ipyleaflet

Jul 29, 2020

Installation

1 Using pip **1**

2 Using conda **3**

3 JupyterLab extension **5**

4 Development installation **7**

5 Usage **9**

6 Map **11**
6.1 Usage . . . 11
6.2 Save to HTML . . . 12
6.3 Attributes . . . 13
6.4 Methods . . . 14

7 Basemaps **15**

8 Tile Layer **17**
8.1 Example . . . 17
8.2 Usage . . . 17
8.3 Attributes . . . 18

9 Local Tile Layer **19**
9.1 Example . . . 19
9.2 Attributes . . . 19

10 Marker **21**
10.1 Example . . . 21
10.2 Attributes . . . 22
10.3 Methods . . . 22

11 Icon **23**
11.1 Example . . . 23
11.2 Attributes . . . 23

12 AwesomeIcon **25**
12.1 Example . . . 25

12.2 Interactions . . . 26
12.3 Attributes . . . 26

13 Popup **27**
13.1 Example . . . 27
13.2 Attributes . . . 28

14 WMS Layer **29**
14.1 Example . . . 29
14.2 Advanced usage . . . 29
14.3 Attributes . . . 31

15 Image overlay and Video overlay **33**
15.1 Example ImageOverlay . . . 33
15.2 Example VideoOverlay . . . 33
15.3 Attributes . . . 34

16 AntPath **35**
16.1 Example . . . 35
16.2 Interactions . . . 36
16.3 Attributes . . . 36

17 Polyline **37**
17.1 Example Polyline . . . 37
17.2 Example MultiPolyline . . . 37
17.3 Attributes . . . 38

18 Polygon/Multipolygon **39**
18.1 Polygon . . . 39
18.2 Polygon with holes . . . 39
18.3 MultiPolygon . . . 40
18.4 Editable Polygon . . . 40
18.5 Attributes . . . 41

19 Rectangle **43**
19.1 Example . . . 43
19.2 Attributes . . . 43

20 Circle **45**
20.1 Example . . . 45
20.2 Attributes . . . 46

21 Circle Marker **47**
21.1 Example . . . 47
21.2 Attributes . . . 48

22 Marker Cluster **49**
22.1 Example . . . 49
22.2 Attributes . . . 49

23 Heatmap **51**
23.1 Example . . . 51
23.2 Attributes . . . 51

24 Velocity **53**
24.1 Example . . . 53

24.2 Attributes . . . 54

25 Layer Group **55**
25.1 Example . . . 55
25.2 Attributes . . . 56
25.3 Methods . . . 56

26 GeoJSON **57**
26.1 Example . . . 57
26.2 Usage . . . 58
26.3 Attributes . . . 58
26.4 Methods . . . 58

27 GeoData **59**
27.1 Example . . . 59
27.2 Attributes . . . 60

28 Choropleth **61**
28.1 Example . . . 61
28.2 Usage . . . 62
28.3 Attributes . . . 63

29 Vector Tile Layer **65**
29.1 Example . . . 65
29.2 Attributes . . . 67

30 Zoom Control **69**
30.1 Example . . . 69
30.2 Attributes . . . 69

31 Scale Control **71**
31.1 Example . . . 71
31.2 Attributes . . . 71

32 Layers Control **73**

33 Fullscreen Control **75**
33.1 Example . . . 75

34 Measure Control **77**
34.1 Example . . . 77
34.2 Attributes . . . 78
34.3 Methods . . . 78

35 SplitMap Control **79**
35.1 Example . . . 79
35.2 Attributes . . . 79

36 Draw Control **81**

37 Widget Control **83**
37.1 Example . . . 83
37.2 Attributes . . . 83

38 Legend Control **85**
38.1 Example . . . 85

38.2 Attributes 86

39 Search Control **87**
39.1 Example 87
39.2 Attributes 88

40 ipyleaflet related projects **89**

CHAPTER 1

Using pip

```
pip install ipyleaflet
jupyter nbextension enable --py --sys-prefix ipyleaflet  # can be skipped for␣
↪notebook 5.3 and above
```

CHAPTER 2

Using conda

```
conda install -c conda-forge ipyleaflet
```

CHAPTER 3

JupyterLab extension

If you have JupyterLab, you will also need to install the JupyterLab extension:

```
jupyter labextension install @jupyter-widgets/jupyterlab-manager jupyter-leaflet
```

CHAPTER 4

Development installation

For a development installation (requires npm):

```
git clone https://github.com/jupyter-widgets/ipyleaflet.git
cd ipyleaflet
pip install -e .
jupyter nbextension install --py --symlink --sys-prefix ipyleaflet
jupyter nbextension enable --py --sys-prefix ipyleaflet
jupyter labextension install @jupyter-widgets/jupyterlab-manager js  # If you are␣
↪developing on JupyterLab
```

Note for developers:

- the `-e` pip option allows one to modify the Python code in-place. Restart the kernel in order to see the changes.
- the `--symlink` argument on Linux or OS X allows one to modify the JavaScript code in-place. This feature is not available with Windows.

  For automatically building the JavaScript code every time there is a change, run the following command from the `ipyleaflet/js/` directory:

  ```
  npm run watch
  ```

  If you are on JupyterLab you also need to run the following in a separate terminal:

  ```
  jupyter lab --watch
  ```

  Every time a JavaScript build has terminated you need to refresh the Notebook page in order to load the JavaScript code again.

CHAPTER 5

Usage

ipyleaflet is an interactive widgets library, it is based on ipywidgets. This means that everything in ipyleaflet (e.g. the `Map`, `TileLayers`, `Markers...`) is interactive: you can dynamically update attributes from Python or from the Notebook interface.

For example, you can create a `Marker` layer and interact with it:

```
from ipyleaflet import Map, Marker

center = (52.204793, 360.121558)

m = Map(center=center, zoom=15)

marker = Marker(location=center, draggable=True)
m.add_layer(marker);

display(m)

# Now that the marker is on the Map, you can drag it with your mouse,
# it will automatically update the `marker.location` attribute in Python

# You can also update the marker location from Python, that will update the
# marker location on the Map:
marker.location = (50, 356)
```

ipywidgets is powered by traitlets, this brings an observer pattern implementation which allows you to react on widget attribute changes.

For example, you can define a Python callback that will be called whenever the marker location has changed:

```
def on_location_changed(event):
    # Do some computation given the new marker location, accessible from `event['new
↪']`
    pass

marker.observe(on_location_changed, 'location')
```

Please check out the traitlets documentation for more details about the observer pattern implementation.

Note: Everything in ipyleaflet **is** an interactive widget, from the `Map` class to `Layer` and `Control` classes. This means that what we achieved here with `marker.location`, you can achieve it with `map.zoom`, `layer.url`, or `heatmap.locations`

You can try ipyleaflet online using binder, no need to install anything on your computer:

CHAPTER 6

Map

6.1 Usage

```
from ipyleaflet import Map, basemaps, basemap_to_tiles

m = Map(
    basemap=basemap_to_tiles(basemaps.NASAGIBS.ModisTerraTrueColorCR, "2017-04-08"),
    center=(52.204793, 360.121558),
    zoom=4
)

m
```

You can find the list of available basemaps in the *Basemaps* page.

You can add multiple layers and controls to the map, using the `add_layer`/`add_control` methods. All those layers and controls are widgets themselves. So you can dynamically update their attributes from Python or by interacting with the map on the page (see *Usage*)

```
from ipyleaflet import Map, Marker, basemaps, basemap_to_tiles

m = Map(
    basemap=basemap_to_tiles(basemaps.NASAGIBS.ModisTerraTrueColorCR, "2017-04-08"),
    center=(52.204793, 360.121558),
    zoom=4
)

m.add_layer(Marker(location=(52.204793, 360.121558)))

m
```

6.2 Save to HTML

You can save the `Map` and all its layers and controls to an HTML page using the `save` method:

```
m.save('my_map.html', title='My Map')
```

Note: The saved file is a static HTML page, so there is no possible interaction with Python anymore. This means that all the Python callbacks you defined (*e.g.* on marker move) cannot be executed. If you want to serve the `Map` widget to an HTML page while keeping a Python kernel alive on the server, you might want to look at Voilà.

6.3 Attributes

Attribute	Default Value	Doc
layers	(default_layer)	Tuple of layers
controls	()	Tuple of controls
center	(0.0, 0.0)	Initial geographic center of the map
zoom	12	Initial map zoom level
max_zoom	18	
min_zoom	1	
zoom_snap	1	Forces the map's zoom level to always be a multiple of this
zoom_delta	1	Controls how much the map's zoom level will change after pressing + or - on the keyboard, or using the zoom controls
crs	projections.EPSG3857	Coordinate reference system, which can be 'Earth', 'EPSG3395', 'EPSG3857', 'EPSG4326', 'Base', 'Simple' or you can define your own projection. (See CustomProjections notebook)
dragging	True	Whether the map be draggable with mouse/touch or not
touch_zoom	True	Whether the map can be zoomed by touch-dragging with two fingers on mobile
scroll_wheel_zoom	False	Whether the map can be zoomed by using the mouse wheel
double_click_zoom	True	Whether the map can be zoomed in by double clicking on it and zoomed out by double clicking while holding shift
box_zoom	True	Whether the map can be zoomed to a rectangular area specified by dragging the mouse while pressing the shift key
tap	True	Enables mobile hacks for supporting instant taps
tap_tolerance	15	The max number of pixels a user can shift his finger during touch for it to be considered a valid tap
world_copy_jump	False	With this option enabled, the map tracks when you pan to another "copy" of the world and seamlessly jumps to
close_popup_on_click	True	Set it to False if you don't want popups to close when user clicks the map
bounce_at_zoomlimits	True	Set it to False if you don't want the map to zoom beyond min/max zoom and then bounce back when pinch-zooming
keyboard	True	Makes the map focusable and allows users to navigate the map with keyboard arrows and +/- keys
keyboard_pan_offset	80	
keyboard_zoom_offset	1	
inertia	True	If enabled, panning of the map will have an inertia effect
inertia_deceleration	3000	The rate with which the inertial movement slows down, in pixels/second^2
inertia_max_speed	1500	Max speed of the inertial movement, in pixels/second
zoom_control	True	
attribution_control	True	
zoom_animation_threshold	4	

6.4 Methods

Method	Arguments	Doc
add_layer	Layer instance	Add a new layer to the map
remove_layer	Layer instance	Remove a layer from the map
substitute_layer	Layer instance	Substitute a layer with a new layer
clear_layers		Remove all layers from the map
add_control	Control instance	Add a new control to the map
remove_control	Control instance	Remove a control from the map
clear_controls		Remove all controls from the map
on_interaction	callable object	Add a callback on interaction
save	output file	Save the map to an HTML file

CHAPTER 7

Basemaps

You can find on this page the default basemaps available in ipyleaflet, of course you can use another provider creating your own `TileLayer` layer.

Note: If one map on this page is completely grey, please report it by opening an issue: https://github.com/jupyter-widgets/ipyleaflet/issues/new

```
from ipyleaflet import Map, basemaps

center = [38.128, 2.588]
zoom = 5

Map(basemap=basemaps.OpenStreetMap.Mapnik, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.OpenStreetMap.BlackAndWhite, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.OpenStreetMap.France, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.OpenStreetMap.HOT, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.OpenTopoMap, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Hydda.Full, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Hydda.Base, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Esri.WorldStreetMap, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Esri.DeLorme, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Esri.WorldTopoMap, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Esri.WorldImagery, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Esri.NatGeoWorldMap, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.HikeBike.HikeBike, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.MtbMap, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.CartoDB.Positron, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.CartoDB.DarkMatter, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.NASAGIBS.ModisTerraTrueColorCR, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.NASAGIBS.ModisTerraBands367CR, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.NASAGIBS.ModisTerraBands721CR, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.NASAGIBS.ModisAquaTrueColorCR, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.NASAGIBS.ModisAquaBands721CR, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.NASAGIBS.ViirsTrueColorCR, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.NASAGIBS.ViirsEarthAtNight2012, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Strava.All, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Strava.Ride, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Strava.Run, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Strava.Water, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Strava.Winter, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Stamen.Terrain, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Stamen.Toner, center=center, zoom=zoom)
```

```
Map(basemap=basemaps.Stamen.Watercolor, center=center, zoom=zoom)
```

CHAPTER 8

Tile Layer

8.1 Example

```
from ipyleaflet import Map, basemaps, basemap_to_tiles

m = Map(center=(52.204793, 360.121558), zoom=9)

dark_matter_layer = basemap_to_tiles(basemaps.CartoDB.DarkMatter)
m.add_layer(dark_matter_layer)
m
```

8.2 Usage

Creating a `TileLayer` is straightforward, a dictionary containing basic tile layers is provided. This dictionary is named `basemaps`.

A `TileLayer` instance can be created using the `basemap_to_tiles` function, specifying the wanted map (e.g. `basemaps.CartoDB.DarkMatter`, `basemaps.Strava.Winter`, `basemaps.NASAGIBS.ModisTerraTrueColorCR`, ...).

Sometimes one could want to specify the date of the given images, for instance with NASA images:

```
nasa_layer = basemap_to_tiles(basemaps.NASAGIBS.ModisTerraTrueColorCR, "2018-04-08");
m.add_layer(nasa_layer);
```

8.3 Attributes

Attribute	Default Value
url	“https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png”
min_zoom	0
max_zoom	18
min_native_zoom	0
max_native_zoom	18
tile_size	256
attribution	“Map data (c) OpenStreetMap contributors”
detect_retina	False
opacity	1.0
visible	True
no_wrap	False
show_loading	False
loading	False (dynamically updated)

CHAPTER 9

Local Tile Layer

9.1 Example

```
from ipyleaflet import Map, LocalTileLayer

m = Map(center=(52.204793, 360.121558), zoom=9)
m.add_layer(LocalTileLayer(path='tiles/{z}/{x}/{y}.png'))

m
```

Note that the behavior is different in Jupyter Notebook and in JupyterLab.

In the classic Jupyter Notebook, the path is relative to the Notebook you are working on.

In JupyterLab, the path is relative to the server (where you started JupyterLab) and you need to prefix the path with "files/".

9.2 Attributes

Attribute	Default Value	Doc
path	""	Relative URL (e.g. 'tiles/{z}/{x}/{y}.png' or 'files/tiles/{z}/{x}/{y}.png' in JupyterLab)

CHAPTER 10

Marker

10.1 Example

```
from ipyleaflet import Map, Marker

center = (52.204793, 360.121558)

m = Map(center=center, zoom=15)

marker = Marker(location=center, draggable=False)
m.add_layer(marker);

m
```

10.2 Attributes

Attribute	Default Value	Doc
location	(0.0, 0.0)	
z_index_offset	0	
draggable	True	Whether the marker is draggable with mouse/touch or not
keyboard	True	Whether the marker can be tabbed to with a keyboard and clicked by pressing enter
title	“”	Text for the browser tooltip that appear on marker hover (no tooltip by default)
alt	“”	Text for the *alt* attribute of the icon image (useful for accessibility)
rise_on_hover	False	The z-index offset used for the *rise_on_hover* feature
opacity	1.0	
visible	True	
rise_offset	250	The z-index offset used for the *rise_on_hover* feature
rotation_angle	0	The rotation angle of the marker in degrees
rotation_origin	‘bottom center’	The rotation origin of the marker
icon	None	The icon for the marker

10.3 Methods

Method	Arguments	Doc
on_move	Callable object	Adds a callback on move event

CHAPTER 11

Icon

11.1 Example

```
from ipyleaflet import Marker, Icon, Map

center = (52.204793, 360.121558)

m = Map(center=center, zoom=10)
icon = Icon(icon_url='https://leafletjs.com/examples/custom-icons/leaf-green.png',␣
↪icon_size=[38, 95], icon_anchor=[22,94])
mark = Marker(location=center, icon=icon, rotation_angle=90, rotation_origin='22px␣
↪94px')
m.add_layer(mark);

m
```

11.2 Attributes

Attribute	Default Value	Doc
icon_url	''	url for icon
shadow_url	None	url for icon shadow
icon_size	(10, 10)	size icon will be rendered
shadow_size	(10, 10)	size icon shadow will be rendered
icon_anchor	(0, 0)	anchor point of icon
shadow_anchor	(0, 0)	anchor point of shadow
popup_anchor	(0, 0)	anchor point of popup

CHAPTER 12

AwesomeIcon

Font-Awesome icons for markers, see https://fontawesome.com/v4.7.0/icons for available icons.

12.1 Example

```
from ipyleaflet import AwesomeIcon, Marker, Map

center = (38.91342738235981, -77.03912909142674)

icon1 = AwesomeIcon(
    name='bus',
    marker_color='red',
    icon_color='black',
    spin=False
)

marker1 = Marker(icon=icon1, location=(center[0], center[1] - 0.05))

icon2 = AwesomeIcon(
    name='gear',
    marker_color='green',
    icon_color='darkgreen',
    spin=True
)

marker2 = Marker(icon=icon2, location=(center[0], center[1] + 0.05))

m = Map(center=center, zoom=13)

m.add_layer(marker1)
m.add_layer(marker2)

m
```

12.2 Interactions

Unlike other widgets in ipyleaflet, the `AwesomeIcon` widget is not dynamic. If you want to dynamically update the marker icon, you need to reassign the `Marker.icon` property with a new icon.

```
marker1.icon = AwesomeIcon(
    name='home',
    marker_color='blue',
    icon_color='black'
)
```

12.3 Attributes

Attribute	Default Value	Doc
name	'home'	Name of the Font-Awesome icon
marker_color	'blue'	Marker background color
icon_color	'white'	Icon color
spin	False	Whether the icon is spinning or not

CHAPTER 13

Popup

13.1 Example

```
from ipywidgets import HTML

from ipyleaflet import Map, Marker, Popup

center = (52.204793, 360.121558)

m = Map(center=center, zoom=9, close_popup_on_click=False)

marker = Marker(location=(52.1, 359.9))
m.add_layer(marker)

message1 = HTML()
message2 = HTML()
message1.value = "Try clicking the marker!"
message2.value = "Hello <b>World</b>"
message2.placeholder = "Some HTML"
message2.description = "Some HTML"

# Popup with a given location on the map:
popup = Popup(
    location=center,
    child=message1,
    close_button=False,
    auto_close=False,
    close_on_escape_key=False
)
m.add_layer(popup)

# Popup associated to a layer
marker.popup = message2
```

(continues on next page)

(continued from previous page)

```
m
```

13.2 Attributes

Attribute	Default Value	Doc
location	(0.0, 0.0)	
child		Content of the popup
max_width	300	Max width of the popup, in pixels
min_width	50	Min width of the popup, in pixels
max_height		If set, creates a scrollable container of the given height inside a popup if its content exceeds it
auto_pan	True	Set it to *False* if you don't want the map to do panning animation to fit the opened popup
auto_pan_padding	(5, 5)	
keep_in_view	False	Set it to *True* if you want to prevent users from panning the popup off of the screen while it is open
close_button	True	Controls the presence of a close button in the popup
close_on_escape_key	True	Set it to *False* if you want to override the default behavior of the ESC key for closing of the popup
class_name	""	A custom CSS class name to assign to the popup

CHAPTER 14

WMS Layer

14.1 Example

```
from ipyleaflet import Map, WMSLayer, basemaps

wms = WMSLayer(
    url='http://mesonet.agron.iastate.edu/cgi-bin/wms/nexrad/n0r.cgi',
    layers='nexrad-n0r-900913',
    format='image/png',
    transparent=True,
    attribution='Weather data © 2012 IEM Nexrad'
)

m = Map(basemap=basemaps.CartoDB.Positron, center=(38.491, -95.712), zoom=4)

m.add_layer(wms)

m
```

14.2 Advanced usage

By default, options like `layers`, `format`, `transparent` are passed in the request URL. If your tiles provider needs any extra parameter, you can define your own `WMSLayer` class which adds new parameters. For example, the following code adds a `time` parameter to the request by defining a custom `TimeWMSLayer`:

```
from traitlets import Unicode


class TimeWMSLayer(WMSLayer):

    time = Unicode('').tag(sync=True, o=True)
```

(continues on next page)

(continued from previous page)

```
time_wms = TimeWMSLayer(
    url='https://mesonet.agron.iastate.edu/cgi-bin/wms/nexrad/n0r-t.cgi?',
    layers='nexrad-n0r-wmst',
    time='2005-08-29T13:00:00Z',
    format='image/png',
    transparent=True,
    attribution='Weather data © 2012 IEM Nexrad'
)

m2 = Map(basemap=basemaps.CartoDB.Positron, center=(30.661, -88.645), zoom=5)

m2.add_layer(time_wms)

m2
```

Because it is a widget, you can dynamically update WMS parameters from Python manually:

```
# This will redraw the layer dynamically
time_wms.time = '2005-08-29T14:00'
```

Or from another widget like a slider: (Note that this example will not work in the documentation as there is no live Python kernel, but it will work in a Jupyter Notebook)

```
from ipywidgets import SelectionSlider

time_options = [
    '13:00', '13:30',
    '14:00', '14:30',
    '15:00', '15:30',
    '16:00', '16:30'
]

slider = SelectionSlider(description='Time:', options=time_options)

def update_wms(change):
    time_wms.time = '2005-08-29T{}'.format(slider.value)

slider.observe(update_wms, 'value')

slider
```

14.3 Attributes

Attribute	Default Value	Doc
url	"https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"	
min_zoom	0	
max_zoom	18	
tile_size	256	
attribution	"Map data (c) OpenStreetMap contributors"	
detect_retina	False	
opacity	1.0	
visible	True	
layers	""	Comma-separated list of WMS layers to show
styles	""	Comma-separated list of WMS styles
format	"image/jpeg"	WMS image format (use *'image/png'* for layers with transparency)
transparent	False	If *True*, the WMS service will return images with transparency
crs	ipyleaflet.projections.EPSG3857	Projection used for this service.

CHAPTER 15

Image overlay and Video overlay

15.1 Example ImageOverlay

```
from ipyleaflet import Map, ImageOverlay

m = Map(center=(25, -115), zoom=4)

image = ImageOverlay(
    url="https://i.imgur.com/06Q1fSz.png",
    # url='../06Q1fSz.png',
    bounds=((13, -130), (32, -100))
)

m.add_layer(image);
m
```

15.2 Example VideoOverlay

```
from ipyleaflet import Map, VideoOverlay

m = Map(center=(25, -115), zoom=4)

video = VideoOverlay(
    url="https://www.mapbox.com/bites/00188/patricia_nasa.webm",
    bounds=((13, -130), (32, -100))
)

m.add_layer(video);
m
```

15.3 Attributes

Attribute	Default Value	Doc
url	“”	An http url to the footage or a relative path to a local file (image/video). Note that absolute local paths are not supported.
bounds	((0.0, 0.0), (0.0, 0.0))	SW and NE corners of the image

CHAPTER 16

AntPath

16.1 Example

```
from ipyleaflet import Map, AntPath

m = Map(center=(51.332, 6.853), zoom=10)

ant_path = AntPath(
    locations=[
        [51.185, 6.773], [51.182, 6.752], [51.185, 6.733], [51.194, 6.729],
        [51.205, 6.732], [51.219, 6.723], [51.224, 6.723], [51.227, 6.728],
        [51.228, 6.734], [51.226, 6.742], [51.221, 6.752], [51.221, 6.758],
        [51.224, 6.765], [51.230, 6.768], [51.239, 6.765], [51.246, 6.758],
        [51.252, 6.745], [51.257, 6.724], [51.262, 6.711], [51.271, 6.701],
        [51.276, 6.702], [51.283, 6.710], [51.297, 6.725], [51.304, 6.732],
        [51.312, 6.735], [51.320, 6.734], [51.326, 6.726], [51.334, 6.713],
        [51.340, 6.696], [51.344, 6.678], [51.349, 6.662], [51.354, 6.655],
        [51.360, 6.655], [51.366, 6.662], [51.369, 6.675], [51.373, 6.704],
        [51.376, 6.715], [51.385, 6.732], [51.394, 6.741], [51.402, 6.743],
        [51.411, 6.742], [51.420, 6.733], [51.429, 6.718], [51.439, 6.711],
        [51.448, 6.716], [51.456, 6.724], [51.466, 6.719], [51.469, 6.713],
        [51.470, 6.701], [51.473, 6.686], [51.479, 6.680], [51.484, 6.680],
        [51.489, 6.685], [51.493, 6.700], [51.497, 6.714]
    ],
    dash_array=[1, 10],
    delay=1000,
    color='#7590ba',
    pulse_color='#3f6fba'
)

m.add_layer(ant_path)

m
```

16.2 Interactions

Like most widgets in ipyleaflet, the `AntPath` can be dynamically updated from Python.

```
# Update the color
ant_path.color = 'red'

# Update the path
ant_path.locations = [[51.185, 6.773], [51.326, 6.726], [51.497, 6.714]]
```

16.3 Attributes

Attribute	Default Value	Doc
locations	[]	List of path points as (lat, lng) couples
color	'#0000FF'	Background color for the path
pulse_color	'#FFFFFF'	Color of the moving ants on the path
paused	False	Whether the ants are moving or not
reverse	False	Whether the ants are moving in reverse or not
use	'polyline'	Which shape is drawn, possible values are 'polyline', 'polygon', 'rectangle' and 'circle'
dash_array	[10, 20]	Dash pattern for lines as a list of non-negative numbers
weight	5	Lines weight
delay	400	Ants speed
radius	10	Radius of the circle, if `use` is set to 'circle'

CHAPTER 17

Polyline

17.1 Example Polyline

```
from ipyleaflet import Map, Polyline

line = Polyline(
    locations=[
        [45.51, -122.68],
        [37.77, -122.43],
        [34.04, -118.22]
    ],
    color="green" ,
    fill=False
)

m = Map(center = (42.5, -41), zoom =2)
m.add_layer(line)
m
```

17.2 Example MultiPolyline

```
from ipyleaflet import Map, Polyline

lines = Polyline(
    locations=[
        [[45.51, -122.68],
         [37.77, -122.43],
         [34.04, -118.2]],
        [[40.78, -73.91],
         [41.83, -87.62],
         [32.76, -96.72]]
```

(continues on next page)

(continued from previous page)

```
    ],
    color="green" ,
    fill=False
)

m = Map(center = (42.5, -41), zoom =2)
m.add_layer(lines)
m
```

17.3 Attributes

Attribute	Default Value	Doc
locations	[[]]	List of list of points of the polygon
stroke	True	Set it to *False* to disable borders
color	"#0033FF"	Stroke color
opacity	1.0	Stroke opacity
weight	5	Stroke width in pixels
fill	True	Whether to fill the polyline or not
fill_color	None	
fill_opacity	0.2	
dash_array		
line_cap	"round"	
line_join	"round"	

CHAPTER 18

Polygon/Multipolygon

18.1 Polygon

You can easily create a `Polygon` providing the list of vertex locations (in lat/lng).

```
from ipyleaflet import Map, Polygon

polygon = Polygon(
    locations=[(42, -49), (43, -49), (43, -48)],
    color="green",
    fill_color="green"
)

m = Map(center=(42.5531, -48.6914), zoom=6)
m.add_layer(polygon);

m
```

Because the `Polygon` an interactive widget, you can dynamically update the locations/color from Python, and you will see updated on the `Map`.

18.2 Polygon with holes

You can define holes in your `Polygon` by using nested lists of vertex locations.

```
from ipyleaflet import Map, Polygon

polygon = Polygon(
    locations= [
        [(37, -109.05), (41, -109.03), (41, -102.05), (37, -102.04)],
        [(37.29, -108.58), (40.71, -108.58), (40.71, -102.50), (37.29, -102.50)]
    ],
```

(continues on next page)

(continued from previous page)

```
    color="green",
    fill_color="green"
)

m = Map(center=(37.5531, -109.6914), zoom=5)
m.add_layer(polygon);

m
```

18.3 MultiPolygon

```
from ipyleaflet import Map, Polygon

multipolygon = Polygon(
    locations=[
        [(42, -49), (43, -49), (43, -48)],
        [(44,-49), (43, -50), (44,-50)]
    ],
    color="green",
    fill_color="green"
)

m = Map(center=(42.5531, -48.6914), zoom=6)
m.add_layer(multipolygon);

m
```

18.4 Editable Polygon

If `transform` is set to `True`, you can dynamically edit the polygon with the mouse.

```
from ipyleaflet import Map, Polygon

polygon = Polygon(
    locations=[(42, -49), (43, -49), (43, -48)],
    color="green",
    fill_color="green",
    transform=True
)

m = Map(center=(42.5531, -48.6914), zoom=6)
m.add_layer(polygon);

m
```

18.5 Attributes

Attribute	Default Value	Doc
locations	[]	List of points of the polygon
stroke	True	Set it to *False* to disable borders
color	"#0033FF"	Stroke color
opacity	1.0	Stroke opacity
weight	5	Stroke width in pixels
fill	True	Whether to fill the polygon or not
fill_color	None	If None, it will be the same as the color value
fill_opacity	0.2	
dash_array		
line_cap	"round"	
line_join	"round"	
transform	False	Whether the polygon is editable with the mouse or not
scaling	True	Whether the polygon scale is editable or not, needs transform set to `True`
rotation	True	Whether the polygon rotation is editable or not, needs transform set to `True`
uniform_scaling	False	Whether to keep the scale ratio when editing the scale, needs transform set to `True`

CHAPTER 19

Rectangle

19.1 Example

```
from ipyleaflet import Map, basemaps, basemap_to_tiles, Rectangle

watercolor = basemap_to_tiles(basemaps.Stamen.Watercolor)

m = Map(layers=(watercolor, ), center=(53, 354), zoom=5)

rectangle = Rectangle(bounds=((52, 354), (53, 360)))

m.add_layer(rectangle)

m
```

19.2 Attributes

Attribute	Default Value	Doc
bounds	()	SW and NE corners of the rectangle
stroke	True	Set it to *False* to disable borders
color	"#0033FF"	Stroke color
opacity	1.0	Stroke opacity
weight	5	Stroke width in pixels
fill	True	Whether to fill the polygon or not
fill_color	None	
fill_opacity	0.2	
dash_array		
line_cap	"round"	
line_join	"round"	

CHAPTER 20

Circle

20.1 Example

```
from ipyleaflet import Map, basemaps, basemap_to_tiles, Circle

watercolor = basemap_to_tiles(basemaps.Stamen.Watercolor)

m = Map(layers=(watercolor, ), center=(53, 354), zoom=5)

circle = Circle()
circle.location = (50, 354)
circle.radius = 50000
circle.color = "green"
circle.fill_color = "green"

m.add_layer(circle)

m
```

20.2 Attributes

Attribute	Default Value	Doc
location	(0.0, 0.0)	Circle location
radius	10	Circle radius in meters
stroke	True	Set it to *false* to disable borders
color	"#0033FF"	Stroke color
opacity	1.0	Stroke opacity
weight	5	Stroke width in pixels
fill	True	Whether to fill the circle or not
fill_color	None	
fill_opacity	0.2	
dash_array		
line_cap	"round"	
line_join	"round"	

CHAPTER 21

Circle Marker

21.1 Example

```
from ipyleaflet import Map, basemaps, basemap_to_tiles, CircleMarker

watercolor = basemap_to_tiles(basemaps.Stamen.Watercolor)

m = Map(layers=(watercolor, ), center=(53, 354), zoom=5)

circle_marker = CircleMarker()
circle_marker.location = (55, 360)
circle_marker.radius = 50
circle_marker.color = "red"
circle_marker.fill_color = "red"

m.add_layer(circle_marker)

m
```

21.2 Attributes

Attribute	Default Value	Doc
location	(0.0, 0.0)	Circle location
radius	10	Circle radius in pixels
stroke	True	Set it to *false* to disable borders
color	"#0033FF"	Stroke color
opacity	1.0	Stroke opacity
weight	5	Stroke width in pixels
fill	True	Whether to fill the circle or not
fill_color	None	
fill_opacity	0.2	
dash_array		
line_cap	"round"	
line_join	"round"	

CHAPTER 22

Marker Cluster

22.1 Example

```
from ipyleaflet import Map, Marker, MarkerCluster

m = Map(center=(50, 0), zoom=5)

marker1 = Marker(location=(48, -2))
marker2 = Marker(location=(50, 0))
marker3 = Marker(location=(52, 2))

marker_cluster = MarkerCluster(
    markers=(marker1, marker2, marker3)
)

m.add_layer(marker_cluster);

m
```

22.2 Attributes

Attribute	Default Value	Doc
markers	()	Tuple of markers

CHAPTER 23

Heatmap

23.1 Example

```
from ipyleaflet import Map, Heatmap
from random import uniform
m = Map(center=(0, 0), zoom=2)

heatmap = Heatmap(
    locations=[[uniform(-80, 80), uniform(-180, 180), uniform(0, 1000)] for i in␣
↪range(1000)],
    radius=20
)

m.add_layer(heatmap);

m
```

23.2 Attributes

Attribute	Default Value	Doc
locations	[]	List of center locations
min_opacity	0.05	Minimum opacity the heat will start at
max_zoom	18	Zoom level where max intensity is reached
max	1.0	Maximum point intensity
radius	25.0	Radius of each “point” of the heatmap
blur	15.0	Amount of blur
gradient	{0.4: ‘blue’, 0.6: ‘cyan’, 0.7: ‘lime’, 0.8: ‘yellow’, 1.0: ‘red’}	Color gradient config

CHAPTER 24

Velocity

24.1 Example

```
from ipyleaflet import Map, TileLayer, basemaps
from ipyleaflet.velocity import Velocity
import xarray as xr
import os

if not os.path.exists('wind-global.nc'):
  url = 'https://github.com/benbovy/xvelmap/raw/master/notebooks/wind-global.nc'
  import requests
  r = requests.get(url)
  wind_data = r.content
  with open('wind-global.nc', 'wb') as f:
      f.write(wind_data)

center = [0, 0]
zoom = 1
m = Map(center=center, zoom=zoom, interpolation='nearest', basemap=basemaps.CartoDB.
↪DarkMatter)

ds = xr.open_dataset('wind-global.nc')
display_options = {
    'velocityType': 'Global Wind',
    'displayPosition': 'bottomleft',
    'displayEmptyString': 'No wind data'
}
wind = Velocity(data=ds,
                zonal_speed='u_wind',
                meridional_speed='v_wind',
                latitude_dimension='lat',
                longitude_dimension='lon',
                velocity_scale=0.01,
                max_velocity=20,
```

(continues on next page)

(continued from previous page)

```
                display_options=display_options)
m.add_layer(wind)

m
```

24.2 Attributes

Attribute	Default Value	Doc
data	Empty dataset	Underlying dataset
zonal_speed	''	Variable name in underlying dataset for the zonal speed
meridional_speed	''	Variable name in underlying dataset for the meridional speed
latitude_dimension	'latitude'	Name of the latitude dimension in underlying dataset
longitude_dimension	'longitude'	Name of the longitude dimension in underlying dataset
units	None	Units
display_values	True	Display velocity data on mouse hover
display_options	{}	Display options
min_velocity	0.0	Used to align color scale
max_velocity	10.0	Used to align color scale
velocity_scale	0.005	Modifier for particle animations
color_scale	[]	Array of hex/rgb colors for user-specified color scale.

CHAPTER 25

Layer Group

25.1 Example

```
from ipyleaflet import (
    Map, basemaps, basemap_to_tiles,
    Circle, Marker, Rectangle, LayerGroup
)

toner = basemap_to_tiles(basemaps.Stamen.Toner)

m = Map(layers=(toner, ), center=(50, 354), zoom=5)

# Create some layers
marker = Marker(location=(50, 354))
circle = Circle(location=(50, 370), radius=50000, color="yellow", fill_color="yellow")
rectangle = Rectangle(bounds=((54, 354), (55, 360)), color="orange", fill_color=
↪"orange")

# Create layer group
layer_group = LayerGroup(layers=(marker, circle))

m.add_layer(layer_group)

layer_group.add_layer(rectangle)

layer_group.remove_layer(circle)

m
```

25.2 Attributes

Attribute	Default Value	Doc
layers	()	List of layers

25.3 Methods

Method	Arguments	Doc
add_layer	Layer instance	Add a new layer to the group
remove_layer	Layer instance	Remove a layer from the group
clear_layers		Remove all layers from the group

CHAPTER 26

GeoJSON

26.1 Example

```
import os
import json
import random
import requests

from ipyleaflet import Map, GeoJSON

if not os.path.exists('europe_110.geo.json'):
    url = 'https://github.com/jupyter-widgets/ipyleaflet/raw/master/examples/europe_
↪110.geo.json'
    r = requests.get(url)
    with open('europe_110.geo.json', 'w') as f:
        f.write(r.content.decode("utf-8"))

with open('europe_110.geo.json', 'r') as f:
    data = json.load(f)

def random_color(feature):
    return {
        'color': 'black',
        'fillColor': random.choice(['red', 'yellow', 'green', 'orange']),
    }

m = Map(center=(50.6252978589571, 0.34580993652344), zoom=3)

geo_json = GeoJSON(
    data=data,
    style={
        'opacity': 1, 'dashArray': '9', 'fillOpacity': 0.1, 'weight': 1
    },
    hover_style={
```

(continues on next page)

(continued from previous page)

```
        'color': 'white', 'dashArray': '0', 'fillOpacity': 0.5
    },
    style_callback=random_color
)
m.add_layer(geo_json)

m
```

26.2 Usage

The `GeoJSON` layer is a widget, which means that you can update the `data` or any other attribute from Python and it will dynamically update the map:

```
geo_json.data = new_data
geo_json.hover_style = new_hover_style
```

26.3 Attributes

Attribute	Doc
data	Data dictionary
style	Style dictionary
hover_style	Hover style dictionary
style_callback	Styling function that is called for each feature, and should return the feature style. This styling function takes the feature as argument.

26.4 Methods

Method	Arguments	Doc
on_click	Callable object	Adds a callback on click event
on_hover	Callable object	Adds a callback on hover event

CHAPTER 27

GeoData

`GeoData` is an `ipyleaflet` class that allows you to visualize a GeoDataFrame on the Map.

27.1 Example

```
from ipyleaflet import Map, GeoData, basemaps, LayersControl
import geopandas
import json

countries = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowres'))
rivers = geopandas.read_file("https://www.naturalearthdata.com/http//www.
↪naturalearthdata.com/download/10m/physical/ne_10m_rivers_lake_centerlines.zip")

m = Map(center=(52.3,8.0), zoom = 3, basemap= basemaps.Esri.WorldTopoMap)

geo_data = GeoData(geo_dataframe = countries,
                   style={'color': 'black', 'fillColor': '#3366cc', 'opacity':0.05,
↪'weight':1.9, 'dashArray':'2', 'fillOpacity':0.6},
                   hover_style={'fillColor': 'red' , 'fillOpacity': 0.2},
                   name = 'Countries')

rivers_data = GeoData(geo_dataframe = rivers,
                   style={'color': 'purple', 'opacity':3, 'weight':1.9, 'dashArray':'2
↪', 'fillOpacity':0.6},
                   hover_style={'fillColor': 'red' , 'fillOpacity': 0.2},
                   name = 'Rivers')

m.add_layer(rivers_data)
m.add_layer(geo_data)
m.add_control(LayersControl())

m
```

27.2 Attributes

Attribute	Doc	Description
geo_data	Data dictionary	GeoDataFrame
style	Style dictionary	
hover_style	Hover style dictionary	

CHAPTER 28

Choropleth

28.1 Example

```
import ipyleaflet
import json
import pandas as pd
import os
import requests
from ipywidgets import link, FloatSlider
from branca.colormap import linear

def load_data(url, filename, file_type):
    r = requests.get(url)
    with open(filename, 'w') as f:
        f.write(r.content.decode("utf-8"))
    with open(filename, 'r') as f:
        return file_type(f)

geo_json_data = load_data(
    'https://raw.githubusercontent.com/jupyter-widgets/ipyleaflet/master/examples/us-
↪states.json',
    'us-states.json',
     json.load)

unemployment = load_data(
    'https://raw.githubusercontent.com/jupyter-widgets/ipyleaflet/master/examples/US_
↪Unemployment_Oct2012.csv',
    'US_Unemployment_Oct2012.csv',
     pd.read_csv)

unemployment =  dict(zip(unemployment['State'].tolist(), unemployment['Unemployment'].
↪tolist()))

layer = ipyleaflet.Choropleth(
```

(continues on next page)

(continued from previous page)

```
    geo_data=geo_json_data,
    choro_data=unemployment,
    colormap=linear.YlOrRd_04,
    border_color='black',
    style={'fillOpacity': 0.8, 'dashArray': '5, 5'})

m = ipyleaflet.Map(center = (43,-100), zoom = 4)
m.add_layer(layer)
m
```

28.2 Usage

The `Choropleth` takes `geo_data` and `choro_data` as arguments.

The `geo_data` is a GeoJSON dictionary, for instance :

```
{
    "type": "FeatureCollection",
    "features":[{
        "type":"Feature",
        "id":"AL",
        "properties":{"name":"Alabama"},
        "geometry":{
            "type":"Polygon",
            "coordinates": [[[-87.359296,35.00118]]] ...
        }
    }]
}
```

The `choro_data` is a dictionary that maps an key to a float value, in order to build the colormap :

```
{'AL': 7.1,
 'AK': 6.8}
```

The `Choropleth` layer is then created specifying on which key the colormap is applied:

```
Choropleth(
    geo_data=geo_data,
    choro_data=choro_data,
    key_on='id'
)
```

28.3 Attributes

Attribute	Default	Doc
geo_data	{}	Data dictionary
choro_data	{}	Mapping key -> float data for constructing the colormap
key_on	'id'	Key used for the colormap construction
value_min		Color scale minimum value
value_max		Color scale maximum value
colormap	OrRd_06	Map of color from branca
style		Style dictionary
hover_style		Hover style dictionary
style_callback		Styling function that is called for each feature, and should return the feature style. This styling function takes the feature, the colormap function and the key data as arguments.

CHAPTER 29

Vector Tile Layer

29.1 Example

```
from ipyleaflet import Map, VectorTileLayer

from traitlets import Unicode, Dict

# This is a custom VectorTileLayer subclass, allowing to pass our api key to the url
class CustomVectorTileLayer(VectorTileLayer):
    api_key = Unicode('gCZXZglvRQa6sB2z7JzL1w').tag(sync=True, o=True)

water_style = dict(
fill="true",
weight=1,
fillColor="#06cccc",
color="#06cccc",
fillOpacity=0.2,
opacity=0.4,
)

waterway_style = dict(
    weight=1, fillColor="#2375e0", color="#2375e0", fillOpacity=0.2, opacity=0.4
)

admin_style = dict(
    weight=1, fillColor="pink", color="pink", fillOpacity=0.2, opacity=0.4
)

landcover_style = dict(
    fill="true",
    weight=1,
    fillColor="#53e033",
    color="#53e033",
    fillOpacity=0.2,
```

(continues on next page)

(continued from previous page)

```
    opacity=0.4,
)

landuse_style = dict(
    fill="true",
    weight=1,
    fillColor="#e5b404",
    color="#e5b404",
    fillOpacity=0.2,
    opacity=0.4,
)

park_style = dict(
    fill="true",
    weight=1,
    fillColor="#84ea5b",
    color="#84ea5b",
    fillOpacity=0.2,
    opacity=0.4,
)

boundary_style = dict(
    weight=1, fillColor="#c545d3", color="#c545d3", fillOpacity=0.2, opacity=0.4
)


aeroway = dict(
    weight=1, fillColor="#51aeb5", color="#51aeb5", fillOpacity=0.2, opacity=0.4
)

road = dict(
    weight=1, fillColor="#f2b648", color="#f2b648", fillOpacity=0.2, opacity=0.4
)

transit = dict(
    weight=0.5, fillColor="#f2b648", color="#f2b648", fillOpacity=0.2, opacity=0.4
)

buildings = dict(
    fill="true",
    weight=1,
    fillColor="#2b2b2b",
    color="#2b2b2b",
    fillOpacity=0.2,
    opacity=0.4,
)

water_name = dict(
    weight=1, fillColor="#022c5b", color="#022c5b", fillOpacity=0.2, opacity=0.4
)

transportation_name = dict(
    weight=1, fillColor="#bc6b38", color="#bc6b38", fillOpacity=0.2, opacity=0.4
)

place = dict(
    weight=1, fillColor="#f20e93", color="#f20e93", fillOpacity=0.2, opacity=0.4
```

(continues on next page)

(continued from previous page)

```
)

housenumber = dict(
    weight=1, fillColor="#ef4c8b", color="#ef4c8b", fillOpacity=0.2, opacity=0.4
)

poi = dict(weight=1, fillColor="#3bb50a", color="#3bb50a", fillOpacity=0.2, opacity=0.
↪4)

earth = dict(
    fill="true",
    weight=1,
    fillColor="#c0c0c0",
    color="#c0c0c0",
    fillOpacity=0.2,
    opacity=0.4,
)

url = 'https://tile.nextzen.org/tilezen/vector/v1/512/all/{z}/{x}/{y}.mvt?api_key=
↪{apiKey}'
vector_tile_layer_styles = dict(
    water=water_style,
    waterway=waterway_style,
    admin=admin_style,
    andcover=landcover_style,
    landuse=landuse_style,
    park=park_style,
    boundaries=boundary_style,
    aeroway=aeroway,
    roads=road,
    transit=transit,
    buildings=buildings,
    water_name=water_name,
    transportation_name=transportation_name,
    places=place,
    housenumber=housenumber,
    pois=poi,
    earth=earth
)

m = Map(center=(52.204793, 360.121558), zoom=9)
vl = CustomVectorTileLayer(url=url, vector_tile_layer_styles=vector_tile_layer_styles)
m.add_layer(vl)
m
```

29.2 Attributes

Attribute	Default Value	Doc
url	''	Url for the source protobuf data.
attribution	'Map data (c) OpenStreetMap contributors'	Attribution for the map.
vector_tile_layer_styles	{}	Styles for the various data layer of protobuf layers.

CHAPTER 30

Zoom Control

30.1 Example

```
from ipyleaflet import Map, ZoomControl

m = Map(zoom=5, center=[51.64, -76.52], zoom_control=False)  # Do not automatically␣
↪create a ZoomControl
m.add_control(ZoomControl(position='topright'))

m
```

30.2 Attributes

Attribute	Default Value	Doc
position	'topleft'	Position of the control, can be 'bottomleft', 'bottomright', 'topleft', or 'topright'
zoom_in_text	'+'	Text to show in the "zoom in" button
zoom_in_title	'Zoom in'	Text to show on mouse hover on the "zoom in" button
zoom_out_text	'-'	Text to show in the "zoom out" button
zoom_out_title	'Zoom out'	Text to show on mouse hover on the "zoom out" button

CHAPTER 31

Scale Control

31.1 Example

```
from ipyleaflet import Map, ScaleControl

m = Map(zoom=5, center=[51.64, -76.52])
m.add_control(ScaleControl(position='bottomleft'))

m
```

31.2 Attributes

Attribute	Default Value	Doc
position	'topleft'	Position of the control, can be 'bottomleft', 'bottomright', 'topleft', or 'topright'
max_width	100	Maximum width of the control in pixels. The width is set dynamically to show round values (e.g. 100, 200, 500).
metric	True	Whether to show the metric scale line (m/km).
imperial	True	Whether to show the imperial scale line (mi/ft).
update_when_idle	False	If true, the control is updated only after ending dragging the Map, otherwise it's always up-to-date (updated during move).

CHAPTER 32

Layers Control

The `LayersControl` allows one to display a layer selector on the map in order to select which layers to display on the map.

All layers have a `name` attribute which is displayed in the selector and can be changed by the user.

```
from ipyleaflet import Map, Marker, LayersControl

m = Map(center=(50, 0), zoom=5)

marker1 = Marker(name='marker1', location=(48, -2))
marker2 = Marker(name='marker2', location=(50, 0))
marker3 = Marker(name='marker3', location=(52, 2))
m.add_layer(marker1)
m.add_layer(marker2)
m.add_layer(marker3)

control = LayersControl(position='topright')
m.add_control(control)

m
```

CHAPTER 33

Fullscreen Control

33.1 Example

```
from ipyleaflet import Map, FullScreenControl

m = Map(zoom=5, center=[51.64, -76.52])
m.add_control(FullScreenControl())

m
```

Attribute	Default Value	Doc
position	'topleft'	Position of the control, can be 'bottomleft', 'bottomright', 'topleft', or 'topright'

CHAPTER 34

Measure Control

34.1 Example

```
from ipyleaflet import Map, MeasureControl, basemaps

m = Map(center=(43.0327, 6.0232), zoom=9, basemap=basemaps.Hydda.Full)

measure = MeasureControl(
    position='bottomleft',
    active_color = 'orange',
    primary_length_unit = 'kilometers'
)
m.add_control(measure)

measure.completed_color = 'red'

measure.add_length_unit('yards', 1.09361, 4)
measure.secondary_length_unit = 'yards'

measure.add_area_unit('sqyards', 1.19599, 4)
measure.secondary_area_unit = 'sqyards'

m
```

34.2 Attributes

Attribute	Default Value	Doc
position	"topright"	Position of the control on the Map, possible values are topleft, topright, bottomleft or bottomright
primary_length_unit	"feet"	Primary length unit, possible values are feet, meters, miles, kilometers or any user defined length unit
secondary_length_unit	None	Secondary length unit, possible values are None, feet, meters, miles, kilometers or any user defined length unit
primary_area_unit	"acres"	Primary area unit, possible values are acres, hectares, sqfeet, sqmeters, sqmiles or any user defined area unit
secondary_area_unit	None	Secondary area unit, possible values are None, acres, hectares, sqfeet, sqmeters, sqmiles or any user defined area unit
active_color	"#ABE67E"	Color of the currently drawn area
completed_color	"#C8F2BE"	Color of the completed areas
popup_options	{'className': 'leaflet-measure-resultpopup', 'autoPanPadding': [10, 10]}	
capture_z_index	10000	Z-index of the marker used to capture measure clicks. Set this value higher than the z-index of all other map layers to disable click events on other layers while a measurement is active.

34.3 Methods

Method	Arguments	Doc
add_length_unit	name, factor, decimals=0	Adds a length unit with a name, a factor (factor to apply when converting to this unit. Length in meters will be multiplied by this factor), and an optional number of displayed decimals
add_area_unit	name, factor, decimals=0	Adds a area unit with a name, a factor (factor to apply when converting to this unit. Area in sqmeters will be multiplied by this factor), and an optional number of displayed decimals

CHAPTER 35

SplitMap Control

35.1 Example

```
from ipyleaflet import Map, basemaps, basemap_to_tiles, SplitMapControl

m = Map(center=(42.6824, 365.581), zoom=5)

right_layer = basemap_to_tiles(basemaps.NASAGIBS.ModisTerraTrueColorCR, "2017-11-11")
left_layer = basemap_to_tiles(basemaps.NASAGIBS.ModisAquaBands721CR, "2017-11-11")

control = SplitMapControl(left_layer=left_layer, right_layer=right_layer)
m.add_control(control)

m
```

35.2 Attributes

Attribute	Type	Default Value	Doc
left_layer	Layer instance		Left layer
right_layer	Layer instance		Right layer

CHAPTER 36

Draw Control

The `DrawControl` allows one to draw shapes on the map such as `Rectangle` `Circle` or lines.

```
from ipyleaflet import Map, basemaps, basemap_to_tiles, DrawControl

watercolor = basemap_to_tiles(basemaps.Stamen.Watercolor)

m = Map(layers=(watercolor, ), center=(50, 354), zoom=5)

draw_control = DrawControl()
draw_control.polyline =  {
    "shapeOptions": {
        "color": "#6bc2e5",
        "weight": 8,
        "opacity": 1.0
    }
}
draw_control.polygon = {
    "shapeOptions": {
        "fillColor": "#6be5c3",
        "color": "#6be5c3",
        "fillOpacity": 1.0
    },
    "drawError": {
        "color": "#dd253b",
        "message": "Oups!"
    },
    "allowIntersection": False
}
draw_control.circle = {
    "shapeOptions": {
        "fillColor": "#efed69",
        "color": "#efed69",
        "fillOpacity": 1.0
    }
```

(continues on next page)

(continued from previous page)

```
}
draw_control.rectangle = {
    "shapeOptions": {
        "fillColor": "#fca45d",
        "color": "#fca45d",
        "fillOpacity": 1.0
    }
}

m.add_control(draw_control)

m
```

CHAPTER 37

Widget Control

37.1 Example

```
from ipyleaflet import Map, basemaps, WidgetControl
from ipywidgets import IntSlider, ColorPicker, jslink

m = Map(center=(46.01, 6.16), zoom=12, basemap=basemaps.Stamen.Terrain)
zoom_slider = IntSlider(description='Zoom level:', min=0, max=15, value=7)
jslink((zoom_slider, 'value'), (m, 'zoom'))
widget_control1 = WidgetControl(widget=zoom_slider, position='topright')
m.add_control(widget_control1)

color_picker = ColorPicker(description='Pick a color:')
widget_control2 = WidgetControl(widget=color_picker, position='bottomright')
m.add_control(widget_control2)
m
```

37.2 Attributes

Attribute	Default Value	Doc
position	'topleft'	Position of the control, can be 'bottomleft', 'bottomright', 'topleft', or 'topright'
widget	None	Widget content
min_width	None	Min width of the widget (in pixels), if None it will respect the content size
max_width	None	Max width of the widget (in pixels), if None it will respect the content size
min_height	None	Min height of the widget (in pixels), if None it will respect the content size
max_height	None	Max height of the widget (in pixels), if None it will respect the content size

CHAPTER 38

Legend Control

38.1 Example

```
from ipyleaflet import Map, LegendControl

m = Map(center=(-10,-45), zoom=4)

legend = LegendControl({"low":"#FAA", "medium":"#A55", "High":"#500"}, name="Legend",␣
↪position="bottomright")
m.add_control(legend)

m
```

```
# Manipulate the legend

# Set/Get legend title
legend.name = "Risk"  # Set name
legend.name  # Get name

# Set/Get legend content
legend.legends = {"el1":"#FAA", "el2":"#A55", "el3":"#500"}  # Set content
legend.legends  # Get content

legend.add_legend_element("el5","#000")  # Add a legend element
legend.remove_legend_element("el5")  # Remove a legend element

# legend position
legend.position = "topright"  # Set position
legend.position  # Get current position
```

```
'topright'
```

38.2 Attributes

Attribute	Default Value	Doc
position	'topleft'	Position of the control, can be 'bottomleft', 'bottomright', 'topleft', or 'topright'
legend	None	A dictionary containing the name->color mapping that represents the legend
title	'Legend'	Legend name

CHAPTER 39

Search Control

39.1 Example

```
from ipyleaflet import Map, SearchControl, Marker, AwesomeIcon

m = Map(zoom=3, center=[19.1646, 72.8493])

marker = Marker(icon=AwesomeIcon(name="check", marker_color='green', icon_color=
↪'darkgreen'))

m.add_control(SearchControl(
  position="topleft",
  url='https://nominatim.openstreetmap.org/search?format=json&q={s}',
  zoom=5,
  marker=marker
))

m
```

You can also search features from GeoJSON layers.

```
import json
import os
import requests

from ipyleaflet import AwesomeIcon, GeoJSON, Map, Marker, LayerGroup, SearchControl

m = Map(zoom=3, center=[19.1646, 72.8493])

if not os.path.exists('countries.geo.json'):
      url = 'https://raw.githubusercontent.com/jupyter-widgets/ipyleaflet/master/
↪examples/countries.geo.json'
      r = requests.get(url)
      with open('countries.geo.json', 'w') as f:
```

(continues on next page)

(continued from previous page)

```
        f.write(r.content.decode("utf-8"))

with open("countries.geo.json") as f:
    data = json.load(f)

countries = GeoJSON(data=data)

layer_group = LayerGroup(layers=(countries,))
marker = Marker(icon=AwesomeIcon(name="check", marker_color='green', icon_color=
↪'darkred'))

m.add_control(SearchControl(
  position="topleft",
  layer=layer_group,
  zoom=4,
  property_name='name',
  marker=marker
))

m
```

39.2 Attributes

Attribute	Default Value	Doc
position	'topleft'	Position of the control, can be 'bottomleft', 'bottomright', 'topleft', or 'topright'
url	''	The url used for the search queries.
layer	None	The LayerGroup used for search queries.
zoom	10	Default zoom level for move to location
marker	Marker()	The marker used by the control.
found_style	{'fillColor': '#3f0', 'color': '#0f0'}	Style for searched feature when searching in LayerGroup.

CHAPTER 40

ipyleaflet related projects

Here is a list of existing open source projects that build functionality upon ipyleaflet.

- geemap: a Python package for interactive mapping with Google Earth Engine, ipyleaflet, and ipywidgets.
- xarray-leaflet: an xarray extension for tiled map plotting, based on ipyleaflet.
- ipygee: another Python package for interactive mapping with Google Earth Engine, ipyleaflet, and ipywidgets.