markers must be configured with either <code>permanent = TRUE</code> or <code>noHide = TRUE</code> .
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.
entries	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	<i>Synchronize multiple Leaflet map</i>
----------------	---

---

**Description**

The plugin allows you to synchronize and unsynchronize multiple leaflet maps in a Shiny application. You can pass additional options to [leafletSyncOptions](#). For more information see [Leaflet.Sync](#)

**Usage**

```
addLeafletsync(  
  map,  
  ids = NULL,  
  syncList = "all",  
  options = leafletsyncOptions()  
)
```

**Arguments**

map	the map
ids	the map ids to be synced. If you use a <code>synclist</code> , you may leave it <code>NULL</code> . The unique names and values of <code>synclist</code> will be used.
synclist	The synchronization list. The default is 'all', which creates a list of all possible combinations of <code>ids</code> . For a more detailed control, a named list can be passed in this form <code>list(m1 = c("m2", "m3"), m2 = c("m1", "m3"), m3 = c("m1", "m2"))</code> , 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 <a href="#">leafletsyncOptions</a> . 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 <code>./inst/examples/offset_continuous.R</code>

**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\(\)](#)

---

addLeafletsyncDependency

*Add the Leaflet Sync JS dependencies*

---

**Description**

Sometimes it makes sense to include the Leaflet Sync dependencies already before synchronizing maps. For example, if you want to use the 'L.Sync.offsetHelper'. See the example in `./inst/examples/offsetHelper.R`

**Usage**

```
addLeafletsyncDependency(map)
```

**Arguments**

map	the map
-----	---------

**Value**

A modified leaflet map

**See Also**

Other leafletsync Functions: [addLeafletsync\(\)](#), [isSynced\(\)](#), [leafletsyncOptions\(\)](#), [unsync\(\)](#)

---

addMapkeyMarkers	<i>Add Mapkey Markers</i>
------------------	---------------------------

---

**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 $\sim x$ 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

group	the name of the group the newly created layers should belong to (for <code>clearGroup</code> and <code>addLayersControl</code> 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 <code>htmlEscape()</code> for security reasons)
popupOptions	A Vector of <code>popupOptions</code> to provide popups
label	a character vector of the HTML content for the labels
labelOptions	A Vector of <code>labelOptions</code> to provide label options for each label. Default NULL
options	a list of extra options for markers. See <code>markerOptions</code>
clusterOptions	if not NULL, markers will be clustered using <code>Leaflet.markercluster</code> ; you can use <code>markerClusterOptions()</code> 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 <code>leaflet()</code> 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**

```
library(leaflet)

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

---

addMovingMarker	<i>Add Moving Markers</i>
-----------------	---------------------------

---

## 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

```
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 $\sim x$ 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 <a href="#">clearGroup</a> and <a href="#">addLayersControl</a> 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 <code>htmlEscape()</code> for security reasons)
popupOptions	A Vector of <code>popupOptions</code> to provide popups
label	a character vector of the HTML content for the labels
labelOptions	A Vector of <code>labelOptions</code> to provide label options for each label. Default NULL
movingOptions	a list of extra options for moving markers. See <code>movingMarkerOptions</code>
options	a list of extra options for markers. See <code>markerOptions</code>
data	the data object from which the argument values are derived; by default, it is the data object provided to <code>leaflet()</code> initially, but can be overridden

**Value**

the new map object

**References**

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

**See Also**

Other MovingMarker Functions: `movingMarkerOptions()`, `startMoving()`

**Examples**

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

df <- sf::st_as_sf(atl1storms2005)[1,]
leaflet() %>%
  addTiles() %>%
  addPolylines(data = df) %>%
  addMovingMarker(data = df,
                  movingOptions = movingMarkerOptions(autostart = TRUE, loop = TRUE),
                  label="I am a pirate!",
                  popup="Arrr")
```

---

addOpenweatherCurrent *Add current OpenWeatherMap Marker*

---

**Description**

Add current OpenWeatherMap Marker

## Usage

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

## Arguments

map	a map widget object created from <a href="#">leaflet()</a>
apikey	a valid Openweathermap-API key. Get one from <a href="#">here</a> .
group	the name of the group the newly created layers should belong to (for <a href="#">clearGroup</a> and <a href="#">addLayersControl</a> 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 <a href="#">openweatherCurrentOptions</a>

## 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

```

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

```

## Arguments

map	a map widget object created from <a href="#">leaflet()</a>
apikey	a valid OpenWeatherMap-API key. Get one from <a href="#">here</a> .
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 <a href="#">clearGroup</a> and <a href="#">addLayersControl</a> 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 <a href="#">openweatherOptions</a>

## 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**

```
## 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**

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

```

  popupOptions = NULL,
  labelOptions = NULL,
  options = playbackOptions(),
  name = NULL
)

```

### 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 <a href="#">makeIcon</a>
pathOpts	style the CircleMarkers with <a href="#">pathOptions</a>
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 <a href="#">popupOptions</a> to provide popups
labelOptions	A Vector of <a href="#">labelOptions</a> to provide label options for each label. Default NULL
options	List of additional options. See <a href="#">playbackOptions</a>
name	A formula with the column names for the feature name

### 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>

### See Also

Other Playback Functions: [playbackOptions\(\)](#), [removePlayback\(\)](#)

### Examples

```

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

```

```

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

leaflet() %>%
  addTiles() %>%
  addPlayback(data = data, label = ~label,
    popup = ~sprintf("I am a popup for <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 <a href="#">Openrouteservice</a>
options	A list of further options. See <a href="#">reachabilityOptions</a>

### 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

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

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

leaflet() %>%
  addTiles() %>%
```

```

setView(8, 50, 10) %>%
addReachability()

## End(Not run)

```

---

addSidebar *Add a Sidebar Leaflet Control*

---

### Description

The sidebar HTML must be created with `sidebar_tabs` and `sidebar_pane` before `leafletOutput` is called.

### Usage

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

### Arguments

<code>map</code>	A leaflet map widget
<code>id</code>	Id of the sidebar-div. Must match with the id of <code>sidebar_tabs</code>
<code>options</code>	A named list with the only option <code>position</code> , which should be either left or right.
<code>ns</code>	The namespace function, if used in Shiny modules.

### 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

```

## Not run:
library(shiny)

# run example app showing a single sidebar
runApp(paste0(system.file("examples", package = "leaflet.extras2"),
"/sidebar_app.R"))

# run example app showing two sidebars

```

```
runApp(paste0(system.file("examples", package = "leaflet.extras2"),
  "/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

```
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 <a href="#">removeSidebyside</a>
leftId	the layerId of the Tile layer that should be visible on the <b>left</b> side
rightId	the layerId of the Tile layer that should be visible on the <b>right</b> 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>

### See Also

Other Sidebyside Functions: [removeSidebyside\(\)](#)

**Examples**

```

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")

```

---

 addSpinner

*Add Spin Plugin*


---

**Description**

Adds an animated loading spinning over the map.

**Usage**

```

addSpinner(map)

startSpinner(map, options = NULL)

stopSpinner(map)

```

**Arguments**

map	A map widget object created from <a href="#">leaflet</a>
options	Spin.js options. Named list. See <a href="http://spin.js.org">http://spin.js.org</a>

**Value**

A leaflet map object

**References**

<https://github.com/makinacorp/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	<i>Adds a Tangram layer to a Leaflet map in a Shiny App.</i>
------------	--

---

**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 <b>.yaml</b> or <b>.zip</b> 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 <a href="#">Tangram repository</a> or the <a href="#">Tangram docs</a> 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 <a href="#">clearGroup</a> and <a href="#">addLayersControl</a> purposes).
options	A list of further options. See the app in the examples/tangram folder or the <a href="#">docs</a> for further information.

**Value**

the new map object

**Note**

Only works correctly in a Shiny-App environment.

**References**

<https://github.com/tangrams/tangram>



**Examples**

```
## 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**

```
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 <b>dash pattern</b>
popup	a character vector of the HTML content for the popups (you are recommended to escape the text using <code>htmlEscape()</code> for security reasons)
popupOptions	A Vector of <code>popupOptions</code> to provide popups
label	a character vector of the HTML content for the labels
labelOptions	A Vector of <code>labelOptions</code> 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 <code>timesliderOptions</code>

### Value

the new map object

### References

<https://github.com/dwilhelm89/LeafletSlider>

### See Also

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

**Examples**

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

data <- sf::st_as_sf(leaflet::at1Storms2005[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**

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

**Arguments**

map	a map widget object created from <a href="#">leaflet()</a>
layerId	the layer id
group	the name of the group the newly created layers should belong to (for <a href="#">clearGroup</a> and <a href="#">addLayersControl</a> 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.

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 <a href="#">demo files</a> for some example data.
options	List of further options. See <a href="#">velocityOptions</a>

**Value**

the new map object

**References**

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

**See Also**

Other Velocity Functions: [removeVelocity\(\)](#), [setOptionsVelocity\(\)](#), [velocityOptions\(\)](#)

**Examples**

```
## Not run:
library(leaflet)
library(leaflet.extras2)
content <- "https://raw.githubusercontent.com/onaci/leaflet-velocity/master/demo/water-gbr.json"
leaflet() %>%
  addTiles(group = "base") %>%
  setView(145, -20, 4) %>%
  addVelocity(content = content, group = "velo", layerId = "veloid") %>%
  addLayersControl(baseGroups = "base", overlayGroups = "velo")

## End(Not run)
```

---

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.

You can also use **CQL-Filters** by appending a string to the 'baseUrl'.

Something like 'http://server/wms?qql\_filter=attribute=value'

**Usage**

```
addWMS(
  map,
  baseUrl,
  layerId = NULL,
  group = NULL,
```

```

    options = WMSTileOptions(),
    attribution = NULL,
    layers = NULL,
    popupOptions = NULL,
    checkempty = FALSE,
    data = getMapData(map)
)

```

### Arguments

map	a map widget object created from <code>leaflet()</code>
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 <code>clearGroup</code> and <code>addLayersControl</code> 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 <code>popupOptions</code> . 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 <code>leaflet()</code> initially, but can be overridden

### Value

the new map object

### References

<https://github.com/heigeo/leaflet.wms>

### Examples

```

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_winddaten_10m",
        popupOptions = popupOptions(maxWidth = 600),

```

```

checkempty = TRUE,
options = WMSTileOptions(
  transparent = TRUE,
  format = "image/png",
  info_format = "text/html"))

```

---

antpathOptions      *Antpath Options*

---

### 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 <b>shape to be used at the end</b> of the stroke
lineJoin	a string that defines <b>shape to be used at the corners</b> 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: [addAntpath\(\)](#), [clearAntpath\(\)](#), [removeAntpath\(\)](#)

---

arrowheadOptions	<i>Arrowhead Options</i>
------------------	--------------------------

---

**Description**

Additional list of options for polylines with arrowheads. You can also pass options inherited from [L.Path](#)

**Usage**

```
arrowheadOptions(
  yawn = 60,
  size = "15%",
  frequency = "allvertices",
  proportionalToTotal = FALSE,
  offsets = NULL,
  perArrowheadOptions = NULL,
  ...
)
```

**Arguments**

yawn	Defines the width of the opening of the arrowhead, given in degrees. The larger the angle, the wider the arrowhead.
size	Determines the size of the arrowhead. Accepts three types of values: <ul style="list-style-type: none"> <li>• A string with the suffix 'm', i.e. '500m' will set the size of the arrowhead to that number of meters.</li> <li>• 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.</li> <li>• 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.</li> </ul>
frequency	How many arrowheads are rendered on a polyline. <ul style="list-style-type: none"> <li>• 'allvertices' renders an arrowhead on each vertex.</li> <li>• 'endonly' renders only one at the end.</li> </ul>

- A numeric value renders that number of arrowheads evenly spaced along the polyline.
- A string with suffix 'm', i.e. '100m' will render arrowheads spaced evenly along the polyline with roughly that many meters between each one.
- 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

*clearAntpath*

---

**Description**

Clear all Antpaths

**Usage**

`clearAntpath(map)`



**Arguments**

map            a map widget object, possibly created from [leaflet\(\)](#) but more likely from [leafletProxy\(\)](#)

**Value**

the new map object

**See Also**

Other Antpath Functions: [addAntpath\(\)](#), [antpathOptions\(\)](#), [removeAntpath\(\)](#)

---

clearArrowhead            *Remove arrowheads from Lines by group*

---

**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 Arrowhead Functions: [addArrowhead\(\)](#), [arrowheadOptions\(\)](#), [removeArrowhead\(\)](#)

---

clearFuture	<i>clearFuture</i>
-------------	--------------------

---

**Description**

Resets the stack of future items.

**Usage**

```
clearFuture(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\(\)](#), [clearHistory\(\)](#), [goBackHistory\(\)](#), [goForwardHistory\(\)](#), [historyOptions\(\)](#)

---

clearHexbin	<i>clearHexbin</i>
-------------	--------------------

---

**Description**

Clears the data of the hexbinLayer.

**Usage**

```
clearHexbin(map)
```

**Arguments**

map                    The map widget

**Value**

the new map object

**See Also**

Other Hexbin-D3 Functions: [addHexbin\(\)](#), [hexbinOptions\(\)](#), [hideHexbin\(\)](#), [showHexbin\(\)](#), [updateHexbin\(\)](#)

---

<code>clearHistory</code>	<i>clearHistory</i>
---------------------------	---------------------

---

**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\(\)](#)

---

<code>closeSidebar</code>	<i>Close the Sidebar</i>
---------------------------	--------------------------

---

**Description**

Close the Sidebar

**Usage**

```
closeSidebar(map, sidebar_id = NULL)
```

**Arguments**

map	A leaflet map widget
sidebar_id	The id of the sidebar (per <a href="#">sidebar_tabs</a> ). Defaults to NULL such that the first sidebar is used.

**Value**

the new map object

**See Also**

Other Sidebar Functions: [addSidebar\(\)](#), [openSidebar\(\)](#), [removeSidebar\(\)](#), [sidebar\\_pane\(\)](#), [sidebar\\_tabs\(\)](#)

---

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: [addContextmenu\(\)](#), [addItemContextmenu\(\)](#), [context\\_markermenuItems\(\)](#), [context\\_menuItem\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

context\_markermenuItem  
*context\_markermenuItem*

---

**Description**

context\_markermenuItem

**Usage**

context\_markermenuItem(...)

**Arguments**

... contextmenu item/s

**Value**

A list of context\_menuItem for markers

**See Also**

Other Contextmenu Functions: [addContextmenu\(\)](#), [addItemContextmenu\(\)](#), [context\\_mapmenuItem\(\)](#), [context\\_menuItem\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItem\(\)](#), [markermenuItem\(\)](#), [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 <a href="https://github.com/araticliffe/Leaflet.contextmenu">https://github.com/araticliffe/Leaflet.contextmenu</a>

**Value**

A contextmenu item list

**See Also**

Other Contextmenu Functions: [addContextmenu\(\)](#), [addItemContextmenu\(\)](#), [context\\_mapmenuItems\(\)](#), [context\\_markermenuItems\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

disableContextmenu      *disableContextmenu*

---

**Description**

Disable the contextmenu

**Usage**

```
disableContextmenu(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\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

easyprintMap	<i>easyprintMap</i>
--------------	---------------------

---

## 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 <a href="#">easyprintOptions</a> . 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) %>%
    })
  }
}
```

```

    addEasyprint(options = easyprintOptions(
      exportOnly = TRUE
    ))
  })
  observeEvent(input$print, {
    leafletProxy("map") %>%
      easyprintMap(sizeModes = input$scene)
  })
}

shinyApp(ui, server)
}

```

---

easyprintOptions	<i>easyprintOptions</i>
------------------	-------------------------

---

## 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 <code>./inst/examples/easyprint_app.R</code> .



defaultSizeTitles	Button tooltips for the default page sizes
exportOnly	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\(\)](#)

---

enableContextmenu	<i>enableContextmenu</i>
-------------------	--------------------------

---

**Description**

Enable the contextmenu

**Usage**

```
enableContextmenu(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\(\)](#), [disableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

<code>gibs_layers</code>	<i>The available GIBS layers with attributes</i>
--------------------------	--

---

**Description**

The available GIBS layers with attributes

**Usage**

```
gibs_layers
```

**Format**

An object of class `data.frame` with 276 rows and 4 columns.

---

<code>goBackHistory</code>	<i>goBackHistory</i>
----------------------------	----------------------

---

**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\(\)](#)

---

goForwardHistory      *goForwardHistory*

---

**Description**

If possible, will go to next map extent. Pushes current extent to the "back" stack.

**Usage**

```
goForwardHistory(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\(\)](#), [goBackHistory\(\)](#), [historyOptions\(\)](#)

---

heightgraphOptions      *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 <code>list(top = 10, right = 30, bottom = 55, left = 50)</code> .
expand	Boolean value that defines if the heightgraph should be expanded on creation. Default is <code>TRUE</code> .
expandCallback	Function to be called if the heightgraph is expanded or reduced. The state of the heightgraph is passed as an argument. It is <code>TRUE</code> when expanded and <code>FALSE</code> when reduced. Default is <code>NULL</code> .
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 <code>FeatureCollection</code> . Default is <code>NULL</code> .
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 <code>highlightStyle</code> parameter. Default is <code>list(color = "red")</code> .
translation	You can change the labels of the heightgraph info field by passing translations for distance, elevation, segment_length, type and legend. Default is <code>NULL</code> .
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\(\)](#)

---

 hexbinOptions

*hexbinOptions*


---

**Description**

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

**Usage**

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

**Arguments**

<code>duration</code>	Transition duration for the hexbin layer
<code>colorScaleExtent</code>	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 <code>c(min= numeric, max= numeric)</code> . Can be a numeric vector or a custom JS array, like <code>(JS("[40, undefined]"))</code>
<code>radiusScaleExtent</code>	This is the same exact configuration option as <code>colorScaleExtent</code> , only applied to the radius extent.
<code>colorRange</code>	Sets the range of the color scale used to fill the hexbins on the layer.
<code>radiusRange</code>	Sets the range of the radius scale used to size the hexbins on the layer.
<code>pointerEvents</code>	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.

<code>resizetoCount</code>	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 <a href="#">JS</a> function.
<code>tooltip</code>	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 <a href="#">JS</a> function.

**Value**

A list of hexbin-specific options

**See Also**

Other Hexbin-D3 Functions: [addHexbin\(\)](#), [clearHexbin\(\)](#), [hideHexbin\(\)](#), [showHexbin\(\)](#), [updateHexbin\(\)](#)

---

<code>hideContextmenu</code>	<i>hideContextmenu</i>
------------------------------	------------------------

---

**Description**

Hide the contextmenu

**Usage**

```
hideContextmenu(map)
```

**Arguments**

<code>map</code>	a map widget object created from <a href="#">leaflet</a>
------------------	--

**Value**

A leaflet map object

**See Also**

Other Contextmenu Functions: [addContextmenu\(\)](#), [addItemContextmenu\(\)](#), [context\\_mapmenuItems\(\)](#), [context\\_markermenuItems\(\)](#), [context\\_menuItem\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

hideHexbin	<i>hideHexbin</i>
------------	-------------------

---

**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	<i>History Options</i>
----------------	------------------------

---

**Description**

History Options

**Usage**

```
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>

**See Also**

Other History Functions: [addHistory\(\)](#), [clearFuture\(\)](#), [clearHistory\(\)](#), [goBackHistory\(\)](#), [goForwardHistory\(\)](#)

**Examples**

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



---

 insertItemContextmenu *insertItemContextmenu*


---

**Description**

Insert a new contextmenu menu item at a specific index

**Usage**

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

**Arguments**

map	a map widget object created from <a href="#">leaflet</a>
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\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [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 Siny 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\(\)](#)

---

`leaflet.extras2`      *leaflet.extras2: Extra Functionality for 'leaflet' Package.*

---

**Description**

This package serves as an add-on to the 'leaflet' package by providing extra functionality via 'leaflet' plugins.

---

`leafletSyncOptions`      *leafletSync Options*

---

**Description**

Additional list of options.

**Usage**

```
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\(\)](#)

---

makeMapkeyIcon	<i>Make Mapkey Icon</i>
----------------	-------------------------

---

**Description**

Make Mapkey Icon

**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.
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;'.
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>

**See Also**

Other Mapkey Functions: [[.leaflet\\_mapkey\\_icon\\_set\(\)](#), [addMapkeyMarkers\(\)](#), [mapkeyIconList\(\)](#), [mapkeyIcons\(\)](#)]

**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**

```
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.
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;'
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>

**See Also**

Other Mapkey Functions: [[.leaflet\\_mapkey\\_icon\\_set\(\)](#), [addMapkeyMarkers\(\)](#), [makeMapkeyIcon\(\)](#), [mapkeyIconList\(\)](#)]

**Examples**

```
## Not run:
library(leaflet)
leaflet() %>%
  addMapkeyMarkers(data = breweries91,
                  icon = mapkeyIcons(
                    color = "red",
                    borderRadius = 0,
                    iconSize = 25))

## End(Not run)
```

---

mapmenuItems

*mapmenuItems*

---

**Description**

mapmenuItems

**Usage**

mapmenuItems(...)

**Arguments**

... contextmenu item/s

**Value**

A list of menuItem for the map

**See Also**

Other Contextmenu Functions: [addContextmenu\(\)](#), [addItemContextmenu\(\)](#), [context\\_mapmenuItems\(\)](#), [context\\_markermenuItems\(\)](#), [context\\_menuItem\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

markermenuItems

*markermenuItems*

---

**Description**

markermenuItems

**Usage**

markermenuItems(...)

**Arguments**

... contextmenu item/s

**Value**

A list of menuItem for markers

**See Also**

Other Contextmenu Functions: [addContextmenu\(\)](#), [addItemContextmenu\(\)](#), [context\\_mapmenuItems\(\)](#), [context\\_markermenuItems\(\)](#), [context\\_menuItem\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

menuItem	<i>menuItem</i>
----------	-----------------

---

**Description**

menuItem

**Usage**

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 <a href="https://github.com/aratcliffe/Leaflet.contextmenu">https://github.com/aratcliffe/Leaflet.contextmenu</a>

**Value**

A contextmenu item list

**See Also**

Other Contextmenu Functions: [addContextmenu\(\)](#), [addItemContextmenu\(\)](#), [context\\_mapmenuItems\(\)](#), [context\\_markermenuItems\(\)](#), [context\\_menuItem\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

movingMarkerOptions	<i>Set options for Moving Markers</i>
---------------------	---------------------------------------

---

**Description**

Set options for Moving Markers

**Usage**

movingMarkerOptions(autostart = FALSE, loop = FALSE, pauseOnZoom = FALSE)



**Arguments**

autostart	If TRUE the marker will start automatically after it is added to map. Default is FALSE
loop	if TRUE the marker will start automatically at the beginning of the polyline when the it arrives at the end. Default is FALSE
pauseOnZoom	Pause the marker while zooming. While this improves the animation, it is not recommended because the animation time is lost and the marker will not appear at the correct time at the next station. Default is FALSE

**Value**

A list of extra options for moving markers

**References**

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

**See Also**

Other MovingMarker Functions: [addMovingMarker\(\)](#), [startMoving\(\)](#)

---

openSidebar	<i>Open the Sidebar by ID</i>
-------------	-------------------------------

---

**Description**

Open the Sidebar by ID

**Usage**

```
openSidebar(map, id, sidebar_id = NULL, ns = NULL)
```

**Arguments**

map	A leaflet map widget
id	The id of the <a href="#">sidebar_pane</a> to open.
sidebar_id	The id of the sidebar (per <a href="#">sidebar_tabs</a> ). Defaults to NULL such that the first sidebar is used.
ns	The namespace function, if used in Shiny modules.

**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, ...)
```

**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 <a href="#">full list of options</a>

**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](#)

---

playbackOptions	<i>playbackOptions</i>
-----------------	------------------------

---

**Description**

A list of options for [addPlayback](#). For a full list please visit the [plugin repository](#).

**Usage**

```
playbackOptions(
  color = "blue",
  radius = 5,
  tickLen = 250,
  speed = 50,
  maxInterpolationTime = 5 * 60 * 1000,
  tracksLayer = TRUE,
  playControl = TRUE,
  dateControl = TRUE,
  sliderControl = TRUE,
  orientIcons = FALSE,
  staleTime = 60 * 60 * 1000,
  transitionpopup = TRUE,
  transitionlabel = TRUE,
  ...
)
```

**Arguments**

color	colors of the CircleMarkers.
radius	a numeric value for the radius of the CircleMarkers.
tickLen	Set tick length in milliseconds. Increasing this value, may improve performance, at the cost of animation smoothness. Default is 250
speed	Set float multiplier for default animation speed. Default is 50
maxInterpolationTime	Set max interpolation time in seconds. Default is 5*60*1000 (5 minutes).
tracksLayer	Set TRUE if you want to show layer control on the map. Default is TRUE
playControl	Set TRUE if play button is needed. Default is TRUE
dateControl	Set TRUE if date label is needed. Default is TRUE
sliderControl	Set TRUE if slider control is needed. Default is TRUE
orientIcons	Set TRUE if you want icons to orient themselves on each tick based on the bearing towards their next location. Default: FALSE
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\(\)](#)

---

reachabilityOptions    *reachabilityOptions*

---

### Description

Add extra options. For a full list please visit the [plugin repository](#).

### Usage

```
reachabilityOptions(
  collapsed = TRUE,
  pane = "overlayPane",
  position = "topleft",
  ...
)
```

### Arguments

collapsed	Should the control widget start in a collapsed mode. Default is TRUE
pane	Leaflet pane to add the isolines GeoJSON to. Default is overlayPane
position	Leaflet control pane position. Default is topleft
...	Further arguments passed to 'L.Control.Reachability'

### Value

A list of options for addReachability

### References

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

### See Also

Other Reachability Functions: [addReachability\(\)](#), [removeReachability\(\)](#)

---

removeallItemsContextmenu  
*removeallItemsContextmenu*

---

### Description

Remove all contextmenu items from the map.

### Usage

```
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\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

removeAntpath	<i>removeAntpath</i>
---------------	----------------------

---

**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	<i>Remove arrowheads from Lines by layerId</i>
-----------------	--

---

**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	<i>removeEasyprint</i>
-----------------	------------------------

---

**Description**

Removes the easyprint control from the map.

**Usage**

```
removeEasyprint(map)
```

**Arguments**

map	the map widget
-----	----------------

**Value**

A leaflet map object

**See Also**

Other EasyPrint Functions: [addEasyprint\(\)](#), [easyprintMap\(\)](#), [easyprintOptions\(\)](#)

---

removeItemContextmenu *removeItemContextmenu*

---

### Description

Remove a contextmenu item by index.

### Usage

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

### Arguments

map	a map widget object created from <a href="#">leaflet</a>
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\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#), [showContextmenu\(\)](#)

---

removePlayback *removePlayback*

---

### Description

Remove the Playback controls and markers.

### Usage

```
removePlayback(map)
```

### Arguments

map	the map widget
-----	----------------

### Value

the new map object



**See Also**

Other Playback Functions: [addPlayback\(\)](#), [playbackOptions\(\)](#)

---

removeReachability      *removeReachability*

---

**Description**

Remove the reachability controls.

**Usage**

```
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**

```
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 removed.

**Value**

the new map object

**See Also**

Other Sidebar Functions: [addSidebar\(\)](#), [closeSidebar\(\)](#), [openSidebar\(\)](#), [sidebar\\_pane\(\)](#), [sidebar\\_tabs\(\)](#)

---

removeSidebyside      *removeSidebyside*

---

**Description**

removeSidebyside

**Usage**

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

**Arguments**

map	a map widget
layerId	the layer id of the <a href="#">addSidebyside</a> layer

**Value**

the new map object

**See Also**

Other Sidebyside Functions: [addSidebyside\(\)](#)

---

removeTimeslider      *removeTimeslider*

---

**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	<i>removeVelocity</i>
----------------	-----------------------

---

**Description**

removeVelocity

**Usage**

```
removeVelocity(map, group)
```

**Arguments**

map	the map widget
group	the group to remove

**Value**

the new map object

**See Also**

Other Velocity Functions: [addVelocity\(\)](#), [setOptionsVelocity\(\)](#), [velocityOptions\(\)](#)

---

setDate	<i>Set Date for GIBS Layers</i>
---------	---------------------------------

---

**Description**

Set a new date for multi-temporal layers.

**Usage**

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

**Arguments**

map	a map widget object created from <a href="#">leaflet()</a>
layers	A character vector of GIBS-layers. See <a href="#">gibs_layers</a>
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)
```

**Arguments**

map	a map widget object created from <a href="#">leaflet</a>
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\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [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 <a href="#">velocityOptions</a>

**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 <a href="#">leaflet()</a>
layers	A character vector of GIBS-layers. See <a href="#">gibs_layers</a>
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	<i>showContextmenu</i>
-----------------	------------------------

---

**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 <a href="#">leaflet()</a>
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 <a href="#">leaflet()</a> initially, but can be overridden

**Value**

A leaflet map object

**See Also**

Other Contextmenu Functions: [addContextmenu\(\)](#), [addItemContextmenu\(\)](#), [context\\_mapmenuItems\(\)](#), [context\\_markermenuItems\(\)](#), [context\\_menuItem\(\)](#), [disableContextmenu\(\)](#), [enableContextmenu\(\)](#), [hideContextmenu\(\)](#), [insertItemContextmenu\(\)](#), [mapmenuItems\(\)](#), [markermenuItems\(\)](#), [menuItem\(\)](#), [removeItemContextmenu\(\)](#), [removeallItemsContextmenu\(\)](#), [setDisabledContextmenu\(\)](#)

---

showHexbin	<i>showHexbin</i>
------------	-------------------

---

**Description**

Show the hexbinLayer.

**Usage**

```
showHexbin(map)
```

**Arguments**

map	The map widget
-----	----------------

**Value**

the new map object

**See Also**

Other Hexbin-D3 Functions: [addHexbin\(\)](#), [clearHexbin\(\)](#), [hexbinOptions\(\)](#), [hideHexbin\(\)](#), [updateHexbin\(\)](#)

---

sidebar_pane	<i>Create a Sidebar Pane</i>
--------------	------------------------------

---

**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/blob/master/doc/usage.md>

**See Also**

Other Sidebar Functions: [addSidebar\(\)](#), [closeSidebar\(\)](#), [openSidebar\(\)](#), [removeSidebar\(\)](#), [sidebar\\_tabs\(\)](#)

**Examples**

```
## 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 <a href="#">addSidebar</a> . 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 <a href="#">sidebar_pane</a> .
...	The individual <a href="#">sidebar_pane</a> 's.

**Value**

A shiny.tag with individual sidebar panes

**References**

<https://github.com/Turbo87/sidebar-v2>, <https://github.com/Turbo87/sidebar-v2/blob/master/doc/usage.md>



**See Also**

Other Sidebar Functions: [addSidebar\(\)](#), [closeSidebar\(\)](#), [openSidebar\(\)](#), [removeSidebar\(\)](#), [sidebar\\_pane\(\)](#)

**Examples**

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

# run example app showing a single sidebar
runApp(paste0(system.file("examples", package = "leaflet.extras2"),
  "/sidebar_app.R"))

# run example app showing two sidebars
runApp(paste0(system.file("examples", package = "leaflet.extras2"),
  "/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**

```
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)
```

**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\(\)](#)

---

timesliderOptions      *timesliderOptions*

---

**Description**

A list of options for [addTimeslider](#).

**Usage**

```
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

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

### Value

A list of options for addTimeslider

### References

<https://github.com/dwilhelm89/LeafletSlider>

### See Also

Other Timeslider Functions: [addTimeslider\(\)](#), [removeTimeslider\(\)](#)

---

to_jsonformat	<i>to_jsonformat Transform object to JSON expected format</i>
---------------	---

---

**Description**

to\_jsonformat Transform object to JSON expected format

**Usage**

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

**Arguments**

data	The data
time	Name of the time column.
popup	Name of the popup column.
label	Name of the label column.
name	Name of the name column.

**Value**

A list that is transformed to the expected JSON format

---

to_ms	<i>to_ms Change POSIX or Date to milliseconds</i>
-------	---

---

**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 *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 <a href="#">leaflet()</a>
data	the data object from which the argument values are derived; by default, it is the data object provided to <a href="#">leaflet()</a> initially, but can be overridden
lng	a numeric vector of longitudes, or a one-sided formula of the form $\sim x$ 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

*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\(\)](#)

---

[.leaflet\_mapkey\_icon\_set  
                           *leaflet\_mapkey\_icon\_set*

---

**Description**

leaflet\_mapkey\_icon\_set

**Usage**

```
## S3 method for class 'leaflet_mapkey_icon_set'
x[i]
```

**Arguments**

x	icons
i	offset

**See Also**

Other Mapkey Functions: [addMapkeyMarkers\(\)](#), [makeMapkeyIcon\(\)](#), [mapkeyIconList\(\)](#), [mapkeyIcons\(\)](#)

# Index

## \* Antpath Functions

- addAntpath, [4](#)
- antpathOptions, [38](#)
- clearAntpath, [40](#)
- removeAntpath, [70](#)

## \* Arrowhead Functions

- addArrowhead, [6](#)
- arrowheadOptions, [39](#)
- clearArrowhead, [41](#)
- removeArrowhead, [71](#)

## \* Contextmenu Functions

- addContextmenu, [8](#)
- addItemContextmenu, [16](#)
- context\_mapmenuItems, [44](#)
- context\_markermenuItems, [45](#)
- context\_menuItem, [45](#)
- disableContextmenu, [46](#)
- enableContextmenu, [49](#)
- hideContextmenu, [54](#)
- insertItemContextmenu, [57](#)
- mapmenuItems, [62](#)
- markermenuItems, [63](#)
- menuItem, [64](#)
- removeallItemsContextmenu, [69](#)
- removeItemContextmenu, [72](#)
- setDisabledContextmenu, [76](#)
- showContextmenu, [78](#)

## \* EasyPrint Functions

- addEasyprint, [9](#)
- easyprintMap, [47](#)
- easyprintOptions, [48](#)
- removeEasyprint, [71](#)

## \* GIBS Functions

- addGIBS, [10](#)
- setDate, [75](#)
- setTransparent, [77](#)

## \* Heightgraph Functions

- addHeightgraph, [11](#)
- heightgraphOptions, [51](#)

## \* Hexbin-D3 Functions

- addHexbin, [13](#)
- clearHexbin, [42](#)
- hexbinOptions, [53](#)
- hideHexbin, [55](#)
- showHexbin, [79](#)
- updateHexbin, [85](#)

## \* History Functions

- addHistory, [15](#)
- clearFuture, [42](#)
- clearHistory, [43](#)
- goBackHistory, [50](#)
- goForwardHistory, [51](#)
- historyOptions, [55](#)

## \* Mapkey Functions

- [.leaflet\_mapkey\_icon\_set, [87](#)
- addMapkeyMarkers, [19](#)
- makeMapkeyIcon, [59](#)
- mapkeyIconList, [60](#)
- mapkeyIcons, [61](#)

## \* MovingMarker Functions

- addMovingMarker, [21](#)
- movingMarkerOptions, [64](#)
- startMoving, [81](#)

## \* Openweathermap Functions

- addOpenweatherCurrent, [22](#)
- addOpenweatherTiles, [24](#)
- openweatherCurrentOptions, [66](#)
- openweatherOptions, [66](#)

## \* Playback Functions

- addPlayback, [25](#)
- playbackOptions, [67](#)
- removePlayback, [72](#)

## \* Reachability Functions

- addReachability, [28](#)
- reachabilityOptions, [69](#)
- removeReachability, [73](#)

## \* Sidebar Functions

- addSidebar, [29](#)



- closeSidebar, 43
- openSidebar, 65
- removeSidebar, 73
- sidebar\_pane, 79
- sidebar\_tabs, 80
- \* **Sidebyside Functions**
  - addSidebyside, 30
  - removeSidebyside, 74
- \* **Tangram Functions**
  - addTangram, 32
- \* **Timeslider Functions**
  - addTimeslider, 33
  - removeTimeslider, 74
  - timesliderOptions, 82
- \* **Velocity Functions**
  - addVelocity, 35
  - removeVelocity, 75
  - setOptionsVelocity, 77
  - velocityOptions, 86
- \* **WMS Functions**
  - addWMS, 36
- \* **datasets**
  - gibs\_layers, 50
- \* **leafletSync Functions**
  - addLeafletsync, 17
  - addLeafletsyncDependency, 18
  - isSynced, 57
  - leafletSyncOptions, 58
  - unsync, 85
- [.leaflet\_mapkey\_icon\_set, 20, 60–62, 87
  
- addAntpath, 4, 39, 41, 70
- addArrowhead, 6, 40, 41, 71
- addContextmenu, 8, 16, 44–46, 50, 54, 57, 63, 64, 70, 72, 76, 78
- addEasyprint, 9, 47, 49, 71
- addGeoJSON, 11
- addGIBS, 10, 76, 78
- addHeightgraph, 11, 53
- addHexbin, 13, 43, 54, 55, 79, 86
- addHistory, 15, 42, 43, 51, 56
- addItemContextmenu, 8, 16, 44–46, 50, 54, 57, 63, 64, 70, 72, 76, 78
- addLabelgun, 16
- addLatLngMoving (startMoving), 81
- addLayersControl, 5, 7, 10, 12, 14, 20, 21, 23, 24, 30, 32, 35, 37
- addLeafletsync, 17, 19, 58, 59, 85
- addLeafletsyncDependency, 18, 18, 58, 59, 85
- addMapkeyMarkers, 19, 60–62, 87
- addMovingMarker, 21, 65, 82
- addOpenweatherCurrent, 22, 25, 66, 67
- addOpenweatherTiles, 23, 24, 66, 67
- addPlayback, 25, 67, 68, 73
- addReachability, 28, 69, 73
- addSidebar, 29, 44, 65, 74, 80, 81
- addSidebyside, 30, 74
- addSpinner, 31
- addStationMoving (startMoving), 81
- addTangram, 32
- addTimeslider, 33, 75, 82, 83
- addVelocity, 35, 75, 77, 87
- addWMS, 36
- antpathOptions, 4, 5, 38, 41, 70
- arrowheadOptions, 6, 7, 39, 41, 71
  
- clearAntpath, 5, 39, 40, 70
- clearArrowhead, 7, 40, 41, 71
- clearFuture, 15, 42, 43, 51, 56
- clearGroup, 5, 7, 10, 12, 14, 20, 21, 23, 24, 32, 35, 37
- clearHexbin, 14, 42, 54, 55, 79, 86
- clearHistory, 15, 42, 43, 51, 56
- closeSidebar, 29, 43, 65, 74, 80, 81
- context\_mapmenuItems, 8, 16, 44, 45, 46, 50, 54, 57, 63, 64, 70, 72, 76, 78
- context\_markermenuItems, 8, 16, 44, 45, 46, 50, 54, 57, 63, 64, 70, 72, 76, 78
- context\_menuItem, 8, 16, 44, 45, 45, 46, 50, 54, 57, 63, 64, 70, 72, 76, 78
  
- disableContextmenu, 8, 16, 44–46, 46, 50, 54, 57, 63, 64, 70, 72, 76, 78
  
- easyprintMap, 10, 47, 49, 71
- easyprintOptions, 9, 10, 47, 48, 71
- enableContextmenu, 8, 16, 44–46, 49, 54, 57, 63, 64, 70, 72, 76, 78
  
- gibs\_layers, 10, 50, 75, 77
- goBackHistory, 15, 42, 43, 50, 51, 56
- goForwardHistory, 15, 42, 43, 51, 51, 56
  
- heightgraphOptions, 12, 13, 51
- hexbinOptions, 14, 43, 53, 55, 79, 86
- hideContextmenu, 8, 16, 44–46, 50, 54, 57, 63, 64, 70, 72, 76, 78

- hideHexbin, [14](#), [43](#), [54](#), [55](#), [79](#), [86](#)
- historyOptions, [15](#), [42](#), [43](#), [51](#), [55](#)
- htmlEscape, [5](#), [7](#), [20](#), [22](#), [34](#)
- insertItemContextmenu, [8](#), [16](#), [44–46](#), [50](#), [54](#), [57](#), [63](#), [64](#), [70](#), [72](#), [76](#), [78](#)
- isSynced, [18](#), [19](#), [57](#), [59](#), [85](#)
- JS, [45](#), [53](#), [54](#), [64](#)
- labelOptions, [5](#), [7](#), [20](#), [22](#), [26](#), [34](#)
- leaflet, [4](#), [6](#), [8–10](#), [12](#), [14–17](#), [23](#), [24](#), [31](#), [35](#), [37](#), [41](#), [46](#), [49](#), [54](#), [57](#), [70](#), [72](#), [75–78](#), [86](#)
- leaflet.extras2, [58](#)
- leafletOptions, [8](#)
- leafletOutput, [29](#)
- leafletProxy, [41–43](#), [50](#), [51](#), [70](#)
- leafletSyncOptions, [17–19](#), [58](#), [58](#), [85](#)
- makeIcon, [26](#)
- makeMapkeyIcon, [20](#), [59](#), [60–62](#), [87](#)
- mapkeyIconList, [20](#), [60](#), [60](#), [62](#), [87](#)
- mapkeyIcons, [20](#), [60](#), [61](#), [61](#), [87](#)
- mapmenuItem, [8](#), [16](#), [44–46](#), [50](#), [54](#), [57](#), [62](#), [63](#), [64](#), [70](#), [72](#), [76](#), [78](#)
- markerClusterOptions, [20](#)
- markermenuItem, [8](#), [16](#), [44–46](#), [50](#), [54](#), [57](#), [63](#), [63](#), [64](#), [70](#), [72](#), [76](#), [78](#)
- markerOptions, [8](#), [20](#), [22](#)
- menuItem, [8](#), [16](#), [44–46](#), [50](#), [54](#), [57](#), [63](#), [64](#), [70](#), [72](#), [76](#), [78](#)
- moveToMoving (startMoving), [81](#)
- movingMarkerOptions, [22](#), [64](#), [82](#)
- openSidebar, [29](#), [44](#), [65](#), [74](#), [80](#), [81](#)
- openweatherCurrentOptions, [23](#), [25](#), [66](#), [67](#)
- openweatherOptions, [23–25](#), [66](#), [66](#)
- pathOptions, [8](#), [12](#), [26](#)
- pauseMoving (startMoving), [81](#)
- playbackOptions, [26](#), [67](#), [73](#)
- popupOptions, [5](#), [7](#), [20](#), [22](#), [26](#), [34](#), [37](#)
- reachabilityOptions, [28](#), [69](#), [73](#)
- removeallItemsContextmenu, [8](#), [16](#), [44–46](#), [50](#), [54](#), [57](#), [63](#), [64](#), [69](#), [72](#), [76](#), [78](#)
- removeAntpath, [5](#), [39](#), [41](#), [70](#)
- removeArrowhead, [7](#), [40](#), [41](#), [71](#)
- removeControl, [12](#)
- removeEasyprint, [10](#), [47](#), [49](#), [71](#)
- removeItemContextmenu, [8](#), [16](#), [44–46](#), [50](#), [54](#), [57](#), [63](#), [64](#), [70](#), [72](#), [76](#), [78](#)
- removePlayback, [26](#), [68](#), [72](#)
- removeReachability, [28](#), [69](#), [73](#)
- removeSidebar, [29](#), [44](#), [65](#), [73](#), [80](#), [81](#)
- removeSidebyside, [30](#), [74](#)
- removeTimeslider, [34](#), [74](#), [83](#)
- removeVelocity, [36](#), [75](#), [77](#), [87](#)
- resumeMoving (startMoving), [81](#)
- setDate, [11](#), [75](#), [78](#)
- setDisabledContextmenu, [8](#), [16](#), [44–46](#), [50](#), [54](#), [57](#), [63](#), [64](#), [70](#), [72](#), [76](#), [78](#)
- setOptionsVelocity, [36](#), [75](#), [77](#), [87](#)
- setTransparent, [11](#), [76](#), [77](#)
- showContextmenu, [8](#), [16](#), [44–46](#), [50](#), [54](#), [57](#), [63](#), [64](#), [70](#), [72](#), [76](#), [78](#)
- showHexbin, [14](#), [43](#), [54](#), [55](#), [79](#), [86](#)
- sidebar\_pane, [29](#), [44](#), [65](#), [74](#), [79](#), [80](#), [81](#)
- sidebar\_tabs, [29](#), [44](#), [65](#), [73](#), [74](#), [80](#), [80](#)
- startMoving, [22](#), [65](#), [81](#)
- startSpinner (addSpinner), [31](#)
- stopMoving (startMoving), [81](#)
- stopSpinner (addSpinner), [31](#)
- timesliderOptions, [34](#), [75](#), [82](#)
- to\_jsonformat, [84](#)
- to\_ms, [84](#)
- unsync, [18](#), [19](#), [58](#), [59](#), [85](#)
- updateHexbin, [14](#), [43](#), [54](#), [55](#), [79](#), [85](#)
- velocityOptions, [36](#), [75](#), [77](#), [86](#)