HERE Map Widget for Jupyter

HERE Europe B.V.

Oct 29, 2021

GETTING STARTED

1 Prerequisites 3

2 Installation 5

3 Map 7

4 Japan Basemap 9

5 Raster Basemaps 11

6 Vector Basemaps 13

7 Map Tile Basemap 15

8 External Basemaps 17

9 GeoJSON 19

10 GeoData Layer 21

11 XYZ Space Layer 23

12 Choropleth Layer 25

13 KML Layer 27

14 HeatMap Layer 29

15 MarkerCluster Layer 31

16 ImageTile Layer 33

17 Circle Object 35

18 DomIcon Object 37

19 DomMarker Object 39

20 Group Object 41

21 Icon Object 43

22 Marker Object 45

23 Overlay Object 47
24 Polygon Object 49
25 Polyline Object 51
26 Rectangle Object 53
27 Point 55
28 LineString 57
29 MultiLineString 59
30 GeoPolygon 61
31 GeoMultiPolygon 63
32 BBOX 65
33 WKT 67
34 Info Bubble Element 69
35 Style Element 71
36 Fullscreen Control 73
37 Map Settings Control 75
38 Measurement Control 77
39 Scale Bar Control 79
40 Search Control 81
41 Split Map Control 83
42 Zoom Control 85
43 Zoom Rectangle Control 87
44 Widget Control 89
45 Layers Control 91
46 Default Layers Names 93
47 Service Names 95
48 Service Urls 97
49 Miscellaneous Values 101
50 CHANGELOG 103
51 Contributing to HERE Map Widget for Jupyter 105

An interactive widget that enables users to use HERE Maps API for JavaScript in Jupyter Notebook. `HERE Map Widget for Jupyter` builds the connection between the analytic capabilities of the Jupyter ecosystem and the superior map visualization and location services capabilities of HERE Maps API for JavaScript.

`HERE Map Widget for Jupyter` is utilizing the extension system of Jupyter widgets to enhance its core functionality with a widget to create interactive maps.

HERE Maps API for JavaScript offers Web Developers a JavaScript library with a rich set of functionalities and capabilities to build location-aware applications. Among other functionality, it supports 2D and 3D visualization, various data formats, an event system, and a seamless integration with HERE Location Services and HERE Data Hub.

Important features of `HERE Map Widget for Jupyter`:

- Interactive HERE Map in Jupyter Notebooks
- Tilt rotate / 3D Map appearance
- Integration with HERE Location Services
- Integration with HERE Data Hub
- HERE Vector Tiles
- HERE Raster Tiles

For full list of functionalities please see below sections.

CHAPTER

ONE

PREREQUISITES

Before you install the `here-map-widget-for-jupyter` make sure you meet the following prerequisites:

- A Python installation, 3.6+ recommended, with the `pip` command available to install dependencies.
- A HERE developer account, freely available under HERE Developer Portal.
- An API key from the HERE Developer Portal.

CHAPTER

TWO

INSTALLATION

Install HERE Map Widget for Jupyter with conda from the Anaconda conda-forge channel using the below command:

```
$ conda install -c conda-forge here-map-widget-for-jupyter
```

Install HERE Map Widget for Jupyter from PyPI using the below command:

```
$ pip install here-map-widget-for-jupyter
```

Below extra commands are required only if you are using classic Jupyter Notebook (version 5.3 or older):

```
$ jupyter nbextension install --py --sys-prefix here_map_widget
$ jupyter nbextension enable here_map_widget --py --sys-prefix
```

Below extra commands are required only if you are using JupyterLab (version 2 or older):

```
$ npm config set @here:registry https://repo.platform.here.com/artifactory/api/npm/here-
↪node/
$ jupyter labextension install @jupyter-widgets/jupyterlab-manager @here/map-widget-for-
↪jupyter
```

2.1 Installation from source repository on GitHub

For a development installation (requires yarn, you can install it with `conda install -c conda-forge yarn`):

```
$ git clone https://github.com/heremaps/here-map-widget-for-jupyter.git
$ cd here-map-widget-for-jupyter
$ pip install -e .
```

If you are using the classic Jupyter Notebook you need to install the nbextension:

```
$ jupyter nbextension install --py --sys-prefix here_map_widget
$ jupyter nbextension enable here_map_widget --py --sys-prefix
```

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 developing with JupyterLab:

```
$ jupyter labextension develop --overwrite here_map_widget
```

CHAPTER

THREE

MAP

Map is a here-map-widget-for-jupyter class that allows making a Map view and all other elements are added on this base object.

3.1 Example

```
from here_map_widget import Map, Style
import os

style = Style(
    config="https://heremaps.github.io/maps-api-for-javascript-examples/"
    + "change-style-at-load/data/dark.yaml",
    base_url="https://js.api.here.com/v3/3.1/styles/omv/",
)


m = Map(
    api_key=os.environ["LS_API_KEY"],
    style=style,
    center=[52.51477270923461, 13.39846691425174],
    zoom=13,
)

m
```

3.2 Attributes

Attribute	Type	Doc
api_key	string	API Key used for authentication
center	list	The current center of the map
zoom	float	The current zoom value of the map
heading	float	The current heading value of the map
incline	float	The current incline value of the map
tilt	float	The current tilt value of the map
basemap	string	The current basemap of the map, default value is vector.normal.map
layers	list of objects	The list of layers that are currently on the map
objects	list of objects	The list of objects that are currently on the map
controls	list of objects	The list of controls that are currently on the map
bubbles	list of objects	The list of bubbles that are currently on the map
style	Style object	Style to apply for basemap
bounds	tuple	Bounding box coordinates South, West, North, East

3.3 Methods

Method	Arguments	Doc
add_layer	Layer object	Adds a Layer on the Map
remove_layer	Layer object	Removes a Layer from the Map
add_object	Object instance	Adds a Object on the Map
add_objects	Object instances	Adds a list of Objects on the Map
remove_object	Object instance	Removes a Object from the Map
add_control	Control object	Adds a Control on the Map
remove_control	Control object	Removes a Control from the Map
add_bubble	InfoBubble object	Adds a InfoBubble on the Map
remove_bubble	InfoBubble object	Removes a InfoBubble from the Map

CHAPTER

FOUR

JAPAN BASEMAP

`Japan Basemap` is the type of basemap of `here-map-widget-for-jupyter` which shows basemap data for Japan. For more information on basemap with Japan data please check this link.

4.1 Example

```
import os
from here_map_widget import Map, OMV, Platform, Style, TileLayer
from here_map_widget import ServiceNames, OMVUrl

services_config = {
    ServiceNames.omv: {
        OMVUrl.scheme: "https",
        OMVUrl.host: "vector.hereapi.com",
        OMVUrl.path: "/v2/vectortiles/core/mc",
    }
}

platform = Platform(api_key=os.environ["LS_API_KEY"], services_config=services_config)

style = Style(
    config="https://js.api.here.com/v3/3.1/styles/omv/oslo/japan/normal.day.yaml",
    base_url="https://js.api.here.com/v3/3.1/styles/omv/oslo/japan/",
)

omv_provider = OMV(path="v2/vectortiles/core/mc", platform=platform, style=style)

omv_layer = TileLayer(provider=omv_provider, style={"max": 22})

center = [35.68026, 139.76744]
m = Map(api_key=os.environ["LS_API_KEY"], center=center, zoom=8, basemap=omv_layer)
m
```

4.2 Attributes

Attribute	Type	Doc
basemap	object	The basemap is object of `TileLayer`.

CHAPTER

FIVE

RASTER BASEMAPS

`Raster Basemaps` are pre-configured set of HERE layers for convenient use as basemap supported by `here-map-widget-for-jupyter`. Detailed information about raster basemaps supported by `HERE Maps API for JavaScript` can be found here.

5.1 Example

```
from here_map_widget import Map, DefaultLayers, DefaultLayerNames, Platform
import os

default_layer = DefaultLayers(layer_name=DefaultLayerNames.raster.satellite.map)

m = Map(
    api_key=os.environ["LS_API_KEY"], basemap=default_layer, center=[44.2002, -72.7566]
)
m
```

5.2 Attributes

Attribute	Type	Doc
basemap	object	Object of `DefaultLayers`. for various types of `layer_name` supported by `DefaultLayers`, please check *config*.

CHAPTER

SIX

VECTOR BASEMAPS

`Vector Basemaps` are pre-configured set of HERE layers for convenient use as basemap supported by `here-map-widget-for-jupyter`. Detailed information about vector basemaps supported by `HERE Maps API for JavaScript` can be found here.

6.1 Example

```
from here_map_widget import Map, DefaultLayers, DefaultLayerNames, Platform
import os

default_layer = DefaultLayers(layer_name=DefaultLayerNames.vector.normal.map)

m = Map(
    api_key=os.environ["LS_API_KEY"], basemap=default_layer, center=[44.2002, -72.7566]
)
m
```

6.2 Attributes

Attribute	Type	Doc
basemap	object	Object of `DefaultLayers`. for various types of `layer_name` supported by `DefaultLayers`, please check *config*.

CHAPTER

SEVEN

MAP TILE BASEMAP

`MapTile Basemap` is the type of basemap of `here-map-widget-for-jupyter` which shows basemap using Map Tile API.

7.1 Example

```
from here_map_widget import Map, Platform, MapTile, TileLayer
from here_map_widget import ServiceNames, MapTileUrl
import os

services_config = {
    ServiceNames.maptile: {
        MapTileUrl.scheme: "https",
        MapTileUrl.host: "maps.ls.hereapi.com",
        MapTileUrl.path: "maptile/2.1",
    }
}

platform = Platform(api_key=os.environ["LS_API_KEY"], services_config=services_config)

maptile = MapTile(
    tile_type="maptile",
    scheme="hybrid.day",
    tile_size=256,
    format="jpg",
    platform=platform,
    type="aerial",
)

maptile_layer = TileLayer(provider=maptile, style={"max": 22})

m = Map(
    api_key=os.environ["LS_API_KEY"],
    center=[35.68026, 139.76744],
    zoom=17,
    basemap=maptile_layer,
)
m
```

7.2 Attributes

Attribute	Type	Doc
basemap	object	The basemap is object of `TileLayer`.

CHAPTER

EIGHT

EXTERNAL BASEMAPS

`here-map-widget-for-jupyter` also supports basemaps from external tile providers. We use tile providers defined in xyzservices `here_map_widget.basemaps` is mapped to `xyzservices.providers`

8.1 Example

```
import os
from here_map_widget import Map, basemaps

m = Map(
    api_key=os.environ["LS_API_KEY"],
    center=[52.51477270923461, 13.39846691425174],
    zoom=4,
    basemap=basemaps.OpenStreetMap.Mapnik,
)
m
```

8.2 Attributes

Attribute	Type	Doc
basemap	object	The basemap is object of `xyzservices.lib.TileProvider`, defined under `here_map_widget.basemaps`.

CHAPTER

NINE

GEOJSON

`GeoJSON` is an `here-map-widget-for-jupyter` class that allows you to visualize a GeoJSON Data on the Map.

9.1 Example

```
from here_map_widget import Map, GeoJSON
import os

# Add GeoJSON from URL
m = Map(api_key=os.environ["LS_API_KEY"])
url = (
    "https://gist.githubusercontent.com/peaksnail/"
    + "5d4f07ca00ed7c653663d7874e0ab8e7/raw/"
    + "64c2a975482efd9c42e54f6f6869f091055053cd/countries.geo.json"
)
geojson = GeoJSON(
    url=url,
    disable_legacy_mode=True,
    style={"fillColor": "rgba(245, 176, 65, 0.5)", "strokeColor": "black"},
    show_bubble=True,
)
m.add_layer(geojson)
m
```

9.2 Attributes

Attribute	Type	Doc
data	dict	GeoJSON Data to be plotted
url	string	GeoJSON Data URL to be plotted
disable_legacy_mode	boolean	Disable legacy mode for parsing GeoJSON data. Default value is True
style	dict	The style to use for rendering GeoJSON data. Example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10, 'lineCap': 'square', lineJoin: 'bevel'}
hover_style	dict	The style to use for rendering data when hovered over. Example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10}
evt_type	string	Event type to be used to apply hover_style, please check *allowed values*, default value is `tap`.
show_bubble	boolean	To determine whether to show info bubble for space data or not
style_callback	Callable object	A callback function which is called for each feature to generate style for the feature
point_style	dict	The style to use for rendering Points in GeoJSON if not provided default Markers will be used. Example: {"strokeColor": 'white', "lineWidth": 1, "fillColor": "#1b468d", "fillOpacity": 0.7, "radius": 5}, radius should be between 2 to 8.

9.3 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

TEN

GEODATA LAYER

GeoData is an here-map-widget-for-jupyter class that allows you to visualize a GeoDataFrame on the Map.

10.1 Example

```
from here_map_widget import Map, GeoData
import geopandas
import json
import os

countries = geopandas.read_file(geopandas.datasets.get_path("naturalearth_lowres"))

m = Map(api_key=os.environ["LS_API_KEY"], center=[52.3, 8.0], zoom=3)

geo_data = GeoData(
    geo_dataframe=countries,
    style={
        "fillColor": "#3366cc",
    },
    hover_style={"fillColor": "red"},
    show_bubble=True,
)
m.add_layer(geo_data)
m
```

10.2 Attributes

Attribute	Type	Doc
geo_data	GeoDataFrame	The GeoPandas dataframe to use
style	dict	The style to use for rendering GeoJSON data
disable_legacy_mode	boolean	Disable legacy mode for parsing GeoJSON data default `True`.
style	dict	The style to use for rendering GeoJSON data. Example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10, 'lineCap': 'square', lineJoin: 'bevel'}
hover_style	dict	The style to use for rendering data when hovered over. Example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10}
evt_type	string	Event type to be used to apply hover_style, please check *allowed values*, default value is `tap`.
show_bubble	boolean	To determine whether to show info bubble for space data or not
style_callback	Callable object	A callback function which is called for each feature to generate style for the feature
point_style	dict	The style to use for rendering Points in GeoJSON if not provided default Markers will be used. Example: {"strokeColor": 'white', "lineWidth": 1, "fillColor": "#1b468d", "fillOpacity": 0.7, "radius": 5}, radius should be between 2 to 8.

CHAPTER

ELEVEN

XYZ SPACE LAYER

`XYZ` is an `here-map-widget-for-jupyter` class that allows you to visualize a XYZ Space on the Map.

11.1 Example

```
from here_map_widget import TileLayer, XYZ
from here_map_widget import Map
import os

m = Map(api_key=os.environ["LS_API_KEY"])
m.zoom = 9
m.center = [44.20022717941052, -72.75660780639646]

style_flagged = {
    "layers.xyz.points.Places": {
        "filter": {"properties.GPSFLG": "Flagged for coordinate check"},
        "draw": {
            "points": {
                "color": "blue",
                "text": {
                    "priority": 0,
                    "font": {
                        "size": "12px",
                        "fill": "red",
                        "stroke": {"color": "white", "width": "0.5px"},
                    },
                },
            }
        },
    }
}

xyz_token = os.environ["XYZ_TOKEN"]
provider = XYZ(space_id="m2pcsiNi", token=xyz_token, show_bubble=True)
space = TileLayer(provider=provider, style=style_flagged)
m.add_layer(space)
m
```

11.2 Attributes

Attribute	Type	Doc
token	string	XYZ token to access XYZ space data
space_id	string	Space id from which to access the data
evt_type	string	Event on which to show info bubble for space data, please check *allowed values*, default value is `tap`.
show_bubble	boolean	To determine whether to show info bubble for space data or not
style	dict/Style object	To provide style to for the XYZ space data
platform	Platform	Optional required only for custom Data Hub endpoints.

11.3 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

TWELVE

CHOROPLETH LAYER

Choropleth is an here-map-widget-for-jupyter class that allows you to generate a Choropleth Map for your data.

12.1 Example

```
import here_map_widget
import json
import pandas as pd
import os
import requests
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())
)
```

(continues on next page)

(continued from previous page)

```
layer = here_map_widget.Choropleth(
    geo_data=geo_json_data,
    choro_data=unemployment,
    colormap=linear.YlOrRd_04,
)

m = here_map_widget.Map(api_key=os.environ["LS_API_KEY"], center=[43, -100], zoom=4)
m.add_layer(layer)
m
```

12.2 Attributes

Attribute	Type	Doc
geo_data	dict	The GeoJSON structure on which to apply the Choropleth effect
choro_data	dict	Data used for building the Choropleth effect
value_min	float	Minimum data value for the color mapping
value_max	float	Maximum data value for the color mapping
colormap	branca.colormap.ColorMap	The **ColorMap** used for the effect
key_on	string	The feature key to use for the colormap effect, default value is `id`.
style	dict	The style to use for rendering GeoJSON data. Example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10, 'lineCap': 'square', lineJoin: 'bevel'}
hover_style	dict	The style to use for rendering data when hovered over. Example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10}
disable_legacy_mode	boolean	Disable legacy mode for parsing GeoJSON data default True
evt_type	string	Event type to be used to apply hover_style, please check *allowed values*, default value is `tap`.
show_bubble	boolean	To determine whether to show info bubble for space data or not
style_callback	Callable object	A callback function which is called for each feature to generate style for the feature
point_style	dict	The style to use for rendering Points in GeoJSON if not provided default Markers will be used. Example: {"strokeColor": 'white', "lineWidth": 1, "fillColor": "#1b468d", "fillOpacity": 0.7, "radius": 5}, radius should be between 2 to 8.

CHAPTER

THIRTEEN

KML LAYER

`KML` is an `here-map-widget-for-jupyter` class that allows you to visualize KML Data on the Map.

13.1 Example

```
from here_map_widget import Map, KML
import os

m = Map(
    api_key=os.environ["LS_API_KEY"], center=[44.20022717941052, -72.75660780639646]
)
url = (
    "https://heremaps.github.io/maps-api-for-javascript-examples/"
    + "display-kml-on-map/data/us-states.kml"
)
kml = KML(url=url)
m.add_layer(kml)
m
```

13.2 Attributes

Attribute	Type	Doc
url	string	KML Data URL to be plotted

CHAPTER

FOURTEEN

HEATMAP LAYER

`Heat Map` is an `here-map-widget-for-jupyter` class that allows you to generate a Heat Map for your data.

14.1 Example

```
from random import uniform
from here_map_widget import TileLayer, HeatMap
from here_map_widget import Map
import os

data = [[uniform(-80, 80), uniform(-180, 180), uniform(0, 1000)] for i in range(1000)]

heat_map_data = []
for row in data:
    heat_map_data.append({"lat": row[0], "lng": row[1], "value": row[2]})

m = Map(api_key=os.environ["LS_API_KEY"])
provider = HeatMap(interpolate=True, opacity=0.6, assume_values=True)
provider.add_data(heat_map_data)
heatmap = TileLayer(provider=provider)
m.add_layer(heatmap)
m
```

14.2 Attributes

Attribute	Type	Doc
colors	dict	An dict object defining the color stops for example *{'0.0': 'blue', '0.5': 'yellow', '1': 'red'}*
interpolate	boolean	A value indicating whether interpolation is to be used to display smooth color transitions in the heat map
opacity	float	The opacity which is used for the rendering of the heatmap in range [0..1]
assume_values	boolean	A Boolean value indicating whether to paint assumed values in regions where no data is available
data	list	Data list which is used to generate heat map
hardReload	boolean	A value indicating whether to invalidate in hard mode (True) or in soft mode (False) while adding data

14.3 Methods

Method	Arguments	Doc
add_data	data	Data list which is used to generate heat map
clear		Clear Heat Map

CHAPTER

FIFTEEN

MARKERCLUSTER LAYER

Sometimes, you may need to display a large set of data on the map, for example several thousand points. There are two potential problems you may encounter: one is a possible performance degradation when all the points (markers) are visible at lower zoom levels, the other is the fact that the markers located in close geographic proximity to one another may visibly overlap and even hide one another at lower zoom levels. A solution to both these problems is offered by `MarkerCluster Layer`.

15.1 Example

```
import os
from here_map_widget import Map, MarkerCluster, ZoomControl, ObjectLayer


m = Map(api_key=os.environ["LS_API_KEY"], center=[51.01, 0.01], zoom=7)

data = """
<!DOCTYPE html>
<html>
<body>
<h1>Marker</h1>
<p>{}</p>
</body>
</html>
"""
p1 = dict(lat=51.01, lng=0.01, data=data.format("First Marker"))
p2 = dict(lat=50.04, lng=1.01, data=data.format("Second Marker"))
p3 = dict(lat=51.45, lng=1.01, data=data.format("Third Marker"))
p4 = dict(lat=51.01, lng=2.01, data=data.format("Fourth Marker"))

provider = MarkerCluster(data_points=[p1, p2, p3, p4], show_bubble=True)
layer = ObjectLayer(provider=provider)
m.add_layer(layer)
zc = ZoomControl(alignment="LEFT_TOP")
m.add_control(zc)
m
```

15.2 Attributes of MarkerCluster

Attribute	Type	Doc
data_points	List	A list of dictionaries, each dictionary will have keys: `lat` for latitude of the point, `lng` for longitude of the point, `data` for data associated with point which should be HTML string, `weight` for weight of the point as a positive number, the default is 1.
eps	Int	The epsilon parameter for cluster calculations. It must not exceed 256 and must take values that are a power of 2. Default is 32.
min_weight	Int	The minimum point weight sum to form a cluster, the default is 2.
min_zoom	Int	The minimum supported zoom level, the default is 0.
max_zoom	Int	The maximum supported zoom level, the default is 22.
show_bubble	Boolean	If `True` then shows data associated with point as `InfoBubble`, default is `False`.
evt_type	string	The type of event to be used to show `InfoBubble`. please check *allowed values*. Default value is `tap`.

CHAPTER

SIXTEEN

IMAGETILE LAYER

To load tiles from XYZ tile servers like OpenStreetMap tile servers or WMTS tile servers, `ImageTileProvider` is used as a source of data for TileLayer.

16.1 XYZ Tile server Example

```
import os
from here_map_widget import Map, ImageTileProvider, TileLayer

m = Map(api_key=os.environ["LS_API_KEY"], center=[39.40, -104.08], zoom=3)

url = "https://a.tile.openstreetmap.org/{z}/{x}/{y}.png"
attribution = (
    'Map data (c) <a href="https://openstreetmap.org">OpenStreetMap</a> contributors'
)
provider = ImageTileProvider(url=url, attribution=attribution)
layer = TileLayer(provider=provider)
m.add_layer(layer)
m
```

16.2 WMTS Example

```
import os
from here_map_widget import Map, ImageTileProvider, TileLayer


m = Map(api_key=os.environ["LS_API_KEY"], center=[39.40, -104.08], zoom=3.66)

url = "https://services.arcgisonline.com/arcgis/rest/services/Demographics/USA_
↪Population_Density/MapServer/WMTS/?layer=0&style=default&tilematrixset=EPSG%3A4326&
↪Service=WMTS&Request=GetTile&Version=1.0.0&Format=image%2Fpng&TileMatrix={z}&TileCol=
↪{x}&TileRow={y}"
provider = ImageTileProvider(url=url)
layer = TileLayer(provider=provider)
m.add_layer(layer)
m
```

16.3 Attributes

Attribute	Type	Doc
url	String	URL of the Tile server.
min_zoom	Int	Optional, The minimum supported zoom level, the default is 0
max_zoom	Int	Optional, The maximum supported zoom level, the default is 22
opacity	Float	Optional, The opacity to use for the rendering of the provided tiles in range [0..1] where 0.0 means full transparent and 1.0 means full opaque.
tile_size	Int	Optional, The size of a tile as edge length in pixels. It must be 2^n where n is in the range [0 . . . 30].
headers	Dict	Optional,A dictionaary of headers to be sent with each request made by the provider.
attribution	String	Optional, Tiles service attribution.

CHAPTER

SEVENTEEN

CIRCLE OBJECT

`Circle` is an `here-map-widget-for-jupyter` class that allows you to visualize a circle on the Map. Style of object is a dictionary, to get more information on all possible keys of style dictionary, example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10, 'lineCap': 'square', lineJoin: 'bevel'}.

17.1 Example

```
from here_map_widget import Map
from here_map_widget import Point, Circle, FullscreenControl
import os

center = [19.152761, 72.87869]

m = Map(api_key=os.environ["LS_API_KEY"], center=center, zoom=10)

style = {"strokeColor": "#829", "lineWidth": 4}

point = Point(lat=19.152761, lng=72.87869)
circle = Circle(center=point, radius=10000, style=style, draggable=True)
m.add_object(circle)
m.add_control(FullscreenControl())
m
```

17.2 Attributes

Attribute	Type	Doc
center	Point object	The geographical coordinates of the center of the circle
radius	float	The radius of the circle in meters
style	dict	The style to be used when tracing the polyline (circle)
draggable	boolean	To make circle draggable
extrusion	float	The extrusion height for the circle in meters, default is 0
elevation	float	The elevation height of the circle in meters, default is 0

CHAPTER

EIGHTEEN

DOMICON OBJECT

DomIcon is an here-map-widget-for-jupyter class that allows you to visualize a marker with a custom DomIcon on the Map.

18.1 Example

```
from here_map_widget import Map, FullscreenControl, DomMarker, DomIcon
import os

m = Map(api_key=os.environ["LS_API_KEY"])
m.center = [41.8625, -87.6166]
m.zoom = 15
icon = DomIcon(
    element='<div style="user-select: none; cursor: default;"> \
<div style="color: red; background-color: blue; border: 2px solid black;'
    + "font: 12px / 12px arial; padding-top: 2px; padding-left: 4px; width: 20px;"
    + 'height: 20px; margin-top: -10px; margin-left: -10px;">C</div> \
</div>'
)
marker = DomMarker(lat=41.8625, lng=-87.6166, icon=icon)
m.add_object(marker)
m
```

18.2 Attributes

Attribute	Type	Doc
element	string	The element or markup to use for this icon

CHAPTER

NINETEEN

DOMMARKER OBJECT

DomMarker is an here-map-widget-for-jupyter class that allows you to visualize a DomIcon on the Map. A marker which supports HTML (and SVG) content, which can be dynamic. Markers of this type are best displayed individually or in small sets.

19.1 Example

```
from here_map_widget import Map, FullscreenControl, DomMarker, DomIcon
import os

m = Map(api_key=os.environ["LS_API_KEY"])
m.center = [41.8625, -87.6166]
m.zoom = 15
icon = DomIcon(
    element='<div style="user-select: none; cursor: default;"> \
<div style="color: red; background-color: blue; border: 2px solid black;'
    + "font: 12px / 12px arial; padding-top: 2px; padding-left: 4px; width: 20px;"
    + 'height: 20px; margin-top: -10px; margin-left: -10px;">C</div> \
</div>'
)
marker = DomMarker(lat=41.8625, lng=-87.6166, icon=icon)
m.add_object(marker)
m
```

19.2 Attributes

Attribute	Type	Doc
lat	float	Latitude for the marker to be added
lng	float	The element or markup to use for this icon
alt	float	Altitude for the marker to be added, default value is 0
icon	DomIcon object	DomIcon object to be added as marker

CHAPTER

TWENTY

GROUP OBJECT

Group is an here-map-widget-for-jupyter class that is a container for other map objects.

20.1 Example

```
from here_map_widget import LineString, Polyline, Marker, Group
from here_map_widget import Map, Bbox, Rectangle, Point, Circle, LineString, Polygon
import os

m = Map(api_key=os.environ["LS_API_KEY"], center=[51.1657, 10.4515])

# Polyline
plstyle = {"lineWidth": 15}
l = [53.3477, -6.2597, 0, 51.5008, -0.1224, 0, 48.8567, 2.3508, 0, 52.5166, 13.3833, 0]
ls = LineString(points=l)
pl = Polyline(object=ls, style=plstyle, draggable=True)


# Rectangle
rstyle = {"strokeColor": "#829", "lineWidth": 4}
bbox = Bbox(top=53.1, left=13.1, bottom=43.1, right=40.1)
rectangle = Rectangle(bbox=bbox, style=rstyle, draggable=True)


# Circle
point = Point(lat=51.1657, lng=10.4515)
cstyle = {"strokeColor": "#829", "lineWidth": 4}
circle = Circle(center=point, radius=1000000, style=cstyle, draggable=True)


# Polygon
pgstyle = {"strokeColor": "#829", "lineWidth": 4}

l = [52, 13, 100, 48, 2, 100, 48, 16, 100, 52, 13, 100]

ls = LineString(points=l)

pg = Polygon(object=ls, style=pgstyle, draggable=True)
```

(continues on next page)

(continued from previous page)

```
marker = Marker(lat=51.1657, lng=10.4515, evt_type="tap", draggable=True)

# Group
group = Group(volatility=True)

group.add_objects([pl, rectangle, circle, pg, marker])

m.add_object(group)
m
```

20.2 Attributes

Attribute	Type	Doc
volatility	boolean	Indicates whether the map object is volatile, the default is false

20.3 Methods

Method	Arguments	Doc
add_object	Object	Add object to the group
add_objects	List of Objects	Add objects to the group
remove_object	Object	Remove object from the group
remove_objects	List of Objects	Remove objects from the group

CHAPTER

TWENTYONE

ICON OBJECT

`Icon` is an `here-map-widget-for-jupyter` class that allows you to visualize a marker with a custom Icon on the Map.

21.1 Example

```
from here_map_widget import Map, Marker, WKT, Icon
import os

m = Map(api_key=os.environ["LS_API_KEY"], zoom=12)
m.center = [19.1663, 72.8526]
svg_markup = (
    '<svg width="24" height="24" '
    + 'xmlns="http://www.w3.org/2000/svg">'
    + '<rect stroke="white" fill="#1b468d" x="1" y="1" width="22" '
    + 'height="22" /><text x="12" y="18" font-size="15pt" '
    + 'font-family="Arial" font-weight="bold" text-anchor="middle" '
    + 'fill="white">M</text></svg>'
)

svg_icon = Icon(bitmap=svg_markup, height=30, width=30)

mumbai_marker = Marker(lat=19.1663, lng=72.8526, icon=svg_icon)
m.add_object(mumbai_marker)
m
```

21.2 Attributes

Attribute	Type	Doc
bitmap	string	An image URL, an SVG (string), an bitmap image or a canvas
height	float	Height of the icon
width	float	Width of the icon
anchor	dict	The anchor point in pixels, the default is bottom-center

CHAPTER

TWENTYTWO

MARKER OBJECT

`Marker` is an `here-map-widget-for-jupyter` class that allows you to visualize an icon on the Map. A "normal" marker that uses a static image as an icon. Large numbers of markers of this types can be added to the map very quickly and efficiently.

22.1 Example

```
from here_map_widget import Map, Marker, WKT, Icon
import os

m = Map(api_key=os.environ["LS_API_KEY"], zoom=12)
m.center = [19.1663, 72.8526]
svg_markup = (
    '<svg width="24" height="24" '
    + 'xmlns="http://www.w3.org/2000/svg">'
    + '<rect stroke="white" fill="#1b468d" x="1" y="1" width="22" '
    + 'height="22" /><text x="12" y="18" font-size="15pt" '
    + 'font-family="Arial" font-weight="bold" text-anchor="middle" '
    + 'fill="white">M</text></svg>'
)

svg_icon = Icon(bitmap=svg_markup, height=30, width=30)

mumbai_marker = Marker(lat=19.1663, lng=72.8526, icon=svg_icon)
m.add_object(mumbai_marker)
m
```

22.2 Attributes

Attribute	Type	Doc
lat	float	Latitude for the marker to be added
lng	float	The element or markup to use for this icon
alt	float	Altitude for the marker to be added, default value is 0
data	string	Optional arbitrary data to be stored with the given map object
draggable	boolean	To make Marker draggable
evt_type	string	Event type on which info bubble is showed, please check *allowed values*, default value is *tap*
show_bubble	boolean	To determine whether to show info bubble for marker or not
object	WKT object	A WKT object to set position of Marker
info	InfoBubble object	A InfoBubble object which is set for the Marker
icon	Icon object	A Icon object which is set for the Marker

CHAPTER

TWENTYTHREE

OVERLAY OBJECT

`Overlay` is an `here-map-widget-for-jupyter` class that allows you to visual a rectangular area on a Map in the form of a bitmap.

23.1 Example

```
from here_map_widget import Map, Overlay, Rectangle, Bbox, FullscreenControl
from ipywidgets import Image
import os

m = Map(api_key=os.environ["LS_API_KEY"])
m.center = [53.1, 13.1]
m.zoom = 3
bbox = Bbox(
    top=70.72849153520343,
    left=-24.085683364175395,
    bottom=29.569664922291,
    right=44.216452317817016,
)

overlay = Overlay(
    boundingBox=bbox,
    bitmap="https://heremaps.github.io/"
    + "maps-api-for-javascript-examples/image-overlay/data/0.png",
    volatility=True,
)
m.add_object(overlay)
m.add_control(FullscreenControl())
m
```

23.2 Attributes

Attribute	Type	Doc
content	string	Optional arbitrary data to be stored with the map overlay
boundingBox	Bbox object	A rectangular area of the overlay defined in terms of the geographical coordinates of its top-left and bottom-right corners
bitmap	string	An image URL, an SVG image (markup), a bitmap image or a canvas
min	float	The minimum zoom level at which the object is visible, the default is -Infinity
max	float	The maximum zoom level at which the object is visible, the default is Infinity
opacity	float	The opacity of the object in range from 0 (transparent) to 1 (opaque), the default is 1
visibility	boolean	Indicates whether the map object is visible, the default is True
volatility	boolean	Indicates whether the map object is volatile, the default is False

23.3 Methods

Method	Arguments	Doc
set_bitmap	string	Set bitmap for Overlay

CHAPTER

TWENTYFOUR

POLYGON OBJECT

`Polygon` is an `here-map-widget-for-jupyter` class that allows you to visualize LineString, WKT, GeoPolygon and GeoMultiPolygon as Polygon on the Map. Style of object is a dictionary, to get more information on all possible keys of style dictionary example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10, 'lineCap': 'square', lineJoin: 'bevel'}.

24.1 Example

```
from here_map_widget import Map, GeoPolygon, LineString, Polygon
import os

center = [51.1657, 10.4515]

l = [52, 13, 100, 48, 2, 100, 48, 16, 100, 52, 13, 100]

ls = LineString(points=l)

gpg = GeoPolygon(linestring=ls)

style = {"strokeColor": "#829", "lineWidth": 4}

obj = Polygon(object=gpg, style=style, draggable=False)

m = Map(api_key=os.environ["LS_API_KEY"], center=center)

m.add_object(obj)
m
```

24.2 Attributes

Attribute	Type	Doc
object	object of LineString or WKT or GeoPolygon or GeoMultiPolygon	The geometry that defines the surface of the polygon
style	dict	The style to be used when tracing the spatial object
draggable	boolean	To make a draggable polygon
extrusion	float	The extrusion height for the polygon in meters, default is 0
elevation	float	The elevation height of the polygon in meters, default is 0

CHAPTER

TWENTYFIVE

POLYLINE OBJECT

`Polyline` is an `here-map-widget-for-jupyter` class that allows you to visualize a LineString, MultiLineString and WKT as Line on the Map. Style of object is a dictionary, to get more information on all possible keys of style dictionary, example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10, 'lineCap': 'square', lineJoin: 'bevel'}.

25.1 Example

```
from here_map_widget import LineString, Polyline
from here_map_widget import Map
import os

m = Map(api_key=os.environ["LS_API_KEY"])
m.center = [51.1657, 10.4515]
style = {"lineWidth": 15}
l = [53.3477, -6.2597, 0, 51.5008, -0.1224, 0, 48.8567, 2.3508, 0, 52.5166, 13.3833, 0]
ls = LineString(points=l)
pl = Polyline(object=ls, style=style)
m.add_object(pl)
m
```

25.2 Attributes

Attribute	Type	Doc
object	object of LineString or WKT or MultiLineString	The geometry that defines the line segments of the polyline
style	dict	The style to be used when tracing the polyline
draggable	boolean	To make a draggable polyline

CHAPTER

TWENTYSIX

RECTANGLE OBJECT

`Rectangle` is an `here-map-widget-for-jupyter` class that allows you to visualize a Bounding Box as Rectangle on the Map. Style of object is a dictionary, to get more information on all possible keys of style dictionary, example: {'fillColor': 'rgba(245, 176, 65, 0.5)', 'strokeColor': 'black', 'lineWidth': 10, 'lineCap': 'square', lineJoin: 'bevel'}.

26.1 Example

```
from here_map_widget import Map
from here_map_widget import Bbox, Rectangle
import os

center = [46.017618898512374, 23.91982511231513]

m = Map(api_key=os.environ["LS_API_KEY"], center=center, zoom=3)

style = {"strokeColor": "#829", "lineWidth": 4}

bbox = Bbox(top=53.1, left=13.1, bottom=43.1, right=40.1)
rectangle = Rectangle(bbox=bbox, style=style, draggable=True)
m.add_object(rectangle)
m
```

26.2 Attributes

Attribute	Type	Doc
bbox	object of Bbox	The geographical bounding box for the rectangle
style	dict	The style to be used when tracing the spatial object
draggable	boolean	To make a draggable rectangle
extrusion	float	The extrusion height for the rectangle in meters, default is 0
elevation	float	The elevation height of the rectangle in meters, default is 0

CHAPTER

TWENTYSEVEN

POINT

A `Point` is subclass of `Geometry` class which can be used in multiple objects. One of the example is to define the center of the Circle.

27.1 Example

```
from here_map_widget import Map
from here_map_widget import Point, Circle
import os

m = Map(api_key=os.environ['LS_API_KEY'], center=[19.152761, 72.87869], zoom=10)

style = {"strokeColor": "#829", "lineWidth": 4}

point = Point(lat=19.152761, lng=72.87869)
circle = Circle(center=point, radius=10000, style=style, draggable=True)
m.add_object(circle)
m
```

27.2 Attributes

Attribute	Type	Doc
lat	Float	Latitude of the Point.
lng	Float	Longitude of the Point.
alt	Float	Altitude of the Point.

CHAPTER

TWENTYEIGHT

LINESTRING

`LineString` is subclass of `Geometry` class which then can be used to create a `Polyline` which will be visualised on the Map.

28.1 Example

```
import os
from here_map_widget import LineString, Polyline
from here_map_widget import Map

m = Map(api_key=os.environ["LS_API_KEY"])
m.center = [51.1657, 10.4515]
style = {"lineWidth": 15}
l = [53.3477, -6.2597, 0, 51.5008, -0.1224, 0, 48.8567, 2.3508, 0, 52.5166, 13.3833, 0]
ls = LineString(points=l)
pl = Polyline(object=ls, style=style)
m.add_object(pl)
m
```

28.2 Attributes

Attribute	Type	Doc
points	A list of Point objects.	A list of Point objects to construct Linestring.

28.3 Methods

Method	Arguments	Doc
push_point	lat,lng,alt	Pushes Point with given lat,lng,alt at the end of LineString.

CHAPTER

TWENTYNINE

MULTILINESTRING

`MultiLineString` is subclass of `Geometry` class which then can be used to create a `Polyline` which will be visualised on the Map.

29.1 Example

```
from here_map_widget import Map
from here_map_widget import LineString, MultiLineString, Polyline
import os

m = Map(api_key=os.environ['LS_API_KEY'], center=[51.1657, 10.4515])
style = { 'lineWidth': 15 }
l = [53.3477, -6.2597, 0, 51.5008, -0.1224, 0, 48.8567, 2.3508, 0, 52.5166, 13.3833, 0]
l_1 = [-53.3477, 6.2597, 0, -51.5008, 0.1224, 0, 48.8567, 2.3508, 0, 52.5166, 13.3833, 0]
ls = LineString(points=l)
ls_1 = LineString(points=l_1)
ml = MultiLineString(lines=[ls])
pl = Polyline(object=ml, style=style)
m.add_object(pl)
m
```

29.2 Attributes

Attribute	Type	Doc
lines	A list of LineString objects.	A list of LineString objects.

Method	Arguments	Doc
push_line	LineString object	Pushes Linestring end of MultiLineString.
add_lines	list of LineString objects	Pushes multiple Linestrings end of MultiLineString.
remove_line	LineString object	Removes LineString from MultiLineString.

CHAPTER

THIRTY

GEOPOLYGON

`GeoPolygon` is a subclass of Geometry class that allows you to create a `Polygon` object which then can be visualised on the Map.

30.1 Example

```
import os
from here_map_widget import Map, GeoPolygon, LineString, Polygon

l = [52, 13, 100, 48, 2, 100, 48, 16, 100, 52, 13, 100]

ls = LineString(points=l)

gpg = GeoPolygon(linestring=ls)

style = {"strokeColor": "#829", "lineWidth": 4}

obj = Polygon(object=gpg, style=style, draggable=False)

m = Map(api_key=os.environ["LS_API_KEY"], center=[51.1657, 10.4515])

m.add_object(obj)
m
```

30.2 Attributes

Attribute	Type	Doc
linestring	LineString object.	The LineString Geometry.
holes	A list of LineString objects.	A list of LineStrings.

30.3 Methods

Method	Arguments	Doc
push_hole	LineString object	Add hole on GeoPolygon.
add_holes	list of LineString objects	Add holes on GeoPolygon
remove_hole	LineString object	Remove hole from the GeoPolygon

CHAPTER

THIRTYONE

GEOMULTIPOLYGON

GeoMultiPolygon is a subclass of Geometry class that allows you to create a Polygon object which then can be visualised on the Map.

31.1 Example

```
import os
from here_map_widget import Map, GeoPolygon, LineString, Polygon, GeoMultiPolygon

m = Map(api_key=os.environ["LS_API_KEY"], center=[51.1657, 10.4515])

l1 = [52, 13, 100, 48, 2, 100, 48, 16, 100, 52, 13, 100]

ls1 = LineString(points=l1)

gpg1 = GeoPolygon(linestring=ls1)

style = {"strokeColor": "#829", "lineWidth": 4}

l2 = [55, 19, 99, 52, 2, 100, 52, 16, 100, 55, 19, 99]
ls2 = LineString(points=l2)

gpg2 = GeoPolygon(linestring=ls2)

gmp = GeoMultiPolygon(polygons=[gpg1])

pg = Polygon(object=gmp, style=style, draggable=False)

m.add_object(pg)
m
```

31.2 Attributes

Attribute	Type	Doc
polygons	A list of GeoPolygon objects.	A list of GeoPolygon objects.

31.3 Methods

Method	Arguments	Doc
push_polygon	GeoPolygon object	Add polygon in the GeoMultiPolygon
add_polygons	list of GeoPolygon objects	Add polygons in the GeoMultiPolygon
remove_polygon	GeoPolygon object	Remove polygon from the GeoMultiPolygon

CHAPTER

THIRTYTWO

BBOX

A `bbox` is a subclass of `Geometry` class which is used to create a `Rectangle` object which can then be visualised on the Map.

32.1 Example

```
import os
from here_map_widget import Map
from here_map_widget import Bbox, Rectangle


m = Map(
    api_key=os.environ["LS_API_KEY"],
    center=[46.017618898512374, 23.91982511231513],
    zoom=3,
)

style = {"strokeColor": "#829", "lineWidth": 4}

bbox = Bbox(top=53.1, left=13.1, bottom=43.1, right=40.1)
rectangle = Rectangle(bbox=bbox, style=style, draggable=True)
m.add_object(rectangle)
m
```

32.2 Attributes

Attribute	Type	Doc
top	Float	A value indicating the northern-most latitude.
left	Float	A value indicating the left-most longitude.
bottom	Float	A value indicating the southern-most latitude.
right	Float	A value indicating the right-most latitude.

CHAPTER

THIRTYTHREE

WKT

A `WKT` is a subclass of `Geometry` class which will be used to visualise WKT data on the Map.

33.1 Example

```
import os
from here_map_widget import WKT, Marker
from here_map_widget import Map

m = Map(api_key=os.environ["LS_API_KEY"])
m.center = [19.0760, 72.8777]
m.zoom = 9
wkt = WKT(data="POINT (72.8777 19.0760)")
mumbai_marker = Marker(object=wkt)
m.add_object(mumbai_marker)
m
```

33.2 Attributes

Attribute	Type	Doc
data	string	A WKT string.

CHAPTER

THIRTYFOUR

INFO BUBBLE ELEMENT

`Info Bubble` is an `here-map-widget-for-jupyter` class that allows to make a information bubble and bound it to a geographic position on the Map.

34.1 Example

```
from here_map_widget import Map, Marker, Point
from here_map_widget import InfoBubble
import os

m = Map(api_key=os.environ["LS_API_KEY"], zoom=8)
m.center = [19.0760, 72.8777]
m.zoom = 8

info = InfoBubble(
    position=Point(lat=18.9389, lng=72.8258),
    content='<div><a href="https://mumbaicricket.com"'
    + 'target="_blank">Mumbai Cricket Association</a> </div>'
    + "<div >Wankhede Stadium<br>Capacity: 33,108</div>",
)

mumbai_marker = Marker(
    lat=18.9389, lng=72.8258, info=info, evt_type="tap", show_bubble=True
)

m.add_object(mumbai_marker)
m
```

34.2 Attributes

Attribute	Type	Doc
content	string	The content to be added to the info bubble
position	object of Point	The geographic location to which this info bubble corresponds

CHAPTER

THIRTYFIVE

STYLE ELEMENT

`Style` is an `here-map-widget-for-jupyter` class that allows to configure style for various elements of the Map.

35.1 Example

```
from here_map_widget import Map, Style
import os

style = Style(
    config="https://heremaps.github.io/maps-api-for-javascript-examples/"
    + "change-style-at-load/data/dark.yaml",
    base_url="https://js.api.here.com/v3/3.1/styles/omv/",
)


m = Map(
    api_key=os.environ["LS_API_KEY"],
    style=style,
    center=[52.51477270923461, 13.39846691425174],
    zoom=13,
)

m
```

35.2 Attributes

Attribute	Type	Doc
config	string	Either a URL to load the style from, YAML formatted string or an object describing the rendering style
base_url	string	The base URL to use for resolving relative URLs in the style like textures, fonts

CHAPTER

THIRTYSIX

FULLSCREEN CONTROL

`Fullscreen Control` is an `here-map-widget-for-jupyter` class that allows you to add a control which contains a button that will put the Map in full-screen when clicked.

36.1 Example

```
from here_map_widget import Map, FullscreenControl
import os

m = Map(api_key=os.environ["LS_API_KEY"])
m.center = [44.20022717941052, -72.75660780639646]
fs = FullscreenControl()
m.add_control(fs)
m
```

36.2 Attributes

Attribute	Type	Doc
name	string	Unique id of the widget, default value is FullscreenControl
alignment	string	The layout alignment which should be applied to the `Fullscreen Control`, please check *allowed values*, defaults to `TOP_LEFT`.

CHAPTER

THIRTYSEVEN

MAP SETTINGS CONTROL

`Map Settings Control` is an `here-map-widget-for-jupyter` class that allows you to add a control that allows the user to select the base map types as well as add additional layers on top.

37.1 Example

```
from here_map_widget import GeoJSON, Map, MapSettingsControl
from here_map_widget import TileLayer, XYZ
import os

center = [51.1657, 10.4515]

m = Map(api_key=os.environ["LS_API_KEY"], center=center)

xyz_token = os.environ["XYZ_TOKEN"]
provider = XYZ(space_id="m2pcsiNi", token=xyz_token)
space = TileLayer(provider=provider)

geojson = GeoJSON(
    url="https://gist.githubusercontent.com/peaksnail/"
    + "5d4f07ca00ed7c653663d7874e0ab8e7/raw/"
    + "64c2a975482efd9c42e54f6f6869f091055053cd/countries.geo.json",
    disable_legacy_mode=True,
    style={"color": "black", "opacity": 0.1},
)


setttings = MapSettingsControl(
    layers=[
        {"label": "space", "layer": space},
        {"label": "countries", "layer": geojson},
    ],
    basemaps=["raster.satellite.map", "raster.terrain.map"],
)

m.add_control(setttings)

m
```

37.2 Attributes

Attribute	Type	Doc
alignment	string	The layout alignment which should be applied to the `Map Settings Control`, please check *allowed values*, defaults to `TOP_RIGHT`.
basemaps	List of strings	The list of base layers to be shown in the map settings control, selecting an entry changes map base layer
layers	List of Layers	The list of layers to be shown in the map settings control after the baseLayers list

37.3 Methods

Method	Arguments	Doc
add_layers	List	Add layers to control
remove_layers	List	Remove layers from control

CHAPTER

THIRTYEIGHT

MEASUREMENT CONTROL

`Measurement Control` is an `here-map-widget-for-jupyter` class that allows you to visualize a distance measurement control on the Map.

38.1 Example

```
from here_map_widget import Map, MeasurementControl, Icon, Marker
import os

m = Map(api_key=os.environ["LS_API_KEY"])
m.center = [44.20022717941052, -72.75660780639646]
start_markup = '<svg width="20" height="20" version="1.1" xmlns="http://www.w3.org/2000/
↪svg">'
start_markup += '<circle cx="10" cy="10" r="7" fill="transparent" stroke="green" stroke-
↪width="4"/>'
start_markup += "</svg>"

stopover_markup = '<svg width="20" height="20" version="1.1" xmlns="http://www.w3.org/
↪2000/svg">'
stopover_markup += '<circle cx="10" cy="10" r="7" fill="transparent" stroke="yellow"␣
↪stroke-width="4"/>'
stopover_markup += "</svg>"

end_markup = '<svg width="20" height="20" version="1.1" xmlns="http://www.w3.org/2000/svg
↪">'
end_markup += '<circle cx="10" cy="10" r="7" fill="transparent" stroke="red" stroke-
↪width="4"/>'
end_markup += "</svg>"

split_markup = '<svg width="20" height="20" version="1.1" xmlns="http://www.w3.org/2000/
↪svg">'
split_markup += '<circle cx="10" cy="10" r="7" fill="transparent" stroke="orange" stroke-
↪width="4"/>'
split_markup += "</svg>"

start_icon = Icon(bitmap=start_markup, height=20, width=20, anchor={"x": 10, "y": 10})
stopover_icon = Icon(
    bitmap=stopover_markup, height=20, width=20, anchor={"x": 10, "y": 10}
)
```

(continues on next page)

(continued from previous page)

```
end_icon = Icon(bitmap=end_markup, height=20, width=20, anchor={"x": 10, "y": 10})
split_icon = Icon(bitmap=split_markup, height=20, width=20, anchor={"x": 10, "y": 10})

mc = MeasurementControl(
    start_icon=start_icon,
    stopover_icon=stopover_icon,
    end_icon=end_icon,
    split_icon=split_icon,
)

m.add_control(mc)
m
```

38.2 Attributes

Attribute	Type	Doc
name	string	Unique id of the widget, default value is MeasurementControl
align-ment	string	The layout alignment which should be applied to the `Measurement Control`, please check *allowed values*, defaults to `RIGHT_BOTTOM`
start_icon	object of Icon	The icon to use for the first measurement point
stopover_icon	object of Icon	The icon to use for the intermediate measurement points
end_icon	object of Icon	The icon to use for the last measurement point
split_icon	object of Icon	The icon to use to indicate the position under the pointer, over the line where a new point will be created once user clicks

CHAPTER

THIRTYNINE

SCALE BAR CONTROL

`Scale Bar Control` is an `here-map-widget-for-jupyter` class that represents a UI element that shows the zoom scale.

39.1 Example

```
from here_map_widget import Map, ScaleBar
import os

m = Map(api_key=os.environ["LS_API_KEY"])
sb = ScaleBar()
m.add_control(sb)
m
```

39.2 Attributes

Attribute	Type	Doc
alignment	string	The layout alignment which should be applied to the `Scale Bar Control`, please check *allowed values*, defaults to `RIGHT_BOTTOM`

CHAPTER

FORTY

SEARCH CONTROL

`Search Control` is an `here-map-widget-for-jupyter` class that allows you to search address and data on the Map.

40.1 Example

Search using HERE Geocoding Service.

```
from here_map_widget import Map, SearchControl, Marker
import os

center = [19.0760, 72.8777]
m = Map(api_key=os.environ['LS_API_KEY'], center=center)

marker = Marker(lat=center[0], lng=center[1])
sc = SearchControl(marker=marker, zoom=5)

m.add_control(sc)

m
```

40.2 GeoJSON Example

User can also search features in GeoJSON layer.

```
from here_map_widget import Map, SearchControl, Marker, GeoJSON
import os

center = [19.0760, 72.8777]
M = Map(api_key=os.environ["LS_API_KEY"], center=center)


geojson = GeoJSON(
    url="https://gist.githubusercontent.com/peaksnail/"
    + "5d4f07ca00ed7c653663d7874e0ab8e7/raw/64c2a975482efd9c42e54f6f6869f091055053cd/"
    + "countries.geo.json",
    disable_legacy_mode=True,
    style={"color": "black", "opacity": 0.1},
```

(continues on next page)

(continued from previous page)

```
)

marker = Marker(lat=center[0], lng=center[1])

sc = SearchControl(marker=marker, zoom=5, layer=geojson, property_name="name")

M.add_control(sc)

M
```

40.3 Attributes

Attribute	Type	Doc
name	string	Unique id of the widget, default value is SearchControl
alignment	string	The layout alignment which should be applied to the `Search Control`, please check *allowed values*, defaults to `TOP_LEFT`
zoom	float	Zoom level to be set for search result, default value is 4
property_name	string	Property name to be used for search in the layer provided, default value is *name*
lang	string	Select the language to be used for result rendering from a list of BCP47 compliant Language Codes.
limit	int	Maximum number of results to be returned, default 10.
marker	object of Marker	Marker which is set for search result
layer	object of Layer	Layer to be searched using the control

40.4 Methods

Method	Arguments	Doc
on_feature_found	Callable object	Adds a callback on found event for searching in GeoJSON layer.

CHAPTER

FORTYONE

SPLIT MAP CONTROL

`Split Map Control` is an `here-map-widget-for-jupyter` class that allows comparing layers by splitting the map in two.

41.1 Example

```
from here_map_widget import Map, SplitMapControl, GeoJSON
import os

m = Map(api_key=os.environ["LS_API_KEY"])
left_geojson = GeoJSON(
    url="https://gist.githubusercontent.com/peaksnail/"
    + "5d4f07ca00ed7c653663d7874e0ab8e7/raw/64c2a975482efd9c42e54f6f6869f091055053cd/"
    + "countries.geo.json",
    disable_legacy_mode=True,
    style={"fillColor": "#ff0000", "color": "black", "opacity": 0.1},
)
right_geojson = GeoJSON(
    url="https://gist.githubusercontent.com/peaksnail/"
    + "5d4f07ca00ed7c653663d7874e0ab8e7/raw/64c2a975482efd9c42e54f6f6869f091055053cd/"
    + "countries.geo.json",
    disable_legacy_mode=True,
    style={"fillColor": "#00ff00", "color": "black", "opacity": 0.1},
)

sp = SplitMapControl(left_layer=left_geojson, right_layer=right_geojson)
m.add_control(sp)
m
```

41.2 Attributes

Attribute	Type	Doc
name	string	Unique id of the widget, default value is SplitMapControl
left_layer	object of layer	Layer to be added on left side of the control
right_layer	object of layer	Layer to be added on right side of the control

Warning: `Split Map Control` is standalone control and can not be used with any other control.

CHAPTER

FORTYTWO

ZOOM CONTROL

`Zoom Control` is an `here-map-widget-for-jupyter` class that represents the UI control that allows the user to change the map zoom level.

42.1 Example

```
from here_map_widget import Map, ZoomControl, GeoJSON
import os

m = Map(api_key=os.environ['LS_API_KEY'])
zc = ZoomControl(alignment='BOTTOM_RIGHT', slider=True)
m.add_control(zc)
m
```

42.2 Attributes

Attribute	Type	Doc
zoomSpeed	float	The zoom speed in levels per second, defaults to 4
fractionalZoom	boolean	A flag indicating whether fractional zoom levels are allowed (True, default) or not (False).
alignment	string	The layout alignment which should be applied to the `Zoom Control`, please check *allowed values*, defaults to `RIGHT_MIDDLE`
slider	boolean	A flag indicating whether to show the slider (True) or not (False, default)
sliderSnaps	boolean	A flag indicating whether the slider should snap to integer values or not, defaults to False

CHAPTER

FORTYTHREE

ZOOM RECTANGLE CONTROL

`Zoom Rectangle Control` is an `here-map-widget-for-jupyter` class that represents a zoom rectangle control element that allows zooming to the selected area on the screen.

43.1 Example

```
from here_map_widget import Map, ZoomRectangle
import os

m = Map(api_key=os.environ['LS_API_KEY'])
zr = ZoomRectangle(alignment='BOTTOM_RIGHT')
m.add_control(zr)
m
```

43.2 Attributes

Attribute	Type	Doc
alignment	string	The layout alignment which should be applied to the `Zoom Rectangle Control`, please check *allowed values*, defaults to `BOTTOM_RIGHT`

CHAPTER

FORTYFOUR

WIDGET CONTROL

`Widget Control` is an `here-map-widget-for-jupyter` class that allows integration of various ipywidgets with `here-map-widget-for-jupyter`.

44.1 Example

```
from here_map_widget import WidgetControl, Map
from ipywidgets import FloatSlider, ColorPicker, jslink
import os

m = Map(api_key=os.environ['LS_API_KEY'])

zoom_slider = FloatSlider(description='Tilt level:', min=0, max=90, value=0)
jslink((zoom_slider, 'value'), (m, 'tilt'))
widget_control1 = WidgetControl(widget=zoom_slider, alignment="TOP_RIGHT", name=
↪"FloatSlider")

m.add_control(widget_control1)
m
```

44.2 Attributes

Attribute	Type	Doc
name	string	Unique id of the widget, default value is SplitMapControl
alignment	string	The layout alignment which should be applied to the `Widget Control`, please check *allowed values*, defaults to `RIGHT_BOTTOM`
widget	object of ipywidget	The ipywidget to be added
transparent_bg	boolean	If set to `True`, widget will be added with transparent background. Default False.

CHAPTER

FORTYFIVE

LAYERS CONTROL

`LayersControl` is an `here-map-widget-for-jupyter` class which is useful to show/hide layers on map.

45.1 Example

```
from here_map_widget import GeoJSON, Map, LayersControl
import os

m = Map(api_key=os.environ['LS_API_KEY'], center=[43.052, -62.49])

countries = GeoJSON(
url="https://gist.githubusercontent.com/peaksnail/5d4f07ca00ed7c653663d7874e0ab8e7/raw/
↪64c2a975482efd9c42e54f6f6869f091055053cd/countries.geo.json",
disable_legacy_mode=True,
style={"color": "black", "opacity": 0.1},
name="world countries",
)

us_states = GeoJSON(
    url="https://raw.githubusercontent.com/PublicaMundi/MappingAPI/master/data/geojson/
↪us-states.json",
    disable_legacy_mode=True,
    style={"color": "black", "opacity": 0.1},
    name="US states",
)
m.add_layers([countries, us_states])
lc = LayersControl(alignment="RIGHT_TOP")
m.add_control(lc)
m
```

45.2 Attributes

Attribute	Type	Doc
alignment	string	The layout alignment which should be applied to the `LayersControl`, please check *allowed values*, defaults to `RIGHT_TOP`

CHAPTER

FORTYSIX

DEFAULT LAYERS NAMES

`DefaultLayers` is a pre-configured set of HERE layers for convenient use as basemap. `DefaulLayerNames` is config of various `layer_name` s which is used to instantiate `DefaultLayers`.

46.1 Vector Layer names example

```
from here_map_widget import DefaultLayerNames

DefaultLayerNames.vector.normal.map
```

```
'vector.normal.map'
```

46.2 Vector Layer Names

Object	Type	Value
DefaultLayerNames.vector.normal.map	string	vector.normal.map
DefaultLayerNames.vector.normal.truck	string	vector.normal.truck
DefaultLayerNames.vector.normal.traffic	string	vector.normal.traffic
DefaultLayerNames.vector.normal.trafficincidents	string	vector.normal.trafficincidents

46.3 Raster Layer names example

```
from here_map_widget import DefaultLayerNames

DefaultLayerNames.raster.normal.map
```

```
'raster.normal.map'
```

46.4 Raster Layer Names

Object	Type	Value
DefaultLayerNames.raster.normal.map	string	raster.normal.map
DefaultLayerNames.raster.normal.mapnight	string	raster.normal.mapnight
DefaultLayerNames.raster.normal.xbase	string	raster.normal.xbase
DefaultLayerNames.raster.normal.xbasenight	string	raster.normal.xbasenight
DefaultLayerNames.raster.normal.base	string	raster.normal.base
DefaultLayerNames.raster.normal.basenight	string	raster.normal.basenight
DefaultLayerNames.raster.normal.trafficincidents	string	raster.normal.trafficincidents
DefaultLayerNames.raster.normal.transit	string	raster.normal.transit
DefaultLayerNames.raster.normal.labels	string	raster.normal.labels
DefaultLayerNames.raster.satellite.map	string	raster.satellite.map
DefaultLayerNames.raster.satellite.xbase	string	raster.satellite.xbase
DefaultLayerNames.raster.satellite.base	string	raster.satellite.base
DefaultLayerNames.raster.terrain.map	string	raster.terrain.map
DefaultLayerNames.raster.terrain.xbase	string	raster.terrain.xbase
DefaultLayerNames.raster.terrain.base	string	raster.terrain.base
DefaultLayerNames.raster.terrain.labels	string	raster.terrain.labels

CHAPTER

FORTYSEVEN

SERVICE NAMES

Config for service names is maintained in `ServiceNames`.

47.1 Service Names Example

```
from here_map_widget import Map, Platform, MapTile
from here_map_widget import ServiceNames, MapTileUrl
import os

services_config = {
    ServiceNames.maptile: {
        MapTileUrl.scheme: "https",
        MapTileUrl.host: "maps.ls.hereapi.com",
        MapTileUrl.path: "maptile/2.1",
    }
}

platform = Platform(api_key=os.environ["LS_API_KEY"], services_config=services_config)
```

47.2 Service Names

Object	Type	Value
ServiceNames.omv	string	omv
ServiceNames.maptile	string	maptile
ServiceNames.xyz	string	xyz

CHAPTER

FORTYEIGHT

SERVICE URLS

Config for service urls is maintained in respective classes of services `OMVUrl`, `MapTileUrl` and `XYZUrl`.

48.1 OMV Url Example

```
from here_map_widget import Platform
from here_map_widget import ServiceNames, OMVUrl
import os

services_config = {
    ServiceNames.omv: {
        OMVUrl.scheme: "https",
        OMVUrl.host: "vector.hereapi.com",
        OMVUrl.path: "/v2/vectortiles/core/mc",
    }
}

platform = Platform(api_key=os.environ["LS_API_KEY"], services_config=services_config)
```

48.2 OMV Url Attributes

Object	Type	Value
OMVUrl.scheme	string	scheme
OMVUrl.host	string	host
OMVUrl.path	string	path

48.3 MapTile Url Example

```
from here_map_widget import Map, Platform, MapTile
from here_map_widget import ServiceNames, MapTileUrl
import os

services_config = {
    ServiceNames.maptile: {
```

(continues on next page)

(continued from previous page)

```
        MapTileUrl.scheme: "https",
        MapTileUrl.host: "maps.ls.hereapi.com",
        MapTileUrl.path: "maptile/2.1",
    }
}

platform = Platform(api_key=os.environ["LS_API_KEY"], services_config=services_config)
```

48.4 MapTile Url Attributes

Object	Type	Value
MapTileUrl.scheme	string	scheme
MapTileUrl.host	string	host
MapTileUrl.path	string	path

48.5 XYZ Url Example

If XYZ Hub or HERE Data Hub is hosted on your local machine.

```
from here_map_widget import TileLayer, XYZ
from here_map_widget import Map, Platform, ServiceNames, XYZUrl
import os

m = Map(api_key=os.environ["LS_API_KEY"], center=[36.59, -98.70], zoom=2)

services_config = {
    ServiceNames.xyz: {
        XYZUrl.scheme: "http",
        XYZUrl.host: "localhost:8080",
        XYZUrl.path: "/hub/spaces",
    }
}

platform = Platform(api_key=os.environ["LS_API_KEY"], services_config=services_config)
provider = XYZ(space_id="YOUR-SPACE-ID", platform=platform)
space_layer = TileLayer(provider=provider)
m.add_layer(space_layer)
m
```

48.6 XYZ Url Attributes

Object	Type	Value
XYZUrl.scheme	string	scheme
XYZUrl.host	string	host
XYZUrl.path	string	path

CHAPTER

FORTYNINE

MISCELLANEOUS VALUES

This section describes values for input parameters of `controls` and `layers` and their description.

49.1 Control Alignment

`alignment` is input parameter for all `controls` it determines where control will be postioned on the map.

Name	Description
TOP_LEFT	control will be added on top of the map on the left corner and control will be horizontal.
TOP_CENTER	control will be added on top of the map on center and control will be horizontal.
TOP_RIGHT	control will be added on top of the map on the right corner and control will be horizontal.
LEFT_TOP	control will be added on the left side of the map and on top and control will be vertical.
LEFT_MIDDLE	control will be added on the left side of the map and in the middle and control will be vertical.
LEFT_BOTTOM	control will be added on the left side of the map and in the bottom and control will be vertical.
RIGHT_TOP	control will be added on the right side of the map and on top and control will be vertical.
RIGHT_MIDDLE	control will be added on the right side of the map and in the middle and control will be vertical.
RIGHT_BOTTOM	control will be added on the right side of the map and in the bottom and control will be vertical.
BOTTOM_LEFT	control will be added on bottom of the map on the left corner and control will be horizontal.
BOTTOM_CENTER	control will be added on bottom of the map on center and control will be horizontal.
BOTTOM_RIGHT	control will be added on bottom of the map on the right corner and control will be horizontal.

49.2 Event Type

`evt_type` is input parameter for some `layers`. Event listeners listen to this type of event.

Name	Description
tap	Represents mouse click by the user.
pointermove	Represents mosue hover by the user.

CHAPTER

FIFTY

CHANGELOG

50.1 here-map-widget-for-jupyter 1.1.2 (2021-08-19)

- Added support for external basemaps using xyzservices.
- Removed support for python3.6 due to xyzservices.

50.2 here-map-widget-for-jupyter 1.1.1 (2021-06-16)

- Added bounds trait to map object
- Added point_style for GeoJSON layer
- Fixed JS bundle load issue for unpkg.com

50.3 here-map-widget-for-jupyter 1.1.0 (2021-06-09)

- Added transparent_bg parameter in WidgetControl
- Added dedicated LayersControl to toggle layers on the map.
- Fix layout issue of ipywidgets
- Fix basemap trait for Map object
- Fix MapSettingsControl dynamic update

50.4 here-map-widget-for-jupyter 1.0.1 (2021-03-31)

- Add `ImageTileProvider` to support XYZ and WMTS Tile layer.
- Add support for += and -= operators to add and remove Layers, Controls, and Objects.

50.5 here-map-widget-for-jupyter 1.0.0 (2021-02-26)

- Add support for JupyterLab 3
- Add support for custom XYZ URL for XYZ space layer.

50.6 here-map-widget-for-jupyter 0.1.1 (2021-02-08)

- Added binder setup for the example notebooks.
- Fixed readthedocs build.
- Updated README.

50.7 here-map-widget-for-jupyter 0.1.0 (2021-02-04)

- Initial release

CHAPTER

FIFTYONE

CONTRIBUTING TO HERE MAP WIDGET FOR JUPYTER

Thank you for taking the time to contribute.

The following is a set of guidelines for contributing to this package. These are mostly guidelines, not rules. Use your best judgement and feel free to propose changes to this document in a pull request.

51.1 Coding Guidelines

1. Lint your code contributions as per pep8 guidelines. To help you out, we have included a *Makefile* in the root directory which supports the commands below:

   Autoformat code using black:

   ```
   make black
   ```

   Check for linting errors:

   ```
   make lint
   ```

2. Sort the imports in each python file as per pep8 guidelines for import Please execute the isort utility to have the imports sorted auto-magically.

51.2 Notebooks

Example notebooks are provided in /examples.

51.3 Signing each Commit

When you file a pull request, we ask that you sign off the Developer Certificate of Origin (DCO) in each commit. Any Pull Request with commits that are not signed off will be rejected by the DCO check.

A DCO is a lightweight way to confirm that a contributor wrote or otherwise has the right to submit code or documentation to a project. Simply add *Signed-off-by* as shown in the example below to indicate that you agree with the DCO.

The git flag *-s* can be used to sign a commit:

```
git commit -s -m 'README.md: Fix minor spelling mistake'
```

The result is a signed commit message:

README.md: Fix minor spelling mistake

Signed-off-by: John Doe <john.doe@example.com>

51.4 Development

For a development installation (requires yarn, you can install it with `conda install -c conda-forge yarn`):

```
$ git clone https://github.com/heremaps/here-map-widget-for-jupyter.git
$ cd here-map-widget-for-jupyter
$ pip install -e .
```

If you are using the classic Jupyter Notebook you need to install the nbextension:

```
$ jupyter nbextension install --py --sys-prefix here_map_widget
$ jupyter nbextension enable here_map_widget --py --sys-prefix
```

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 developing with JupyterLab:

```
$ jupyter labextension develop here_map_widget
```