ipyleaflet

Dec 30, 2020

Installation

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 Example

```
from ipyleaflet import Map, basemaps, basemap_to_tiles

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

m
```

6.2 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	
crs	'EPSG3857'	Coordinate reference system, which can be 'Earth', 'EPSG3395', 'EPSG3857', 'EPSG4326', 'Base', or 'Simple'
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_zoom_limits	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	

6.3 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
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

CHAPTER 7

Tile Layer

7.1 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
```

7.2 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);
```

7.3 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) <a href=’https://openstreetmap.org’>OpenStreetMap</a> contributors”
detect_retina	False
opacity	1.0
visible	True
no_wrap	False
show_loading	False

CHAPTER 8

Local Tile Layer

8.1 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/".

8.2 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 9

Marker

9.1 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
```

9.2 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

9.3 Methods

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

CHAPTER 10

Icon

10.1 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
```

10.2 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 11

Popup

11.1 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
```

11.2 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 12

WMS Layer

12.1 Example

```python
from ipyleaflet import Map, WMSLayer

wms = WMSLayer(
    url="https://demo.boundlessgeo.com/geoserver/ows?",
    layers="nasa:bluemarble"
)

m = Map(layers=(wms, ), center=(42.5531, -48.6914), zoom=3)

m
```

12.2 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) <a href='https://openstreetmap.org'>OpenStreetMap</a> contributors"	
detect_retina	False	
opacity	1.0	
visible	True	
service	"WMS"	
request	"GetMap"	
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
version	"1.1.1"	Version of the WMS service to use
crs	""	

CHAPTER 13

Image overlay and Video overlay

13.1 Example

```python
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
```

13.2 Attributes

Attribute	Default Value	Doc
url	“”	Url to the footage
bounds	((0.0, 0.0), (0.0, 0.0))	SW and NE corners of the image

CHAPTER 14

Polyline

14.1 Example Polyline

```
from ipyleaflet import Map, Polyline

line = Polyline(
    locations = [[
    [[45.51, -122.68],
     [37.77, -122.43],
     [34.04, -118.2]],]],
    color = "green" ,
    fill_color= "green")
m = Map(center = (42.5, -41), zoom =2)
m.add_layer(line)
m
```

14.2 Example MultiPolyline

```
from ipyleaflet import Map, Polyline

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

(continues on next page)

(continued from previous page)

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

14.3 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	“#0033FF”	
fill_opacity	0.2	
dash_array		
line_cap	“round”	
line_join	“round”	

CHAPTER 15

Polygon/Multipolygon

15.1 Example Polygon

```
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
```

15.2 Example Polygon with hole

```
from ipyleaflet import Map, Polygon

hole_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)]],

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

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

(continues on next page)

(continued from previous page)

```
m
```

15.3 Example 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
```

15.4 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	“#0033FF”	
fill_opacity	0.2	
dash_array		
line_cap	“round”	
line_join	“round”	

CHAPTER 16

Rectangle

16.1 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
```

16.2 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	"#0033FF"	
fill_opacity	0.2	
dash_array		
line_cap	"round"	
line_join	"round"	

CHAPTER 17

Circle

17.1 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
```

17.2 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	“#0033FF”	
fill_opacity	0.2	
dash_array		
line_cap	“round”	
line_join	“round”	

CHAPTER 18

Circle Marker

18.1 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
```

18.2 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	"#0033FF"	
fill_opacity	0.2	
dash_array		
line_cap	"round"	
line_join	"round"	

CHAPTER 19

Marker Cluster

19.1 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
```

19.2 Attributes

Attribute	Default Value	Doc
markers	()	Tuple of markers

CHAPTER 20

Heatmap

20.1 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
```

20.2 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 21

Velocity

21.1 Example

```
from ipyleaflet import Map, Velocity, TileLayer, basemaps
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,
                display_options=display_options)
```

(continues on next page)

(continued from previous page)

```
m.add_layer(wind)

m
```

21.2 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 22

Layer Group

22.1 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
```

22.2 Attributes

Attribute	Default Value	Doc
layers	()	List of layers

22.3 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 23

GeoJSON

23.1 Example

```python
from ipyleaflet import Map, GeoJSON
import json
import os
import requests

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)

m = Map(center=(50.6252978589571, 0.34580993652344), zoom=3)
geo_json = GeoJSON(data=data, style = {'color': 'green', 'opacity':1, 'weight':1.9,
↪'dashArray':'9', 'fillOpacity':0.1})
m.add_layer(geo_json)
m
```

23.2 Attributes

Attribute	Doc
data	Data dictionary
style	Style dictionary
hover_style	Hover style dictionary

23.3 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 24

GeoData

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

24.1 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
```

24.2 Attributes

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

CHAPTER 25

Choropleth

25.1 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, nom_fichier, type_fichier):
    r = requests.get(url)
    with open(nom_fichier, 'w') as f:
        f.write(r.content.decode("utf-8"))
    with open(nom_fichier, 'r') as f:
        return type_fichier(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
```

25.2 Information

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 takes `'id'` from `'features'` as key and float as value, in order to build the colormap :

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

25.3 Attributes

Attribute	Doc	Description
geo_data	Data dictionary	GeoJSON dictionary
choro_data	Choropleth data dictionary	Dictionary id/float
value_min	Color scale minimum value	
value_max	Color scale maximum value	
colormap	Map of color from branca	

CHAPTER 26

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.

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

```
from ipyleaflet import (
    Map, basemaps, basemap_to_tiles,
    WMSLayer, LayersControl
)

m = Map(center=(50, 354), zoom=4)

nasa_layer = basemap_to_tiles(basemaps.NASAGIBS.ModisTerraTrueColorCR, "2018-03-30")
m.add_layer(nasa_layer)

wms = WMSLayer(
    url="https://demo.boundlessgeo.com/geoserver/ows?",
    layers="nasa:bluemarble",
    name="nasa:bluemarble"
)
m.add_layer(wms)

m.add_control(LayersControl())

m
```

CHAPTER 27

Fullscreen Control

27.1 Example

```python
from ipyleaflet import Map, FullScreenControl

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

m
```

CHAPTER 28

Measure Control

28.1 Example

```
from ipyleaflet import Map, MeasureControl

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
```

28.2 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.

28.3 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 29

SplitMap Control

29.1 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
```

29.2 Attributes

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

CHAPTER 30

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 31

Widget Control

31.1 Example

```python
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
```

31.2 Attributes

Attribute	Doc
widget	Widget content
min_width	Min width of the widget
max_width	Min width of the widget
min_height	Min height of the widget
max_height	Min height of the widget