

```
//IMPORTANT INFORMATION:  
//The JavaScript component is the main component of the code.  
//The variable data is the dataset containing the information to build the map and is already defined  
based on the Python code.  
//This is based on HTML and SVG. It only runs with the right HTML and SVG already existing.  
  
//A choropleth map is a map where shapes are filled in, such as countries in the map in this program.
```

```
loadMap();  
function loadMap(){  
  
    //Opens where the map should go in the SVG  
    var svg = document.getElementById("polygroup");  
    var svgMetadata = data["map"]["metadata"];  
    var entireSVG = document.getElementById("svgmap");  
  
    //Adjust height and width  
    entireSVG.setAttribute("height",svgMetadata["height"]);  
    entireSVG.setAttribute("width",svgMetadata["width"]);  
  
    //Map points  
    var svgPoints = data["map"]["data"];  
  
    //Choropleth map data  
    var choropleth = data["data"]["data"][data["choropleth"]["source"]];  
    var choropleth_min = Math.min.apply(null,choropleth);  
    var choropleth_max = Math.max.apply(null,choropleth);  
    var choropleth_range = choropleth_max - choropleth_min;  
  
    //Choropleth map colors  
    var choropleth_colors = data["choropleth"]["colors"];  
    var choropleth_red_min = choropleth_colors["r"][0];  
    var choropleth_red_max = choropleth_colors["r"][1];  
    var choropleth_green_min = choropleth_colors["g"][0];  
    var choropleth_green_max = choropleth_colors["g"][1];  
    var choropleth_blue_min = choropleth_colors["b"][0];  
    var choropleth_blue_max = choropleth_colors["b"][1];  
    var choropleth_red_range = choropleth_red_max - choropleth_red_min;  
    var choropleth_green_range = choropleth_green_max - choropleth_green_min;  
    var choropleth_blue_range = choropleth_blue_max - choropleth_blue_min;  
  
    //Ordered list of territories  
    var territory_list = data["data"]["territory_list"];  
  
    //Infobox  
    var infobox_styles = data["infobox"]["style"];  
    var infobox_styles_text = infobox_styles["text"];  
    var infobox_text = data["infobox"]["text"];  
  
    //Draw map  
    var polyPoints;  
    var jsonPoint;  
    var country;  
    var poly;  
  
    var rescaleY = svgMetadata["height"] / 180;  
    var rescaleX = svgMetadata["width"] / 360;  
    var translateY = svgMetadata["translate"][1];  
    var translateX = svgMetadata["translate"][0];  
    var zoom = svgMetadata["scale"];  
    var pointX;  
    var pointY;
```

```

    for (var countryData in svgPoints){
        polyPoints = ""
        for (jsonPoint in svgPoints[countryData]["Points"]){
            pointX = (svgPoints[countryData]["Points"][jsonPoint][0] - translateX) *
rescaleX * zoom;
            pointY = (180 / zoom - svgPoints[countryData]["Points"][jsonPoint][1] +
translateY) * rescaleY * zoom;
            polyPoints = polyPoints + pointX + "," + pointY + " ";
        }
        polyPoints = polyPoints.substring(0,polyPoints.length-1);
        poly = document.createElementNS("http://www.w3.org/2000/svg","polygon");
        poly.setAttributeNS(null,"points",polyPoints);
        poly.setAttributeNS(null,"class",svgPoints[countryData]["Country"]);
        poly.setAttributeNS(null,"style","fill:lime;stroke:purple;stroke-width:0.5");
        svg.appendChild(poly);
    }
    colorMap();
}
function colorMap(){

    //Choropleth map data
    var choropleth = data["data"]["data"][data["choropleth"]["source"]];
    var choropleth_min = Math.min.apply(null,choropleth);
    var choropleth_max = Math.max.apply(null,choropleth);
    var choropleth_range = choropleth_max - choropleth_min;

    //Choropleth map colors
    var choropleth_colors = data["choropleth"]["colors"];
    var choropleth_red_min = choropleth_colors["r"][0];
    var choropleth_red_max = choropleth_colors["r"][1];
    var choropleth_green_min = choropleth_colors["g"][0];
    var choropleth_green_max = choropleth_colors["g"][1];
    var choropleth_blue_min = choropleth_colors["b"][0];
    var choropleth_blue_max = choropleth_colors["b"][1];
    var choropleth_red_range = choropleth_red_max - choropleth_red_min;
    var choropleth_green_range = choropleth_green_max - choropleth_green_min;
    var choropleth_blue_range = choropleth_blue_max - choropleth_blue_min;

    //Ordered list of territories
    var territory_list = data["data"]["territory_list"];

    //Color map
    for (var j in territory_list){

        //Get country on map
        country = document.getElementsByClassName(territory_list[j]);

        //Scale data
        var data_proportional = (choropleth[j] - choropleth_min) / choropleth_range;

        //Calculate colors
        var red = data_proportional * choropleth_red_range + choropleth_red_min;
        var green = data_proportional * choropleth_green_range + choropleth_green_min;
        var blue = data_proportional * choropleth_blue_range + choropleth_blue_min;

        var color = "rgb(" + red + "," + green + "," + blue + ")";

        for (var i = 0; i < country.length; i++){
            country[i].style.fill = color;
            country[i].setAttribute("onmouseover",data["inputs"]["onHover"].replaceAll("[COUNTRY]",j));
            country[i].setAttribute("onmouseout",data["inputs"]
["onMouseOut"].replaceAll("[COUNTRY]",j));
            country[i].setAttribute("onclick",data["inputs"]["onClick"].replaceAll("

```

```

[COUNTRY"],j));
    }
}

function showInfobox(c){

    //Infobox
    var infobox_styles = data["infobox"]["style"];
    var infobox_styles_text = infobox_styles["text"];
    var infobox_text = data["infobox"]["text"];

    //Text
    var textPopUpParent = document.getElementById("dataInfo");
    var textPopUpList = [];
    var textPopUpText;

    //Sort text containers into lines
    for(var i in infobox_text){
        while(textPopUpList.length <= infobox_text[i]["line"]){
            textPopUpList.push([]);
        }
        textPopUpText = {};
        textPopUpText["text"] = infobox_text[i]["text"];
        textPopUpText["type"] = infobox_text[i]["type"];
        textPopUpText["style"] = infobox_styles_text[i];
        textPopUpList[infobox_text[i]["line"]].push(textPopUpText);
    }

    //Create text
    var textX;
    var textY;
    var numberLines = textPopUpList.length;
    var lineWidth;
    var ttlHeight = 0;
    var maxWidth = 0;
    var maxHeight;
    var bbox;
    var textColorHTML;
    var textColorDict;
    var font_size;
    var font;
    var iReverse;
    var textPopUpGroup = document.getElementById("dataInfo");
    var textPopUpDelete = document.getElementsByClassName("infoboxText")

    //Delete old text
    while(textPopUpDelete.length > 0){
        textPopUpDelete[0].parentNode.removeChild(textPopUpDelete[0]);
    }
    for(i in textPopUpList){
        lineWidth = 0;
        maxHeight = 0;
        iReverse = numberLines - 1 - i;
        for(var j in textPopUpList[iReverse]){
            textPopUp = document.createElementNS("http://www.w3.org/2000/svg","text");
            if(textPopUpList[iReverse][j]["type"] == "text"){
                textPopUp.innerHTML = textPopUpList[iReverse][j]["text"].replaceAll("
","&nbsp;");
            } else {
                textPopUp.innerHTML = eval(textPopUpList[iReverse][j]["text"]);
            }

            //Calculate x and y of text

```

```

        textX = event.pageX - 72 + lineWidth;
        textY = event.pageY - 65 - ttlHeight;

        //Style
        textColorDict = textPopUpList[iReverse][j]["style"];
        textColorHTML = "rgb(" + textColorDict["r"] + "," + textColorDict["g"] + ","
+ textColorDict["b"] + ");"
        font_size = textColorDict["size"];
        font_family = textColorDict["font"];

        //Set property and style
        textPopUp.setAttributeNS(null,"x",textX);
        textPopUp.setAttributeNS(null,"y",textY);
        textPopUp.setAttributeNS(null,"font-family",font_family);
        textPopUp.setAttributeNS(null,"font-size",font_size);
        textPopUp.setAttributeNS(null,"fill",textColorHTML);
        textPopUp.setAttributeNS(null,"class","infoboxText");
        textPopUpGroup.appendChild(textPopUp);

        //Adjust future elements by accounting for element width
        bbox = textPopUp.getBBox();
        lineWidth = lineWidth + bbox.width;
        if(bbox.height > maxHeight){
            maxHeight = bbox.height;
        }
    }
    if(lineWidth > maxWidth){
        maxWidth = lineWidth;
    }
    ttlHeight = ttlHeight + maxHeight;
}

//Box
var boxPopUp = document.getElementById("dataInfoBox");
boxPopUp.setAttribute("x",event.pageX-82);
boxPopUp.setAttribute("y",event.pageY-65-ttlHeight);
boxPopUp.setAttribute("height",ttlHeight+10);
boxPopUp.setAttribute("width",maxWidth+20);

//Set stroke width
boxPopUp["style"]["stroke-width"] = infobox_styles["border"]["stroke-width"];

//Fill colors
var red = infobox_styles["color"]["r"];
var green = infobox_styles["color"]["g"];
var blue = infobox_styles["color"]["b"];

var color = "rgb(" + red + "," + green + "," + blue + ")";
boxPopUp.setAttribute("fill",color);

//Border colors

red = infobox_styles["border"]["color"]["r"];
green = infobox_styles["border"]["color"]["g"];
blue = infobox_styles["border"]["color"]["b"];

color = "rgb(" + red + "," + green + "," + blue + ")";
boxPopUp.style.stroke = color;

//Make visible
boxPopUp.style.visibility = "visible";
}

function hideInfobox(c){
    var textPopUp = document.getElementsByClassName("infoboxText");

```

```
while(textPopUp.length > 0){  
    textPopUp[0].parentNode.removeChild(textPopUp[0]);  
}  
var boxPopUp = document.getElementById("dataInfoBox");  
boxPopUp.style.visibility = "hidden";  
}
```

#IMPORTANT: This is the Python library portion.  
#It is necessary for the Javascript portion to work properly.

#Shapefile reader built off PySHP library which was not made by me.

```
import shapefile
import json
```

```
class Coordinates(dict):
```

```
    #Most of the __init__ function was written by me, but prior to the start of the performance task
    for another project. It is not used in any written responses.
```

```
    def __init__(self,file,recordId,res=5,isl=1,countries=[],zoom=1,translate=
    [0,0],height=180,width=360):
```

```
    #Open shapefile with PySHP
```

```
        self.shp = shapefile.Reader(file)
        shapes = self.shp.shapes()
```

```
    #Define some variables
```

```
        svg = []
        svgPoints = []
        recordList = []
```

```
    #Counter variable to track progress
```

```
        i = 0
```

```
    #Loop through each country or territory
```

```
    for country in shapes:
```

```
        i = i + 1
```

```
    #Restrict parsing to specific countries
```

```
    if((self.shp.shapeRecord(i-1).record[recordId] in countries) or countries==[]):
```

```
        #Track progress
```

```
        print(i)
```

```
        #Reset/define some variables
```

```
            main = []
            rounded = []
```

```
        #Loop through each point
```

```
        for point in country.points:
```

```
            #Find lat and long
```

```
            data = ','.join(str(x) for x in point)
```

```
            long = float(data.split(',')[0])
```

```
            lat = float(data.split(',')[1])
```

```
        #Convert to positive numbers
```

```
            long = long + 180
```

```
            lat = lat + 90
```

```
        #Convert original unrounded data to string
```

```
            data = str(long) + ',' + str(lat)
```

```
            data = data + ' '
```

```
        #Check if point appears twice. If so, it is a complete polygon
```

```
        if data in main:
```

```
            main.append(data)
```

```
        #Round to reduce filesize
```

```
            pointRounded = [str(round(long*res)/res),str(round(lat*res)/res)]
```

```

        #List of rounded points
        rounded.append(pointRounded)

        #Exclude small islands
        if(len(rounded) > isl):

            #Add shape to list
            svgJson = {'Points':rounded,'Country':self.shp.shapeRecord(i-
1).record[recordId].replace(' ','')}
            svg.append(svgJson)

        #Reset lists
        main = []
        rounded = []
    else:

        #Round data as before and save the point
        main.append(data)
        roundedData = [str(round(long*res)/res),str(round(lat*res)/res)]
        if not(roundedData in rounded):
            rounded.append(roundedData)
        recordList.append(self.shp.shapeRecord(i-1).record[recordId].replace(' ',''))

#Create object from the points
self['data'] = svg
self['metadata'] = {'scale':zoom,'translate':translate,'height':height,'width':width}
self['recordList'] = recordList

```

#Builds the MapData class

```

class MapData(dict):
    def __init__(self,data,tlist='territory_list'):
        if(type(data) is dict):
            self['data'] = data
            try:
                #Finds which list is the territory list and marks it
                self['territory_list'] = self['data'][tlist]
            except:
                raise IndexError('Territory list not found in data.')
        else:
            raise TypeError('Data must be a dictionary.')

```

#Builds the infobox class

```

class Infobox(dict):
    def __init__(self):
        #Sets style information
        self['style'] = {}
        self['style']['border'] = {}
        self['style']['border']['stroke-width'] = 3
        self['style']['border']['color'] = {}
        self['style']['border']['color']['r'] = 0
        self['style']['border']['color']['g'] = 0
        self['style']['border']['color']['b'] = 0
        self['style']['color'] = {}
        self['style']['color']['r'] = 255
        self['style']['color']['g'] = 255
        self['style']['color']['b'] = 255
        self['style']['text'] = []
        self['text'] = []

```

#IMPORTANT: Text is based off of text containers that contain some text all formatted the same way. There can be multiple text containers in a line. The text containers can contain text or data.

```

def addText(self,text,textType='text',line=None,textId=None,font='Calibri',size=15,r=0,g=0,b=0):

```

```

#Create text
textDict = {}
textDict['type'] = textType

#Assign unique ID
if textId == None:

    #Default ID is just a number, but a string
    textDict['id'] = str(len(self['text']))
else:

    #Custom ID
    textDict['id'] = textId

#Decide which line text belongs on
if line == None:
    if len(self['text']) == 0:

        #Text starts on first line
        textDict['line'] = 0
    else:

        #Text continues on same line as before
        textDict['line'] = self['text'][-1]['line']
else:

    #Decide line manually
    textDict['line'] = line

textDict['text'] = text

#Create styles
styleDict = {}
styleDict['font'] = font
styleDict['size'] = size
styleDict['r'] = r
styleDict['g'] = g
styleDict['b'] = b

#Add to object
self['style']['text'].append(styleDict)
self['text'].append(textDict)

def removeText(self, textId):
    if type(textId) is int:
        del self['style']['text'][textId]
        del self['text'][textId]
    else:
        for text in self['text']:
            if(text['id'] == textId):
                del self['style']['text'][self['text'].index(text)]
                del self['text'][self['text'].index(text)]

class SVGMap(dict):
    def __init__(self, data=None, coords=None):
        self['map'] = coords
        self['choropleth'] = {}
        self['choropleth']['source'] = None

        self['choropleth']['colors'] = {}
        self['choropleth']['colors']['r'] = [0,255]
        self['choropleth']['colors']['g'] = [0,255]
        self['choropleth']['colors']['b'] = [0,255]

        self['layers'] = []

```



```
self['infobox'] = {}
```

```
#Supports two inputs: Hovering and clicking
```

```
self['inputs'] = {}
```

```
self['inputs']['onHover'] = ''
```

```
self['inputs']['onMouseOut'] = ''
```

```
self['inputs']['onClick'] = ''
```

```
self['data'] = data
```

```
#Code returned by returnCode is the output of the library portion.
```

```
def returnCode(self):
```

```
#Saves the Javascript code and defines the variable data to be the dataset
```

```
#as seen in the Javascript portion.
```

```
return 'var data = ' + json.dumps(self) + ';'
```

#IMPORTANT: This Python code is not part of the library but uses it.  
#Code from this portion is used once in a written response.

```
#Import library classes
from pymap import Coordinates
from pymap import Infobox
from pymap import SVGMap
from pymap import MapData
```

```
#Create map from coordinates. The shapefile is located at ne_10m_admin_0_countries and is from
Natural Earth Data.
countries = ['Italy','France','Germany','Switzerland','Spain','United
Kingdom','Poland','Ukraine','Czechia','Portugal','Ireland','Belgium','Netherlands','Romania','Hungary
','Republic of
Serbia','Slovakia','Luxembourg','Denmark','Norway','Iceland','Austria','Slovenia','Sweden','Moldova',
'Bulgaria','Greece','Finland','Belarus','Estonia','Latvia','Lithuania','Albania','Kosovo','Montenegro
','Croatia']
myMap = Coordinates('ne_10m_admin_0_countries',3,isl=5,height=600,width=1200,zoom=3,translate=
[150,125],countries=countries,res=100)
```

```
#Remove spaces
countriesNew = countries.copy()
countriesNew[5] = 'UnitedKingdom'
countriesNew[15] = 'RepublicofSerbia'
```

```
#Data sources: IKShPK (Kosovo), Worldometers (others)
cases =
[4159122,5877787,3602939,679510,3604799,4450777,2854079,2153864,1652840,842182,255672,1030071,1598164
,1072899,798147,706548,387523,69028,266503,119299,6537,637097,249310,1037126,253845,414192,377090,901
05,376341,127053,127978,265536,132015,106969,98852,350390]
myData = {'territory_list':countriesNew,'cases':cases,'original':countries}
myData = MapData(myData)
mySVGMap = SVGMap(myData,myMap)
```

```
#The cases dataset is used to color the map
mySVGMap['choropleth']['source'] = 'cases'
```

```
#Set colors
mySVGMap['choropleth']['colors']['r'] = [0,0]
mySVGMap['choropleth']['colors']['g'] = [0,0]
mySVGMap['choropleth']['colors']['b'] = [0,255]
```

```
#Create infobox
mySVGMap['infobox'] = Infobox()
```

```
#Line 0
mySVGMap['infobox'].addText('Country: ')
mySVGMap['infobox'].addText('data["data"]["data"]["original"][c]','int')
```

```
#Line 1
mySVGMap['infobox'].addText('COVID-19 Cases: ',line=1)
mySVGMap['infobox'].addText('data["data"]["data"]["cases"][c]','int')
```

```
#Set inputs
#Also calls Javascript function
#The ONLY place the functions showInfobox() and hideInfobox() are called is below
#They are automatically added to the HTML and Javascript
mySVGMap['inputs']['onHover'] = 'showInfobox([COUNTRY]);'
mySVGMap['inputs']['onMouseOut'] = 'hideInfobox([COUNTRY]);'
```

```
#Get javascript code
code = mySVGMap.returnCode()
```

```
#Save code
file = open('autogenerated.js','w')
file.write(code)
```