

Package ‘mapgl’

January 13, 2025

Title Interactive Maps with 'Mapbox GL JS' and 'MapLibre GL JS'

Version 0.2.0

Date 2025-01-13

Description Provides an interface to the 'Mapbox GL JS' (<<https://docs.mapbox.com/mapbox-gl-js/guides>>) and the 'MapLibre GL JS' (<<https://maplibre.org/maplibre-gl-js/docs/>>) interactive mapping libraries to help users create custom interactive maps in R. Users can create interactive globe visualizations; layer 'sf' objects to create filled maps, circle maps, 'heatmaps', and three-dimensional graphics; and customize map styles and views. The package also includes utilities to use 'Mapbox' and 'MapLibre' maps in 'Shiny' web applications.

URL <https://walker-data.com/mapgl/>

BugReports <https://github.com/walkerke/mapgl/issues>

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports htmlwidgets, geojsonsf, sf, rlang, htmltools, grDevices, base64enc, terra, classInt, shiny

Suggests mapboxapi, usethis, leaflet

NeedsCompilation no

Author Kyle Walker [aut, cre]

Maintainer Kyle Walker <kyle@walker-data.com>

Repository CRAN

Date/Publication 2025-01-13 21:00:02 UTC

Contents

add_categorical_legend	3
add_circle_layer	5

add_continuous_legend	8
add_draw_control	9
add_fill_extrusion_layer	10
add_fill_layer	12
add_fullscreen_control	14
add_geocoder_control	15
add_geolocate_control	16
add_globe_minimap	17
add_heatmap_layer	18
add_image	20
add_image_source	22
add_layer	22
add_layers_control	24
add_legend	25
add_line_layer	27
add_markers	29
add_navigation_control	31
add_raster_dem_source	32
add_raster_layer	33
add_raster_source	35
add_reset_control	36
add_scale_control	36
add_source	37
add_symbol_layer	38
add_vector_source	44
add_video_source	44
carto_style	45
clear_controls	45
clear_layer	46
clear_legend	46
clear_markers	47
cluster_options	47
compare	48
ease_to	50
fit_bounds	50
fly_to	51
get_column	51
get_drawn_features	52
interpolate	53
jump_to	54
mapboxgl	54
mapboxglOutput	55
mapboxgl_proxy	56
mapbox_style	56
maplibre	57
maplibreOutput	58
maplibre_proxy	58
maptiler_style	59

match_expr	59
move_layer	60
on_section	60
renderMapboxgl	61
renderMaplibre	61
set_config_property	62
set_filter	62
set_fog	63
set_layout_property	63
set_paint_property	64
set_projection	65
set_source	65
set_style	66
set_terrain	67
set_tooltip	68
set_view	68
step_expr	69
story_leaflet	70
story_map	71
story_maplibre	72
story_section	73

Index	74
--------------	-----------

add_categorical_legend

Add a categorical legend to a Mapbox GL map

Description

This function adds a categorical legend to a Mapbox GL map. It supports customizable colors, sizes, and shapes for legend items.

Usage

```
add_categorical_legend(
  map,
  legend_title,
  values,
  colors,
  circular_patches = FALSE,
  position = "top-left",
  unique_id = NULL,
  sizes = NULL,
  add = FALSE,
  width = NULL
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> function.
legend_title	The title of the legend.
values	A vector of categories or values to be displayed in the legend.
colors	The corresponding colors for the values. Can be a vector of colors or a single color.
circular_patches	Logical, whether to use circular patches in the legend. Default is <code>FALSE</code> .
position	The position of the legend on the map. One of "top-left", "bottom-left", "top-right", "bottom-right". Default is "top-left".
unique_id	A unique ID for the legend container. If <code>NULL</code> , a random ID will be generated.
sizes	An optional numeric vector of sizes for the legend patches, or a single numeric value. If provided as a vector, it should have the same length as <code>values</code> . If <code>circular_patches</code> is <code>FALSE</code> (for square patches), sizes represent the width and height of the patch in pixels. If <code>circular_patches</code> is <code>TRUE</code> , sizes represent the radius of the circle.
add	Logical, whether to add this legend to existing legends (<code>TRUE</code>) or replace existing legends (<code>FALSE</code>). Default is <code>FALSE</code> .
width	The width of the legend. Can be specified in pixels (e.g., "250px") or as "auto". Default is <code>NULL</code> , which uses the built-in default.

Value

The updated map object with the legend added.

Examples

```
## Not run:
library(mapboxgl)
map <- mapboxgl(
  center = c(-96, 37.8),
  zoom = 3
)
map %>% add_categorical_legend(
  legend_title = "Population",
  values = c("Low", "Medium", "High"),
  colors = c("#FED976", "#FEB24C", "#FD8D3C"),
  circular_patches = TRUE,
  sizes = c(10, 15, 20),
  width = "300px"
)

## End(Not run)
```

add_circle_layer	<i>Add a circle layer to a Mapbox GL map</i>
------------------	--

Description

Add a circle layer to a Mapbox GL map

Usage

```
add_circle_layer(  
  map,  
  id,  
  source,  
  source_layer = NULL,  
  circle_blur = NULL,  
  circle_color = NULL,  
  circle_opacity = NULL,  
  circle_radius = NULL,  
  circle_sort_key = NULL,  
  circle_stroke_color = NULL,  
  circle_stroke_opacity = NULL,  
  circle_stroke_width = NULL,  
  circle_translate = NULL,  
  circle_translate_anchor = "map",  
  visibility = "visible",  
  slot = NULL,  
  min_zoom = NULL,  
  max_zoom = NULL,  
  popup = NULL,  
  tooltip = NULL,  
  hover_options = NULL,  
  before_id = NULL,  
  filter = NULL,  
  cluster_options = NULL  
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> function.
id	A unique ID for the layer.
source	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
source_layer	The source layer (for vector sources).
circle_blur	Amount to blur the circle.

circle_color	The color of the circle.
circle_opacity	The opacity at which the circle will be drawn.
circle_radius	Circle radius.
circle_sort_key	Sorts features in ascending order based on this value.
circle_stroke_color	The color of the circle's stroke.
circle_stroke_opacity	The opacity of the circle's stroke.
circle_stroke_width	The width of the circle's stroke.
circle_translate	The geometry's offset. Values are c(x, y) where negatives indicate left and up.
circle_translate_anchor	Controls the frame of reference for circle-translate.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).
filter	An optional filter expression to subset features in the layer.
cluster_options	A list of options for clustering circles, created by the cluster_options() function.

Value

The modified map object with the new circle layer added.

Examples

```
## Not run:
library(mapgl)
library(sf)
library(dplyr)

# Set seed for reproducibility
set.seed(1234)
```

```
# Define the bounding box for Washington DC (approximately)
bbox <- st_bbox(
  c(
    xmin = -77.119759,
    ymin = 38.791645,
    xmax = -76.909393,
    ymax = 38.995548
  ),
  crs = st_crs(4326)
)

# Generate 30 random points within the bounding box
random_points <- st_as_sf(
  data.frame(
    id = 1:30,
    lon = runif(30, bbox["xmin"], bbox["xmax"]),
    lat = runif(30, bbox["ymin"], bbox["ymax"])
  ),
  coords = c("lon", "lat"),
  crs = 4326
)

# Assign random categories
categories <- c("music", "bar", "theatre", "bicycle")
random_points <- random_points %>%
  mutate(category = sample(categories, n(), replace = TRUE))

# Map with circle layer
mapboxgl(style = mapbox_style("light")) %>%
  fit_bounds(random_points, animate = FALSE) %>%
  add_circle_layer(
    id = "poi-layer",
    source = random_points,
    circle_color = match_expr(
      "category",
      values = c(
        "music", "bar", "theatre",
        "bicycle"
      ),
    ),
    stops = c(
      "#1f78b4", "#33a02c",
      "#e31a1c", "#ff7f00"
    )
  ),
  circle_radius = 8,
  circle_stroke_color = "#ffffff",
  circle_stroke_width = 2,
  circle_opacity = 0.8,
  tooltip = "category",
  hover_options = list(
    circle_radius = 12,
    circle_color = "#ffff99"
  )
)
```

```

) %>%
add_categorical_legend(
  legend_title = "Points of Interest",
  values = c("Music", "Bar", "Theatre", "Bicycle"),
  colors = c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00"),
  circular_patches = TRUE
)

## End(Not run)

```

add_continuous_legend *Add a continuous legend*

Description

Add a continuous legend

Usage

```

add_continuous_legend(
  map,
  legend_title,
  values,
  colors,
  position = "top-left",
  unique_id = NULL,
  add = FALSE,
  width = NULL
)

```

Arguments

map	A map object created by the mapboxgl function.
legend_title	The title of the legend.
values	The values being represented on the map (vector of stops).
colors	The colors used to generate the color ramp.
position	The position of the legend on the map (one of "top-left", "bottom-left", "top-right", "bottom-right").
unique_id	A unique ID for the legend container. Defaults to NULL.
add	Logical, whether to add this legend to existing legends (TRUE) or replace existing legends (FALSE). Default is FALSE.
width	The width of the legend. Can be specified in pixels (e.g., "250px") or as "auto". Default is NULL, which uses the built-in default.

Value

The updated map object with the legend added.

add_draw_control	<i>Add a draw control to a map</i>
------------------	------------------------------------

Description

Add a draw control to a map

Usage

```
add_draw_control(  
  map,  
  position = "top-left",  
  freehand = FALSE,  
  simplify_freehand = FALSE,  
  orientation = "vertical",  
  ...  
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
position	A string specifying the position of the draw control. One of "top-right", "top-left", "bottom-right", or "bottom-left".
freehand	Logical, whether to enable freehand drawing mode. Default is FALSE.
simplify_freehand	Logical, whether to apply simplification to freehand drawings. Default is FALSE.
orientation	A string specifying the orientation of the draw control. Either "vertical" (default) or "horizontal".
...	Additional named arguments. See https://github.com/mapbox/mapbox-gl-draw/blob/main/docs/API.md#options for a list of options.

Value

The modified map object with the draw control added.

Examples

```
## Not run:  
library(mapgl)  
  
mapboxgl(  
  style = mapbox_style("streets"),  
  center = c(-74.50, 40),  
  zoom = 9  
) |>  
  add_draw_control()  
  
## End(Not run)
```

`add_fill_extrusion_layer`*Add a fill-extrusion layer to a Mapbox GL map*

Description

Add a fill-extrusion layer to a Mapbox GL map

Usage

```
add_fill_extrusion_layer(  
  map,  
  id,  
  source,  
  source_layer = NULL,  
  fill_extrusion_base = NULL,  
  fill_extrusion_color = NULL,  
  fill_extrusion_height = NULL,  
  fill_extrusion_opacity = NULL,  
  fill_extrusion_pattern = NULL,  
  fill_extrusion_translate = NULL,  
  fill_extrusion_translate_anchor = "map",  
  visibility = "visible",  
  slot = NULL,  
  min_zoom = NULL,  
  max_zoom = NULL,  
  popup = NULL,  
  tooltip = NULL,  
  hover_options = NULL,  
  before_id = NULL,  
  filter = NULL  
)
```

Arguments

<code>map</code>	A map object created by the <code>mapboxgl</code> function.
<code>id</code>	A unique ID for the layer.
<code>source</code>	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
<code>source_layer</code>	The source layer (for vector sources).
<code>fill_extrusion_base</code>	The base height of the fill extrusion.
<code>fill_extrusion_color</code>	The color of the fill extrusion.

fill_extrusion_height	The height of the fill extrusion.
fill_extrusion_opacity	The opacity of the fill extrusion.
fill_extrusion_pattern	Name of image in sprite to use for drawing image fills.
fill_extrusion_translate	The geometry's offset. Values are c(x, y) where negatives indicate left and up.
fill_extrusion_translate_anchor	Controls the frame of reference for fill-extrusion-translate.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).
filter	An optional filter expression to subset features in the layer.

Value

The modified map object with the new fill-extrusion layer added.

Examples

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

maplibre(
  style = maptiler_style("basic"),
  center = c(-74.0066, 40.7135),
  zoom = 15.5,
  pitch = 45,
  bearing = -17.6
) |>
  add_vector_source(
    id = "openmaptiles",
    url = paste0(
      "https://api.maptiler.com/tiles/v3/tiles.json?key=",
      Sys.getenv("MAPTILER_API_KEY")
    )
  ) |>
  add_fill_extrusion_layer(
```

```
        id = "3d-buildings",
        source = "openmaptiles",
        source_layer = "building",
        fill_extrusion_color = interpolate(
            column = "render_height",
            values = c(0, 200, 400),
            stops = c("lightgray", "royalblue", "lightblue")
        ),
        fill_extrusion_height = list(
            "interpolate",
            list("linear"),
            list("zoom"),
            15,
            0,
            16,
            list("get", "render_height")
        )
    )
)

## End(Not run)
```

add_fill_layer

Add a fill layer to a map

Description

Add a fill layer to a map

Usage

```
add_fill_layer(
  map,
  id,
  source,
  source_layer = NULL,
  fill_antialias = TRUE,
  fill_color = NULL,
  fill_emissive_strength = NULL,
  fill_opacity = NULL,
  fill_outline_color = NULL,
  fill_pattern = NULL,
  fill_sort_key = NULL,
  fill_translate = NULL,
  fill_translate_anchor = "map",
  fill_z_offset = NULL,
  visibility = "visible",
  slot = NULL,
  min_zoom = NULL,
  max_zoom = NULL,
```

```

    popup = NULL,
    tooltip = NULL,
    hover_options = NULL,
    before_id = NULL,
    filter = NULL
)

```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
id	A unique ID for the layer.
source	The ID of the source, alternatively an sf object (which will be converted to a GeoJSON source) or a named list that specifies type and url for a remote source.
source_layer	The source layer (for vector sources).
fill_antialias	Whether or not the fill should be antialiased.
fill_color	The color of the filled part of this layer.
fill_emissive_strength	Controls the intensity of light emitted on the source features.
fill_opacity	The opacity of the entire fill layer.
fill_outline_color	The outline color of the fill.
fill_pattern	Name of image in sprite to use for drawing image fills.
fill_sort_key	Sorts features in ascending order based on this value.
fill_translate	The geometry's offset. Values are c(x, y) where negatives indicate left and up.
fill_translate_anchor	Controls the frame of reference for fill-translate.
fill_z_offset	Specifies an uniform elevation in meters.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).
filter	An optional filter expression to subset features in the layer.

Value

The modified map object with the new fill layer added.

Examples

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

fl_age <- get_acs(
  geography = "tract",
  variables = "B01002_001",
  state = "FL",
  year = 2022,
  geometry = TRUE
)

mapboxgl() |>
  fit_bounds(fl_age, animate = FALSE) |>
  add_fill_layer(
    id = "fl_tracts",
    source = fl_age,
    fill_color = interpolate(
      column = "estimate",
      values = c(20, 80),
      stops = c("lightblue", "darkblue"),
      na_color = "lightgrey"
    ),
    fill_opacity = 0.5
  )

## End(Not run)
```

add_fullscreen_control

Add a fullscreen control to a map

Description

Add a fullscreen control to a map

Usage

```
add_fullscreen_control(map, position = "top-right")
```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
position	A string specifying the position of the fullscreen control. One of "top-right", "top-left", "bottom-right", or "bottom-left".

Value

The modified map object with the fullscreen control added.

Examples

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

maplibre(
  style = maptiler_style("streets"),
  center = c(11.255, 43.77),
  zoom = 13
) |>
  add_fullscreen_control(position = "top-right")

## End(Not run)
```

add_geocoder_control *Add a geocoder control to a map*

Description

This function adds a Geocoder search bar to a Mapbox GL or MapLibre GL map. By default, a marker will be added at the selected location and the map will fly to that location. The results of the geocode are accessible in a Shiny session at `input$MAPID_geocoder$result`, where MAPID is the name of your map.

Usage

```
add_geocoder_control(
  map,
  position = "top-right",
  placeholder = "Search",
  collapsed = FALSE,
  ...
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> function.
position	The position of the control. Can be one of "top-left", "top-right", "bottom-left", or "bottom-right". Default is "top-right".
placeholder	A string to use as placeholder text for the search bar. Default is "Search".
collapsed	Whether the control should be collapsed until hovered or clicked. Default is FALSE.
...	Additional parameters to pass to the Geocoder.

Value

The modified map object with the geocoder control added.

Examples

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

mapboxgl() |>
  add_geocoder_control(position = "top-left", placeholder = "Enter an address")

maplibre() |>
  add_geocoder_control(position = "top-right", placeholder = "Search location")

## End(Not run)
```

add_geolocate_control *Add a geolocate control to a map*

Description

This function adds a Geolocate control to a Mapbox GL or MapLibre GL map. The geolocate control allows users to track their current location on the map.

Usage

```
add_geolocate_control(
  map,
  position = "top-right",
  track_user = FALSE,
  show_accuracy_circle = TRUE,
  show_user_location = TRUE,
  show_user_heading = FALSE,
  fit_bounds_options = list(maxZoom = 15),
  position_options = list(enableHighAccuracy = FALSE, timeout = 6000)
)
```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
position	The position of the control. Can be one of "top-left", "top-right", "bottom-left", or "bottom-right". Default is "top-right".
track_user	Whether to actively track the user's location. If TRUE, the map will continuously update as the user moves. Default is FALSE.
show_accuracy_circle	Whether to show a circle indicating the accuracy of the location. Default is TRUE.
show_user_location	Whether to show a dot at the user's location. Default is TRUE.

show_user_heading	Whether to show an arrow indicating the device's heading when tracking location. Only works when track_user is TRUE. Default is FALSE.
fit_bounds_options	A list of options for fitting bounds when panning to the user's location. Default maxZoom is 15.
position_options	A list of Geolocation API position options. Default has enableHighAccuracy=FALSE and timeout=6000.

Value

The modified map object with the geolocate control added.

Examples

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

mapboxgl() |>
  add_geolocate_control(
    position = "top-right",
    track_user = TRUE,
    show_user_heading = TRUE
  )

## End(Not run)
```

add_globe_minimap	<i>Add a Globe Minimap to a map</i>
-------------------	-------------------------------------

Description

This function adds a globe minimap control to a Mapbox GL or Maplibre map.

Usage

```
add_globe_minimap(
  map,
  position = "bottom-right",
  globe_size = 82,
  land_color = "white",
  water_color = "rgba(30 40 70/60%)",
  marker_color = "#ff2233",
  marker_size = 1
)
```

Arguments

map	A mapboxgl or maplibre object.
position	A string specifying the position of the minimap.
globe_size	Number of pixels for the diameter of the globe. Default is 82.
land_color	HTML color to use for land areas on the globe. Default is 'white'.
water_color	HTML color to use for water areas on the globe. Default is 'rgba(30 40 70/60%)'.
marker_color	HTML color to use for the center point marker. Default is '#ff2233'.
marker_size	Scale ratio for the center point marker. Default is 1.

Value

The modified map object with the globe minimap added.

Examples

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

m <- mapboxgl() %>%
  add_globe_minimap()

m <- maplibre() %>%
  add_globe_minimap()

## End(Not run)
```

add_heatmap_layer *Add a heatmap layer to a Mapbox GL map*

Description

Add a heatmap layer to a Mapbox GL map

Usage

```
add_heatmap_layer(
  map,
  id,
  source,
  source_layer = NULL,
  heatmap_color = NULL,
  heatmap_intensity = NULL,
  heatmap_opacity = NULL,
  heatmap_radius = NULL,
  heatmap_weight = NULL,
  visibility = "visible",
```

```

    slot = NULL,
    min_zoom = NULL,
    max_zoom = NULL,
    before_id = NULL,
    filter = NULL
  )

```

Arguments

map	A map object created by the <code>mapboxgl</code> function.
id	A unique ID for the layer.
source	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
source_layer	The source layer (for vector sources).
heatmap_color	The color of the heatmap points.
heatmap_intensity	The intensity of the heatmap points.
heatmap_opacity	The opacity of the heatmap layer.
heatmap_radius	The radius of influence of each individual heatmap point.
heatmap_weight	The weight of each individual heatmap point.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).
filter	An optional filter expression to subset features in the layer.

Value

The modified map object with the new heatmap layer added.

Examples

```

## Not run:
library(mapgl)

mapboxgl(
  style = mapbox_style("dark"),
  center = c(-120, 50),
  zoom = 2
) |>
  add_heatmap_layer(
    id = "earthquakes-heat",

```

```

source = list(
  type = "geojson",
  data = "https://docs.mapbox.com/mapbox-gl-js/assets/earthquakes.geojson"
),
heatmap_weight = interpolate(
  column = "mag",
  values = c(0, 6),
  stops = c(0, 1)
),
heatmap_intensity = interpolate(
  property = "zoom",
  values = c(0, 9),
  stops = c(1, 3)
),
heatmap_color = interpolate(
  property = "heatmap-density",
  values = seq(0, 1, 0.2),
  stops = c(
    "rgba(33,102,172,0)", "rgb(103,169,207)",
    "rgb(209,229,240)", "rgb(253,219,199)",
    "rgb(239,138,98)", "rgb(178,24,43)"
  )
),
heatmap_opacity = 0.7
)

## End(Not run)

```

add_image

Add an image to the map

Description

This function adds an image to the map's style. The image can be used with `icon-image`, `background-pattern`, `fill-pattern`, or `line-pattern`.

Usage

```

add_image(
  map,
  id,
  url,
  content = NULL,
  pixel_ratio = 1,
  sdf = FALSE,
  stretch_x = NULL,
  stretch_y = NULL
)

```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
id	A string specifying the ID of the image.
url	A string specifying the URL of the image to be loaded or a path to a local image file. Must be PNG or JPEG format.
content	A vector of four numbers $c(x1, y1, x2, y2)$ defining the part of the image that can be covered by the content in text-field if <code>icon-text-fit</code> is used.
pixel_ratio	A number specifying the ratio of pixels in the image to physical pixels on the screen.
sdf	A logical value indicating whether the image should be interpreted as an SDF image.
stretch_x	A list of number pairs defining the part(s) of the image that can be stretched horizontally.
stretch_y	A list of number pairs defining the part(s) of the image that can be stretched vertically.

Value

The modified map object with the image added.

Examples

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

# Path to your local image file OR a URL to a remote image file
# that is not blocked by CORS restrictions
image_path <- "/path/to/your/image.png"

pts <- tigris::landmarks("DE")[1:100, ]

maplibre(bounds = pts) |>
  add_image("local_icon", image_path) |>
  add_symbol_layer(
    id = "local_icons",
    source = pts,
    icon_image = "local_icon",
    icon_size = 0.5,
    icon_allow_overlap = TRUE
  )

## End(Not run)
```

add_image_source *Add an image source to a Mapbox GL or Maplibre GL map*

Description

Add an image source to a Mapbox GL or Maplibre GL map

Usage

```
add_image_source(
  map,
  id,
  url = NULL,
  data = NULL,
  coordinates = NULL,
  colors = NULL
)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
url	A URL pointing to the image source.
data	A SpatRaster object from the terra package or a RasterLayer object.
coordinates	A list of coordinates specifying the image corners in clockwise order: top left, top right, bottom right, bottom left. For SpatRaster or RasterLayer objects, this will be extracted for you.
colors	A vector of colors to use for the raster image.

Value

The modified map object with the new source added.

add_layer *Add a layer to a map from a source*

Description

In many cases, you will use add_layer() internal to other layer-specific functions in mapgl. Advanced users will want to use add_layer() for more fine-grained control over the appearance of their layers.

Usage

```

add_layer(
  map,
  id,
  type = "fill",
  source,
  source_layer = NULL,
  paint = list(),
  layout = list(),
  slot = NULL,
  min_zoom = NULL,
  max_zoom = NULL,
  popup = NULL,
  tooltip = NULL,
  hover_options = NULL,
  before_id = NULL,
  filter = NULL
)

```

Arguments

map	A map object created by the <code>mapboxgl()</code> or <code>maplibre()</code> functions.
id	A unique ID for the layer.
type	The type of the layer (e.g., "fill", "line", "circle").
source	The ID of the source, alternatively an sf object (which will be converted to a GeoJSON source) or a named list that specifies type and url for a remote source.
source_layer	The source layer (for vector sources).
paint	A list of paint properties for the layer.
layout	A list of layout properties for the layer.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).
filter	An optional filter expression to subset features in the layer.

Value

The modified map object with the new layer added.

Examples

```
## Not run:
# Load necessary libraries
library(mapgl)
library(tigris)

# Load geojson data for North Carolina tracts
nc_tracts <- tracts(state = "NC", cb = TRUE)

# Create a Mapbox GL map
map <- mapboxgl(
  style = mapbox_style("light"),
  center = c(-79.0193, 35.7596),
  zoom = 7
)

# Add a source and fill layer for North Carolina tracts
map %>%
  add_source(
    id = "nc-tracts",
    data = nc_tracts
  ) %>%
  add_layer(
    id = "nc-layer",
    type = "fill",
    source = "nc-tracts",
    paint = list(
      "fill-color" = "#888888",
      "fill-opacity" = 0.4
    )
  )

## End(Not run)
```

add_layers_control *Add a layers control to the map*

Description

Add a layers control to the map

Usage

```
add_layers_control(
  map,
  position = "top-left",
  layers = NULL,
  collapsible = FALSE
)
```


Arguments

map	A map object.
position	The position of the control on the map (one of "top-left", "top-right", "bottom-left", "bottom-right").
layers	A vector of layer IDs to be included in the control. If NULL, all layers will be included.
collapsible	Whether the control should be collapsible.

Value

The modified map object with the layers control added.

Examples

```
## Not run:
library(tigris)
options(tigris_use_cache = TRUE)

rds <- roads("TX", "Tarrant")
tr <- tracts("TX", "Tarrant", cb = TRUE)

maplibre() |>
  fit_bounds(rds) |>
  add_fill_layer(
    id = "Census tracts",
    source = tr,
    fill_color = "purple",
    fill_opacity = 0.6
  ) |>
  add_line_layer(
    "Local roads",
    source = rds,
    line_color = "pink"
  ) |>
  add_layers_control(collapsible = TRUE)

## End(Not run)
```

add_legend

Add a legend to a Mapbox GL map

Description

Add a legend to a Mapbox GL map

Usage

```
add_legend(  
  map,  
  legend_title,  
  values,  
  colors,  
  type = c("continuous", "categorical"),  
  circular_patches = FALSE,  
  position = "top-left",  
  sizes = NULL,  
  add = FALSE,  
  width = NULL  
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> function.
legend_title	The title of the legend.
values	The values being represented on the map (either a vector of categories or a vector of stops).
colors	The corresponding colors for the values (either a vector of colors, a single color, or an interpolate function).
type	One of "continuous" or "categorical".
circular_patches	Logical, whether to use circular patches in the legend (only for categorical legends).
position	The position of the legend on the map (one of "top-left", "bottom-left", "top-right", "bottom-right").
sizes	An optional numeric vector of sizes for the legend patches, or a single numeric value (only for categorical legends).
add	Logical, whether to add this legend to existing legends (TRUE) or replace existing legends (FALSE). Default is FALSE.
width	The width of the legend. Can be specified in pixels (e.g., "250px") or as "auto". Default is NULL, which uses the built-in default.

Value

The updated map object with the legend added.

add_line_layer	<i>Add a line layer to a map</i>
----------------	----------------------------------

Description

Add a line layer to a map

Usage

```
add_line_layer(  
    map,  
    id,  
    source,  
    source_layer = NULL,  
    line_blur = NULL,  
    line_cap = NULL,  
    line_color = NULL,  
    line_dasharray = NULL,  
    line_emissive_strength = NULL,  
    line_gap_width = NULL,  
    line_gradient = NULL,  
    line_join = NULL,  
    line_miter_limit = NULL,  
    line_occlusion_opacity = NULL,  
    line_offset = NULL,  
    line_opacity = NULL,  
    line_pattern = NULL,  
    line_round_limit = NULL,  
    line_sort_key = NULL,  
    line_translate = NULL,  
    line_translate_anchor = "map",  
    line_trim_color = NULL,  
    line_trim_fade_range = NULL,  
    line_trim_offset = NULL,  
    line_width = NULL,  
    line_z_offset = NULL,  
    visibility = "visible",  
    slot = NULL,  
    min_zoom = NULL,  
    max_zoom = NULL,  
    popup = NULL,  
    tooltip = NULL,  
    hover_options = NULL,  
    before_id = NULL,  
    filter = NULL  
)
```

Arguments

<code>map</code>	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
<code>id</code>	A unique ID for the layer.
<code>source</code>	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
<code>source_layer</code>	The source layer (for vector sources).
<code>line_blur</code>	Amount to blur the line, in pixels.
<code>line_cap</code>	The display of line endings. One of "butt", "round", "square".
<code>line_color</code>	The color with which the line will be drawn.
<code>line_dasharray</code>	Specifies the lengths of the alternating dashes and gaps that form the dash pattern.
<code>line_emissive_strength</code>	Controls the intensity of light emitted on the source features.
<code>line_gap_width</code>	Draws a line casing outside of a line's actual path. Value indicates the width of the inner gap.
<code>line_gradient</code>	A gradient used to color a line feature at various distances along its length.
<code>line_join</code>	The display of lines when joining.
<code>line_miter_limit</code>	Used to automatically convert miter joins to bevel joins for sharp angles.
<code>line_occlusion_opacity</code>	Opacity multiplier of the line part that is occluded by 3D objects.
<code>line_offset</code>	The line's offset.
<code>line_opacity</code>	The opacity at which the line will be drawn.
<code>line_pattern</code>	Name of image in sprite to use for drawing image lines.
<code>line_round_limit</code>	Used to automatically convert round joins to miter joins for shallow angles.
<code>line_sort_key</code>	Sorts features in ascending order based on this value.
<code>line_translate</code>	The geometry's offset. Values are <code>c(x, y)</code> where negatives indicate left and up, respectively.
<code>line_translate_anchor</code>	Controls the frame of reference for <code>line-translate</code> .
<code>line_trim_color</code>	The color to be used for rendering the trimmed line section.
<code>line_trim_fade_range</code>	The fade range for the trim-start and trim-end points.
<code>line_trim_offset</code>	The line part between <code>c(trim_start, trim_end)</code> will be painted using <code>line_trim_color</code> .
<code>line_width</code>	Stroke thickness.
<code>line_z_offset</code>	Vertical offset from ground, in meters.
<code>visibility</code>	Whether this layer is displayed.

slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels)
filter	An optional filter expression to subset features in the layer.

Value

The modified map object with the new line layer added.

Examples

```
## Not run:
library(mapgl)
library(tigris)

loving_roads <- roads("TX", "Loving")

maplibre(style = maptiler_style("backdrop")) |>
  fit_bounds(loving_roads) |>
  add_line_layer(
    id = "tracks",
    source = loving_roads,
    line_color = "navy",
    line_opacity = 0.7
  )

## End(Not run)
```

 add_markers

Add markers to a Mapbox GL or Maplibre GL map

Description

Add markers to a Mapbox GL or Maplibre GL map

Usage

```
add_markers(  
  map,  
  data,  
  color = "red",  
  rotation = 0,  
  popup = NULL,  
  marker_id = NULL,  
  draggable = FALSE,  
  ...  
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
data	A length-2 numeric vector of coordinates, a list of length-2 numeric vectors, or an <code>sf POINT</code> object.
color	The color of the marker (default is "red").
rotation	The rotation of the marker (default is 0).
popup	A column name for popups (if data is an <code>sf</code> object) or a string for a single popup (if data is a numeric vector or list of vectors).
marker_id	A unique ID for the marker. For lists, names will be inherited from the list names. For <code>sf</code> objects, this should be a column name.
draggable	A boolean indicating if the marker should be draggable (default is <code>FALSE</code>).
...	Additional options passed to the marker.

Value

The modified map object with the markers added.

Examples

```
## Not run:  
library(mapgl)  
library(sf)  
  
# Create a map object  
map <- mapboxgl(  
  style = mapbox_style("streets"),  
  center = c(-74.006, 40.7128),  
  zoom = 10  
)  
  
# Add a single draggable marker with an ID  
map <- add_markers(  
  map,  
  c(-74.006, 40.7128),  
  color = "blue",
```

```

    rotation = 45,
    popup = "A marker",
    draggable = TRUE,
    marker_id = "marker1"
  )

# Add multiple markers from a named list of coordinates
coords_list <- list(marker2 = c(-74.006, 40.7128),
                    marker3 = c(-73.935242, 40.730610))
map <- add_markers(
  map,
  coords_list,
  color = "green",
  popup = "Multiple markers",
  draggable = TRUE
)

# Create an sf POINT object
points_sf <- st_as_sf(data.frame(
  id = c("marker4", "marker5"),
  lon = c(-74.006, -73.935242),
  lat = c(40.7128, 40.730610)
), coords = c("lon", "lat"), crs = 4326)
points_sf$popup <- c("Point 1", "Point 2")

# Add multiple markers from an sf object with IDs from a column
map <- add_markers(
  map,
  points_sf,
  color = "red",
  popup = "popup",
  draggable = TRUE,
  marker_id = "id"
)

## End(Not run)

```

```
add_navigation_control
```

Add a navigation control to a map

Description

Add a navigation control to a map

Usage

```
add_navigation_control(
  map,
  show_compass = TRUE,
```

```

    show_zoom = TRUE,
    visualize_pitch = FALSE,
    position = "top-right",
    orientation = "vertical"
  )

```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
show_compass	Whether to show the compass button.
show_zoom	Whether to show the zoom-in and zoom-out buttons.
visualize_pitch	Whether to visualize the pitch by rotating the X-axis of the compass.
position	The position on the map where the control will be added. Possible values are "top-left", "top-right", "bottom-left", and "bottom-right".
orientation	The orientation of the navigation control. Can be "vertical" (default) or "horizontal".

Value

The updated map object with the navigation control added.

Examples

```

## Not run:
library(mapgl)

mapboxgl() |>
  add_navigation_control(visualize_pitch = TRUE)

## End(Not run)

```

`add_raster_dem_source` *Add a raster DEM source to a Mapbox GL or Maplibre GL map*

Description

Add a raster DEM source to a Mapbox GL or Maplibre GL map

Usage

```
add_raster_dem_source(map, id, url, tileSize = 512, maxzoom = NULL)
```


Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
url	A URL pointing to the raster DEM source.
tileSize	The size of the raster tiles.
maxzoom	The maximum zoom level for the raster tiles.

Value

The modified map object with the new source added.

add_raster_layer	<i>Add a raster layer to a Mapbox GL map</i>
------------------	--

Description

Add a raster layer to a Mapbox GL map

Usage

```
add_raster_layer(
  map,
  id,
  source,
  source_layer = NULL,
  raster_brightness_max = NULL,
  raster_brightness_min = NULL,
  raster_contrast = NULL,
  raster_fade_duration = NULL,
  raster_hue_rotate = NULL,
  raster_opacity = NULL,
  raster_resampling = NULL,
  raster_saturation = NULL,
  visibility = "visible",
  slot = NULL,
  min_zoom = NULL,
  max_zoom = NULL,
  before_id = NULL
)
```

Arguments

map	A map object created by the mapboxgl function.
id	A unique ID for the layer.
source	The ID of the source.

source_layer	The source layer (for vector sources).
raster_brightness_max	The maximum brightness of the image.
raster_brightness_min	The minimum brightness of the image.
raster_contrast	Increase or reduce the brightness of the image.
raster_fade_duration	The duration of the fade-in/fade-out effect.
raster_hue_rotate	Rotates hues around the color wheel.
raster_opacity	The opacity at which the raster will be drawn.
raster_resampling	The resampling/interpolation method to use for overscaling.
raster_saturation	Increase or reduce the saturation of the image.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new raster layer added.

Examples

```
## Not run:
mapboxgl(
  style = mapbox_style("dark"),
  zoom = 5,
  center = c(-75.789, 41.874)
) |>
  add_image_source(
    id = "radar",
    url = "https://docs.mapbox.com/mapbox-gl-js/assets/radar.gif",
    coordinates = list(
      c(-80.425, 46.437),
      c(-71.516, 46.437),
      c(-71.516, 37.936),
      c(-80.425, 37.936)
    )
  ) |>
  add_raster_layer(
    id = "radar-layer",
```

```
        source = "radar",
        raster_fade_duration = 0
    )

## End(Not run)
```

add_raster_source *Add a raster tile source to a Mapbox GL or Maplibre GL map*

Description

Add a raster tile source to a Mapbox GL or Maplibre GL map

Usage

```
add_raster_source(
  map,
  id,
  url = NULL,
  tiles = NULL,
  tileSize = 256,
  maxzoom = 22
)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
url	A URL pointing to the raster tile source. (optional)
tiles	A vector of tile URLs for the raster source. (optional)
tileSize	The size of the raster tiles.
maxzoom	The maximum zoom level for the raster tiles.

Value

The modified map object with the new source added.

add_reset_control *Add a reset control to a map*

Description

This function adds a reset control to a Mapbox GL or MapLibre GL map. The reset control allows users to return to the original zoom level and center.

Usage

```
add_reset_control(map, position = "top-right", animate = TRUE, duration = NULL)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
position	The position of the control. Can be one of "top-left", "top-right", "bottom-left", or "bottom-right". Default is "top-right".
animate	Whether or not to animate the transition to the original map view; defaults to TRUE. If FALSE, the view will "jump" to the original view with no transition.
duration	The length of the transition from the current view to the original view, specified in milliseconds. This argument only works with <code>animate</code> is TRUE.

Value

The modified map object with the reset control added.

Examples

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

mapboxgl() |>
  add_reset_control(position = "top-left")

## End(Not run)
```

add_scale_control *Add a scale control to a map*

Description

This function adds a scale control to a Mapbox GL or MapLibre GL map.

Usage

```
add_scale_control(
  map,
  position = "bottom-left",
  unit = "metric",
  max_width = 100
)
```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
position	The position of the control. Can be one of "top-left", "top-right", "bottom-left", or "bottom-right". Default is "bottom-left".
unit	The unit of the scale. Can be either "imperial", "metric", or "nautical". Default is "metric".
max_width	The maximum length of the scale control in pixels. Default is 100.

Value

The modified map object with the scale control added.

Examples

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

mapboxgl() |>
  add_scale_control(position = "bottom-right", unit = "imperial")

## End(Not run)
```

add_source	<i>Add a GeoJSON or sf source to a Mapbox GL or Maplibre GL map</i>
------------	---

Description

Add a GeoJSON or sf source to a Mapbox GL or Maplibre GL map

Usage

```
add_source(map, id, data, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
data	An sf object or a URL pointing to a remote GeoJSON file.
...	Additional arguments to be passed to the JavaScript addSource method.

Value

The modified map object with the new source added.

add_symbol_layer	<i>Add a symbol layer to a map</i>
------------------	------------------------------------

Description

Add a symbol layer to a map

Usage

```
add_symbol_layer(  
  map,  
  id,  
  source,  
  source_layer = NULL,  
  icon_allow_overlap = NULL,  
  icon_anchor = NULL,  
  icon_color = NULL,  
  icon_color_brightness_max = NULL,  
  icon_color_brightness_min = NULL,  
  icon_color_contrast = NULL,  
  icon_color_saturation = NULL,  
  icon_emissive_strength = NULL,  
  icon_halo_blur = NULL,  
  icon_halo_color = NULL,  
  icon_halo_width = NULL,  
  icon_ignore_placement = NULL,  
  icon_image = NULL,  
  icon_image_cross_fade = NULL,  
  icon_keep_upright = NULL,  
  icon_offset = NULL,  
  icon_opacity = NULL,  
  icon_optional = NULL,  
  icon_padding = NULL,  
  icon_pitch_alignment = NULL,  
  icon_rotate = NULL,  
  icon_rotation_alignment = NULL,  
  icon_size = NULL,  
  icon_text_fit = NULL,  
  icon_text_fit_padding = NULL,  
  icon_translate = NULL,  
  icon_translate_anchor = NULL,  
  symbol_avoid_edges = NULL,  
  symbol_placement = NULL,
```

```
symbol_sort_key = NULL,  
symbol_spacing = NULL,  
symbol_z_elevate = NULL,  
symbol_z_offset = NULL,  
symbol_z_order = NULL,  
text_allow_overlap = NULL,  
text_anchor = NULL,  
text_color = "black",  
text_emissive_strength = NULL,  
text_field = NULL,  
text_font = NULL,  
text_halo_blur = NULL,  
text_halo_color = NULL,  
text_halo_width = NULL,  
text_ignore_placement = NULL,  
text_justify = NULL,  
text_keep_upright = NULL,  
text_letter_spacing = NULL,  
text_line_height = NULL,  
text_max_angle = NULL,  
text_max_width = NULL,  
text_offset = NULL,  
text_opacity = NULL,  
text_optional = NULL,  
text_padding = NULL,  
text_pitch_alignment = NULL,  
text_radial_offset = NULL,  
text_rotate = NULL,  
text_rotation_alignment = NULL,  
text_size = NULL,  
text_transform = NULL,  
text_translate = NULL,  
text_translate_anchor = NULL,  
text_variable_anchor = NULL,  
text_writing_mode = NULL,  
visibility = "visible",  
slot = NULL,  
min_zoom = NULL,  
max_zoom = NULL,  
popup = NULL,  
tooltip = NULL,  
hover_options = NULL,  
before_id = NULL,  
filter = NULL,  
cluster_options = NULL  
)
```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
id	A unique ID for the layer.
source	The ID of the source, alternatively an sf object (which will be converted to a GeoJSON source) or a named list that specifies type and url for a remote source.
source_layer	The source layer (for vector sources).
icon_allow_overlap	If TRUE, the icon will be visible even if it collides with other previously drawn symbols.
icon_anchor	Part of the icon placed closest to the anchor.
icon_color	The color of the icon. This is not supported for many Mapbox icons; read more at https://docs.mapbox.com/help/troubleshooting/using-recolorable-images-in-mapbox-m
icon_color_brightness_max	The maximum brightness of the icon color.
icon_color_brightness_min	The minimum brightness of the icon color.
icon_color_contrast	The contrast of the icon color.
icon_color_saturation	The saturation of the icon color.
icon_emissive_strength	The strength of the icon's emissive color.
icon_halo_blur	The blur applied to the icon's halo.
icon_halo_color	The color of the icon's halo.
icon_halo_width	The width of the icon's halo.
icon_ignore_placement	If TRUE, the icon will be visible even if it collides with other symbols.
icon_image	Name of image in sprite to use for drawing an image background. To use values in a column of your input dataset, use get_column('YOUR_ICON_COLUMN_NAME'). Images can also be loaded with the add_image() function which should precede the add_symbol_layer() function.
icon_image_cross_fade	The cross-fade parameter for the icon image.
icon_keep_upright	If TRUE, the icon will be kept upright.
icon_offset	Offset distance of icon.
icon_opacity	The opacity at which the icon will be drawn.
icon_optional	If TRUE, the icon will be optional.
icon_padding	Padding around the icon.

icon_pitch_alignment	Alignment of the icon with respect to the pitch of the map.
icon_rotate	Rotates the icon clockwise.
icon_rotation_alignment	Alignment of the icon with respect to the map.
icon_size	The size of the icon, specified relative to the original size of the image. For example, a value of 5 would make the icon 5 times larger than the original size, whereas a value of 0.5 would make the icon half the size of the original.
icon_text_fit	Scales the text to fit the icon.
icon_text_fit_padding	Padding for text fitting the icon.
icon_translate	The offset distance of the icon.
icon_translate_anchor	Controls the frame of reference for icon-translate.
symbol_avoid_edges	If TRUE, the symbol will be avoided when near the edges.
symbol_placement	Placement of the symbol on the map.
symbol_sort_key	Sorts features in ascending order based on this value.
symbol_spacing	Spacing between symbols.
symbol_z_elevate	If TRUE, positions the symbol on top of a fill-extrusion layer. Requires symbol_placement to be set to "point" and symbol-z-order to be set to "auto".
symbol_z_offset	The elevation of the symbol, in meters. Use get_column() to get elevations from a column in the dataset.
symbol_z_order	Orders the symbol z-axis.
text_allow_overlap	If TRUE, the text will be visible even if it collides with other previously drawn symbols.
text_anchor	Part of the text placed closest to the anchor.
text_color	The color of the text.
text_emissive_strength	The strength of the text's emissive color.
text_field	Value to use for a text label.
text_font	Font stack to use for displaying text.
text_halo_blur	The blur applied to the text's halo.
text_halo_color	The color of the text's halo.
text_halo_width	The width of the text's halo.

text_ignore_placement	If TRUE, the text will be visible even if it collides with other symbols.
text_justify	The justification of the text.
text_keep_upright	If TRUE, the text will be kept upright.
text_letter_spacing	Spacing between text letters.
text_line_height	Height of the text lines.
text_max_angle	Maximum angle of the text.
text_max_width	Maximum width of the text.
text_offset	Offset distance of text.
text_opacity	The opacity at which the text will be drawn.
text_optional	If TRUE, the text will be optional.
text_padding	Padding around the text.
text_pitch_alignment	Alignment of the text with respect to the pitch of the map.
text_radial_offset	Radial offset of the text.
text_rotate	Rotates the text clockwise.
text_rotation_alignment	Alignment of the text with respect to the map.
text_size	The size of the text.
text_transform	Transform applied to the text.
text_translate	The offset distance of the text.
text_translate_anchor	Controls the frame of reference for text-translate.
text_variable_anchor	Variable anchor for the text.
text_writing_mode	Writing mode for the text.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover. Not all elements of SVG icons can be styled.

`before_id` The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

`filter` An optional filter expression to subset features in the layer.

`cluster_options` A list of options for clustering symbols, created by the `cluster_options()` function.

Value

The modified map object with the new symbol layer added.

Examples

```
## Not run:
library(mapgl)
library(sf)
library(dplyr)

# Set seed for reproducibility
set.seed(1234)

# Define the bounding box for Washington DC (approximately)
bbox <- st_bbox(
  c(
    xmin = -77.119759,
    ymin = 38.791645,
    xmax = -76.909393,
    ymax = 38.995548
  ),
  crs = st_crs(4326)
)

# Generate 30 random points within the bounding box
random_points <- st_as_sf(
  data.frame(
    id = 1:30,
    lon = runif(30, bbox["xmin"], bbox["xmax"]),
    lat = runif(30, bbox["ymin"], bbox["ymax"])
  ),
  coords = c("lon", "lat"),
  crs = 4326
)

# Assign random icons
icons <- c("music", "bar", "theatre", "bicycle")
random_points <- random_points |>
  mutate(icon = sample(icons, n(), replace = TRUE))

# Map with icons
mapboxgl(style = mapbox_style("light")) |>
  fit_bounds(random_points, animate = FALSE) |>
  add_symbol_layer(
```

```

        id = "points-of-interest",
        source = random_points,
        icon_image = c("get", "icon"),
        icon_allow_overlap = TRUE,
        tooltip = "icon"
    )

## End(Not run)

```

add_vector_source *Add a vector tile source to a Mapbox GL or Maplibre GL map*

Description

Add a vector tile source to a Mapbox GL or Maplibre GL map

Usage

```
add_vector_source(map, id, url)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
url	A URL pointing to the vector tile source.

Value

The modified map object with the new source added.

add_video_source *Add a video source to a Mapbox GL or Maplibre GL map*

Description

Add a video source to a Mapbox GL or Maplibre GL map

Usage

```
add_video_source(map, id, urls, coordinates)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
urls	A vector of URLs pointing to the video sources.
coordinates	A list of coordinates specifying the video corners in clockwise order: top left, top right, bottom right, bottom left.

Value

The modified map object with the new source added.

carto_style	<i>Get CARTO Style URL</i>
-------------	----------------------------

Description

Get CARTO Style URL

Usage

```
carto_style(style_name)
```

Arguments

style_name The name of the style (e.g., "voyager", "positron", "dark-matter").

Value

The style URL corresponding to the given style name.

clear_controls	<i>Clear all controls from a Mapbox GL or Maplibre GL map in a Shiny app</i>
----------------	--

Description

Clear all controls from a Mapbox GL or Maplibre GL map in a Shiny app

Usage

```
clear_controls(map)
```

Arguments

map A map object created by the mapboxgl or maplibre function.

Value

The modified map object with all controls removed.

clear_layer	<i>Clear a layer from a map using a proxy</i>
-------------	---

Description

This function allows a layer to be removed from an existing Mapbox GL map using a proxy object.

Usage

```
clear_layer(proxy, layer_id)
```

Arguments

proxy	A proxy object created by <code>mapboxgl_proxy</code> or <code>maplibre_proxy</code> .
layer_id	The ID of the layer to be removed.

Value

The updated proxy object.

clear_legend	<i>Clear legend from a map in a proxy session</i>
--------------	---

Description

Clear legend from a map in a proxy session

Usage

```
clear_legend(map)
```

Arguments

map	A map object created by the <code>mapboxgl_proxy</code> or <code>maplibre_proxy</code> function.
-----	--

Value

The updated map object with the legend cleared.

clear_markers	<i>Clear markers from a map in a Shiny session</i>
---------------	--

Description

Clear markers from a map in a Shiny session

Usage

```
clear_markers(map)
```

Arguments

map A map object created by the `mapboxgl_proxy` or `maplibre_proxy` function.

Value

The modified map object with the markers cleared.

cluster_options	<i>Prepare cluster options for circle layers</i>
-----------------	--

Description

This function creates a list of options for clustering circle layers.

Usage

```
cluster_options(  
  max_zoom = 14,  
  cluster_radius = 50,  
  color_stops = c("#51bbd6", "#f1f075", "#f28cb1"),  
  radius_stops = c(20, 30, 40),  
  count_stops = c(0, 100, 750),  
  circle_blur = NULL,  
  circle_opacity = NULL,  
  circle_stroke_color = NULL,  
  circle_stroke_opacity = NULL,  
  circle_stroke_width = NULL,  
  text_color = "black"  
)
```

Arguments

<code>max_zoom</code>	The maximum zoom level at which to cluster points.
<code>cluster_radius</code>	The radius of each cluster when clustering points.
<code>color_stops</code>	A vector of colors for the circle color step expression.
<code>radius_stops</code>	A vector of radii for the circle radius step expression.
<code>count_stops</code>	A vector of point counts for both color and radius step expressions.
<code>circle_blur</code>	Amount to blur the circle.
<code>circle_opacity</code>	The opacity of the circle.
<code>circle_stroke_color</code>	The color of the circle's stroke.
<code>circle_stroke_opacity</code>	The opacity of the circle's stroke.
<code>circle_stroke_width</code>	The width of the circle's stroke.
<code>text_color</code>	The color to use for labels on the cluster circles.

Value

A list of cluster options.

Examples

```
cluster_options(
  max_zoom = 14,
  cluster_radius = 50,
  color_stops = c("#51bbd6", "#f1f075", "#f28cb1"),
  radius_stops = c(20, 30, 40),
  count_stops = c(0, 100, 750),
  circle_blur = 1,
  circle_opacity = 0.8,
  circle_stroke_color = "#ffffff",
  circle_stroke_width = 2
)
```

compare

Create a Compare slider widget

Description

This function creates a comparison view between two Mapbox GL or Maplibre GL maps, allowing users to swipe between the two maps to compare different styles or data layers.

Usage

```
compare(  
  map1,  
  map2,  
  width = "100%",  
  height = NULL,  
  elementId = NULL,  
  mousemove = FALSE,  
  orientation = "vertical"  
)
```

Arguments

map1	A mapboxgl or maplibre object representing the first map.
map2	A mapboxgl or maplibre object representing the second map.
width	Width of the map container.
height	Height of the map container.
elementId	An optional string specifying the ID of the container for the comparison. If NULL, a unique ID will be generated.
mousemove	A logical value indicating whether to enable swiping during cursor movement (rather than only when clicked).
orientation	A string specifying the orientation of the swiper, either "horizontal" or "vertical".

Value

A comparison widget.

Examples

```
## Not run:  
library(mapgl)  
  
library(mapgl)  
  
m1 <- mapboxgl(style = mapbox_style("light"))  
  
m2 <- mapboxgl(style = mapbox_style("dark"))  
  
compare(m1, m2)  
  
## End(Not run)
```

ease_to	<i>Ease to a given view</i>
---------	-----------------------------

Description

Ease to a given view

Usage

```
ease_to(map, center, zoom = NULL, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function or a proxy object.
center	A numeric vector of length 2 specifying the target center of the map (longitude, latitude).
zoom	The target zoom level.
...	Additional named arguments for easing to the view.

Value

The updated map object.

fit_bounds	<i>Fit the map to a bounding box</i>
------------	--------------------------------------

Description

Fit the map to a bounding box

Usage

```
fit_bounds(map, bbox, animate = FALSE, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function or a proxy object.
bbox	A bounding box specified as a numeric vector of length 4 (minLng, minLat, maxLng, maxLat), or an sf object from which a bounding box will be calculated.
animate	A logical value indicating whether to animate the transition to the new bounds. Defaults to FALSE.
...	Additional named arguments for fitting the bounds.

Value

The updated map object.

fly_to	<i>Fly to a given view</i>
--------	----------------------------

Description

Fly to a given view

Usage

```
fly_to(map, center, zoom = NULL, ...)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> function or a proxy object.
center	A numeric vector of length 2 specifying the target center of the map (longitude, latitude).
zoom	The target zoom level.
...	Additional named arguments for flying to the view.

Value

The updated map object.

get_column	<i>Get column or property for use in mapping</i>
------------	--

Description

This function returns an expression to get a specified column from a dataset (or a property from a layer).

Usage

```
get_column(column)
```

Arguments

column	The name of the column or property to get.
--------	--

Value

A list representing the expression to get the column.

get_drawn_features *Get drawn features from the map*

Description

Get drawn features from the map

Usage

```
get_drawn_features(map)
```

Arguments

map A map object created by the mapboxgl function, or a mapboxgl proxy.

Value

An sf object containing the drawn features.

Examples

```
## Not run:
# In a Shiny application
library(shiny)
library(mapgl)

ui <- fluidPage(
  mapboxglOutput("map"),
  actionButton("get_features", "Get Drawn Features"),
  verbatimTextOutput("feature_output")
)

server <- function(input, output, session) {
  output$map <- renderMapboxgl({
    mapboxgl(
      style = mapbox_style("streets"),
      center = c(-74.50, 40),
      zoom = 9
    ) |>
    add_draw_control()
  })

  observeEvent(input$get_features, {
    drawn_features <- get_drawn_features(mapboxgl_proxy("map"))
    output$feature_output <- renderPrint({
      print(drawn_features)
    })
  })
}
```

```
shinyApp(ui, server)

## End(Not run)
```

interpolate *Create an interpolation expression*

Description

This function generates an interpolation expression that can be used to style your data.

Usage

```
interpolate(
  column = NULL,
  property = NULL,
  type = "linear",
  values,
  stops,
  na_color = NULL
)
```

Arguments

column	The name of the column to use for the interpolation. If specified, property should be NULL.
property	The name of the property to use for the interpolation. If specified, column should be NULL.
type	The interpolation type. Can be one of "linear", <code>list("exponential", base)</code> where base specifies the rate at which the output increases, or <code>list("cubic-bezier", x1, y1, x2, y2)</code> where you define a cubic bezier curve with control points.
values	A numeric vector of values at which stops occur.
stops	A vector of corresponding stops (colors, sizes, etc.) for the interpolation.
na_color	The color to use for missing values. Mapbox GL JS defaults to black if this is not supplied.

Value

A list representing the interpolation expression.

Examples

```
interpolate(
  column = "estimate",
  type = "linear",
  values = c(1000, 200000),
  stops = c("#eff3ff", "#08519c")
)
```

jump_to	<i>Jump to a given view</i>
---------	-----------------------------

Description

Jump to a given view

Usage

```
jump_to(map, center, zoom = NULL, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function or a proxy object.
center	A numeric vector of length 2 specifying the target center of the map (longitude, latitude).
zoom	The target zoom level.
...	Additional named arguments for jumping to the view.

Value

The updated map object.

mapboxgl	<i>Initialize a Mapbox GL Map</i>
----------	-----------------------------------

Description

Initialize a Mapbox GL Map

Usage

```
mapboxgl(  
  style = NULL,  
  center = c(0, 0),  
  zoom = 0,  
  bearing = 0,  
  pitch = 0,  
  projection = "globe",  
  parallels = NULL,  
  access_token = NULL,  
  bounds = NULL,  
  width = "100%",  
  height = NULL,  
  ...  
)
```

Arguments

style	The Mapbox style to use.
center	A numeric vector of length 2 specifying the initial center of the map.
zoom	The initial zoom level of the map.
bearing	The initial bearing (rotation) of the map, in degrees.
pitch	The initial pitch (tilt) of the map, in degrees.
projection	The map projection to use (e.g., "mercator", "globe").
parallels	A vector of two numbers representing the standard parallels of the projection. Only available when the projection is "albers" or "lambertConformalConic".
access_token	Your Mapbox access token.
bounds	An sf object or bounding box to fit the map to.
width	The width of the output htmlwidget.
height	The height of the output htmlwidget.
...	Additional named parameters to be passed to the Mapbox GL map.

Value

An HTML widget for a Mapbox map.

Examples

```
## Not run:
mapboxgl(projection = "globe")

## End(Not run)
```

mapboxglOutput	<i>Create a Mapbox GL output element for Shiny</i>
----------------	--

Description

Create a Mapbox GL output element for Shiny

Usage

```
mapboxglOutput(outputId, width = "100%", height = "400px")
```

Arguments

outputId	The output variable to read from
width	The width of the element
height	The height of the element

Value

A Mapbox GL output element for use in a Shiny UI

mapboxgl_proxy	<i>Create a proxy object for a Mapbox GL map in Shiny</i>
----------------	---

Description

This function allows updates to be sent to an existing Mapbox GL map in a Shiny application without redrawing the entire map.

Usage

```
mapboxgl_proxy(mapId, session = shiny::getDefaultReactiveDomain())
```

Arguments

mapId	The ID of the map output element.
session	The Shiny session object.

Value

A proxy object for the Mapbox GL map.

mapbox_style	<i>Get Mapbox Style URL</i>
--------------	-----------------------------

Description

Get Mapbox Style URL

Usage

```
mapbox_style(style_name)
```

Arguments

style_name	The name of the style (e.g., "standard", "streets", "outdoors", etc.).
------------	--

Value

The style URL corresponding to the given style name.

maplibre	<i>Initialize a Maplibre GL Map</i>
----------	-------------------------------------

Description

Initialize a Maplibre GL Map

Usage

```
maplibre(  
  style = carto_style("voyager"),  
  center = c(0, 0),  
  zoom = 0,  
  bearing = 0,  
  pitch = 0,  
  bounds = NULL,  
  width = "100%",  
  height = NULL,  
  ...  
)
```

Arguments

style	The style JSON to use.
center	A numeric vector of length 2 specifying the initial center of the map.
zoom	The initial zoom level of the map.
bearing	The initial bearing (rotation) of the map, in degrees.
pitch	The initial pitch (tilt) of the map, in degrees.
bounds	An sf object or bounding box to fit the map to.
width	The width of the output htmlwidget.
height	The height of the output htmlwidget.
...	Additional named parameters to be passed to the Mapbox GL map.

Value

An HTML widget for a Mapbox map.

Examples

```
## Not run:  
maplibre()  
  
## End(Not run)
```

maplibreOutput	<i>Create a Maplibre GL output element for Shiny</i>
----------------	--

Description

Create a Maplibre GL output element for Shiny

Usage

```
maplibreOutput(outputId, width = "100%", height = "400px")
```

Arguments

outputId	The output variable to read from
width	The width of the element
height	The height of the element

Value

A Maplibre GL output element for use in a Shiny UI

maplibre_proxy	<i>Create a proxy object for a Maplibre GL map in Shiny</i>
----------------	---

Description

This function allows updates to be sent to an existing Maplibre GL map in a Shiny application without redrawing the entire map.

Usage

```
maplibre_proxy(mapId, session = shiny::getDefaultReactiveDomain())
```

Arguments

mapId	The ID of the map output element.
session	The Shiny session object.

Value

A proxy object for the Maplibre GL map.

maptiler_style	<i>Get MapTiler Style URL</i>
----------------	-------------------------------

Description

Get MapTiler Style URL

Usage

```
maptiler_style(style_name, api_key = NULL)
```

Arguments

style_name	The name of the style (e.g., "basic", "streets", "toner", etc.).
api_key	Your MapTiler API key (required)

Value

The style URL corresponding to the given style name.

match_expr	<i>Create a match expression</i>
------------	----------------------------------

Description

This function generates a match expression that can be used to style your data.

Usage

```
match_expr(column = NULL, property = NULL, values, stops, default = "#cccccc")
```

Arguments

column	The name of the column to use for the match expression. If specified, property should be NULL.
property	The name of the property to use for the match expression. If specified, column should be NULL.
values	A vector of values to match against.
stops	A vector of corresponding stops (colors, etc.) for the matched values.
default	A default value to use if no matches are found.

Value

A list representing the match expression.

Examples

```
match_expr(
  column = "category",
  values = c("A", "B", "C"),
  stops = c("#ff0000", "#00ff00", "#0000ff"),
  default = "#cccccc"
)
```

<code>move_layer</code>	<i>Move a layer to a different z-position</i>
-------------------------	---

Description

This function allows a layer to be moved to a different z-position in an existing Mapbox GL or Maplibre GL map using a proxy object.

Usage

```
move_layer(proxy, layer_id, before_id = NULL)
```

Arguments

<code>proxy</code>	A proxy object created by <code>mapboxgl_proxy</code> or <code>maplibre_proxy</code> .
<code>layer_id</code>	The ID of the layer to move.
<code>before_id</code>	The ID of an existing layer to insert the new layer before. Important: this means that the layer will appear <i>immediately behind</i> the layer defined in <code>before_id</code> . If omitted, the layer will be appended to the end of the layers array and appear above all other layers.

Value

The updated proxy object.

<code>on_section</code>	<i>Observe events on story map section transitions</i>
-------------------------	--

Description

For a given `story_section()`, you may want to trigger an event when the section becomes visible. This function wraps `shiny::observeEvent()` to allow you to modify the state of your map or invoke other Shiny actions on user scroll.

Usage

```
on_section(map_id, section_id, handler)
```

Arguments

map_id	The ID of your map output
section_id	The ID of the section to trigger on, defined in story_section()
handler	Expression to execute when section becomes visible.

renderMapboxgl	<i>Render a Mapbox GL output element in Shiny</i>
----------------	---

Description

Render a Mapbox GL output element in Shiny

Usage

```
renderMapboxgl(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

expr	An expression that generates a Mapbox GL map
env	The environment in which to evaluate expr
quoted	Is expr a quoted expression

Value

A rendered Mapbox GL map for use in a Shiny server

renderMaplibre	<i>Render a Maplibre GL output element in Shiny</i>
----------------	---

Description

Render a Maplibre GL output element in Shiny

Usage

```
renderMaplibre(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

expr	An expression that generates a Maplibre GL map
env	The environment in which to evaluate expr
quoted	Is expr a quoted expression

Value

A rendered Maplibre GL map for use in a Shiny server

set_config_property *Set a configuration property for a Mapbox GL map*

Description

Set a configuration property for a Mapbox GL map

Usage

```
set_config_property(map, import_id, config_name, value)
```

Arguments

map	A map object created by the mapboxgl function or a proxy object defined with mapboxgl_proxy().
import_id	The name of the imported style to set the config for (e.g., 'basemap').
config_name	The name of the configuration property from the style.
value	The value to set for the configuration property.

Value

The updated map object with the configuration property set.

set_filter *Set a filter on a map layer*

Description

This function sets a filter on a map layer, working with both regular map objects and proxy objects.

Usage

```
set_filter(map, layer_id, filter)
```

Arguments

map	A map object created by the mapboxgl or maplibre function, or a proxy object.
layer_id	The ID of the layer to which the filter will be applied.
filter	The filter expression to apply.

Value

The updated map object.

set_fog *Set fog on a Mapbox GL map*

Description

Set fog on a Mapbox GL map

Usage

```
set_fog(  
  map,  
  range = NULL,  
  color = NULL,  
  horizon_blend = NULL,  
  high_color = NULL,  
  space_color = NULL,  
  star_intensity = NULL  
)
```

Arguments

map	A map object created by the mapboxgl function or a proxy object.
range	A numeric vector of length 2 defining the minimum and maximum range of the fog.
color	A string specifying the color of the fog.
horizon_blend	A number between 0 and 1 controlling the blending of the fog at the horizon.
high_color	A string specifying the color of the fog at higher elevations.
space_color	A string specifying the color of the fog in space.
star_intensity	A number between 0 and 1 controlling the intensity of the stars in the fog.

Value

The updated map object.

set_layout_property *Set a layout property on a map layer*

Description

Set a layout property on a map layer

Usage

```
set_layout_property(map, layer, name, value)
```

Arguments

map	A map object created by the mapboxgl or maplibre function, or a proxy object.
layer	The ID of the layer to update.
name	The name of the layout property to set.
value	The value to set the property to.

Value

The updated map object.

set_paint_property *Set a paint property on a map layer*

Description

Set a paint property on a map layer

Usage

```
set_paint_property(map, layer, name, value)
```

Arguments

map	A map object created by the mapboxgl or maplibre function, or a proxy object.
layer	The ID of the layer to update.
name	The name of the paint property to set.
value	The value to set the property to.

Value

The updated map object.

set_projection	<i>Set Projection for a Mapbox/Maplibre Map</i>
----------------	---

Description

This function sets the projection dynamically after map initialization.

Usage

```
set_projection(map, projection)
```

Arguments

map	A map object created by mapboxgl() or maplibre() functions, or their respective proxy objects
projection	A string representing the projection name (e.g., "mercator", "globe", "albers", "equalEarth", etc.)

Value

The modified map object

set_source	<i>Set source of a map layer</i>
------------	----------------------------------

Description

Set source of a map layer

Usage

```
set_source(map, layer, source)
```

Arguments

map	A map object created by the mapboxgl or maplibre function, or a proxy object.
layer	The ID of the layer to update.
source	An sf object (which will be converted to a GeoJSON source).

Value

The updated map object.

set_style	<i>Update the style of a map</i>
-----------	----------------------------------

Description

Update the style of a map

Usage

```
set_style(map, style, config = NULL, diff = TRUE)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> function, or a proxy object.
style	The new style URL to be applied to the map.
config	A named list of options to be passed to the style config.
diff	A boolean that attempts a diff-based update rather than re-drawing the full style. Not available for all styles.

Value

The modified map object.

Examples

```
## Not run:
map <- mapboxgl(
  style = mapbox_style("streets"),
  center = c(-74.006, 40.7128),
  zoom = 10,
  access_token = "your_mapbox_access_token"
)

# Update the map style in a Shiny app
observeEvent(input$change_style, {
  mapboxgl_proxy("map", session) %>%
    set_style(mapbox_style("dark"), config = list(showLabels = FALSE), diff = TRUE)
})

## End(Not run)
```

set_terrain	<i>Set terrain properties on a map</i>
-------------	--

Description

Set terrain properties on a map

Usage

```
set_terrain(map, source, exaggeration = 1)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
source	The ID of the raster DEM source.
exaggeration	The terrain exaggeration factor.

Value

The modified map object with the terrain settings applied.

Examples

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

mapboxgl(
  style = mapbox_style("standard-satellite"),
  center = c(-114.26608, 32.7213),
  zoom = 14,
  pitch = 80,
  bearing = 41
) |>
add_raster_dem_source(
  id = "mapbox-dem",
  url = "mapbox://mapbox.mapbox-terrain-dem-v1",
  tileSize = 512,
  maxzoom = 14
) |>
set_terrain(
  source = "mapbox-dem",
  exaggeration = 1.5
)

## End(Not run)
```

set_tooltip	<i>Set tooltip on a map layer</i>
-------------	-----------------------------------

Description

Set tooltip on a map layer

Usage

```
set_tooltip(map, layer, tooltip)
```

Arguments

map	A map object created by the mapboxgl or maplibre function, or a proxy object.
layer	The ID of the layer to update.
tooltip	The name of the tooltip to set.

Value

The updated map object.

set_view	<i>Set the map center and zoom level</i>
----------	--

Description

Set the map center and zoom level

Usage

```
set_view(map, center, zoom)
```

Arguments

map	A map object created by the mapboxgl or maplibre function or a proxy object.
center	A numeric vector of length 2 specifying the center of the map (longitude, latitude).
zoom	The zoom level.

Value

The updated map object.

step_expr	<i>Create a step expression</i>
-----------	---------------------------------

Description

This function generates a step expression that can be used in your styles.

Usage

```
step_expr(column = NULL, property = NULL, base, values, stops, na_color = NULL)
```

Arguments

column	The name of the column to use for the step expression. If specified, property should be NULL.
property	The name of the property to use for the step expression. If specified, column should be NULL.
base	The base value to use for the step expression.
values	A numeric vector of values at which steps occur.
stops	A vector of corresponding stops (colors, sizes, etc.) for the steps.
na_color	The color to use for missing values. Mapbox GL JS defaults to black if this is not supplied.

Value

A list representing the step expression.

Examples

```
step_expr(  
  column = "value",  
  base = "#ffffff",  
  values = c(1000, 5000, 10000),  
  stops = c("#ff0000", "#00ff00", "#0000ff")  
)
```

story_leaflet	<i>Create a scrollytelling story map with Leaflet</i>
---------------	---

Description

Create a scrollytelling story map with Leaflet

Usage

```
story_leaflet(
  map_id,
  sections,
  root_margin = "-20% 0px -20% 0px",
  threshold = 0,
  styles = NULL,
  bg_color = "rgba(255,255,255,0.9)",
  text_color = "#34495e",
  font_family = NULL
)
```

Arguments

map_id	The ID of your mapboxgl, maplibre, or leaflet output defined in the server, e.g. "map"
sections	A named list of story_section objects. Names will correspond to map events defined within the server using on_section().
root_margin	The margin around the viewport for triggering sections by the intersection observer. Should be specified as a string, e.g. "-20% 0px -20% 0px".
threshold	A number that indicates the visibility ratio for a story ' panel to be used to trigger a section; should be a number between 0 and 1. Defaults to 0, meaning that the section is triggered as soon as the first pixel is visible.
styles	Optional custom CSS styles. Should be specified as a character string within shiny::tags\$style().
bg_color	Default background color for all sections
text_color	Default text color for all sections
font_family	Default font family for all sections

 story_map

Create a scrollytelling story map

Description

Create a scrollytelling story map

Usage

```
story_map(
  map_id,
  sections,
  map_type = c("mapboxgl", "maplibre", "leaflet"),
  root_margin = "-20% 0px -20% 0px",
  threshold = 0,
  styles = NULL,
  bg_color = "rgba(255,255,255,0.9)",
  text_color = "#34495e",
  font_family = NULL
)
```

Arguments

map_id	The ID of your mapboxgl, maplibre, or leaflet output defined in the server, e.g. "map"
sections	A named list of story_section objects. Names will correspond to map events defined within the server using on_section().
map_type	One of "mapboxgl", "maplibre", or "leaflet". This will use either mapboxglOutput(), maplibreOutput(), or leafletOutput() respectively, and must correspond to the appropriate render*() function used in the server.
root_margin	The margin around the viewport for triggering sections by the intersection observer. Should be specified as a string, e.g. "-20% 0px -20% 0px".
threshold	A number that indicates the visibility ratio for a story ' panel to be used to trigger a section; should be a number between 0 and 1. Defaults to 0, meaning that the section is triggered as soon as the first pixel is visible.
styles	Optional custom CSS styles. Should be specified as a character string within shiny::tags\$style().
bg_color	Default background color for all sections
text_color	Default text color for all sections
font_family	Default font family for all sections

story_maplibre	<i>Create a scrollytelling story map with MapLibre</i>
----------------	--

Description

Create a scrollytelling story map with MapLibre

Usage

```
story_maplibre(
  map_id,
  sections,
  root_margin = "-20% 0px -20% 0px",
  threshold = 0,
  styles = NULL,
  bg_color = "rgba(255,255,255,0.9)",
  text_color = "#34495e",
  font_family = NULL
)
```

Arguments

map_id	The ID of your mapboxgl, maplibre, or leaflet output defined in the server, e.g. "map"
sections	A named list of story_section objects. Names will correspond to map events defined within the server using on_section().
root_margin	The margin around the viewport for triggering sections by the intersection observer. Should be specified as a string, e.g. "-20% 0px -20% 0px".
threshold	A number that indicates the visibility ratio for a story ' panel to be used to trigger a section; should be a number between 0 and 1. Defaults to 0, meaning that the section is triggered as soon as the first pixel is visible.
styles	Optional custom CSS styles. Should be specified as a character string within shiny::tags\$style().
bg_color	Default background color for all sections
text_color	Default text color for all sections
font_family	Default font family for all sections

story_section	<i>Create a story section for story maps</i>
---------------	--

Description

Create a story section for story maps

Usage

```
story_section(  
  title,  
  content,  
  position = c("left", "center", "right"),  
  width = 400,  
  bg_color = "rgba(255,255,255,0.9)",  
  text_color = "#34495e",  
  font_family = NULL  
)
```

Arguments

title	Section title
content	Section content - can be text, HTML, or Shiny outputs
position	Position of text block ("left", "center", "right")
width	Width of text block in pixels (default: 400)
bg_color	Background color (with alpha) for text block
text_color	Text color
font_family	Font family for the section

Index

[add_categorical_legend](#), 3
[add_circle_layer](#), 5
[add_continuous_legend](#), 8
[add_draw_control](#), 9
[add_fill_extrusion_layer](#), 10
[add_fill_layer](#), 12
[add_fullscreen_control](#), 14
[add_geocoder_control](#), 15
[add_geolocate_control](#), 16
[add_globe_minimap](#), 17
[add_heatmap_layer](#), 18
[add_image](#), 20
[add_image_source](#), 22
[add_layer](#), 22
[add_layers_control](#), 24
[add_legend](#), 25
[add_line_layer](#), 27
[add_markers](#), 29
[add_navigation_control](#), 31
[add_raster_dem_source](#), 32
[add_raster_layer](#), 33
[add_raster_source](#), 35
[add_reset_control](#), 36
[add_scale_control](#), 36
[add_source](#), 37
[add_symbol_layer](#), 38
[add_vector_source](#), 44
[add_video_source](#), 44

[carto_style](#), 45
[clear_controls](#), 45
[clear_layer](#), 46
[clear_legend](#), 46
[clear_markers](#), 47
[cluster_options](#), 47
[compare](#), 48

[ease_to](#), 50

[fit_bounds](#), 50

[fly_to](#), 51

[get_column](#), 51
[get_drawn_features](#), 52

[interpolate](#), 53

[jump_to](#), 54

[mapbox_style](#), 56
[mapboxgl](#), 54
[mapboxgl_proxy](#), 56
[mapboxglOutput](#), 55
[maplibre](#), 57
[maplibre_proxy](#), 58
[maplibreOutput](#), 58
[maptiler_style](#), 59
[match_expr](#), 59
[move_layer](#), 60

[on_section](#), 60

[renderMapboxgl](#), 61
[renderMaplibre](#), 61

[set_config_property](#), 62
[set_filter](#), 62
[set_fog](#), 63
[set_layout_property](#), 63
[set_paint_property](#), 64
[set_projection](#), 65
[set_source](#), 65
[set_style](#), 66
[set_terrain](#), 67
[set_tooltip](#), 68
[set_view](#), 68
[step_expr](#), 69
[story_leaflet](#), 70
[story_map](#), 71
[story_maplibre](#), 72
[story_section](#), 73