2. Programmierstrukturen und WFS

Konzept
Während WMS („Web map service“) „nur“ georeferenzierte Bilder liefert, also Bilder oder Geokacheln, deren einzelnen Pixeln Geokoordinaten zugeordnet sind, liefern WFS („Web Feature services“) geometrische Figuren (Polygone, Kreise, Punkte), die durch Geokoordinaten definiert sind. Bei einem WMS ist man darauf angewiesen, dass der Ersteller auf den Kartenkacheln, den Rastern, bestimmten Koordinaten Eigenschaften zugewiesen hat. Bei unserem Coronabeispiel sind bei einigen Koordinaten Kreise eingezeichnet. Beim Klick auf diese Kreise, oder beim Mouseover werden die Eigenschaften (‚Features‘) abgerufen. Damit dies bei jeder Auflösungsstufe funktioniert, musste die Auflösung mitgeliefert werden.
WFS sind unabhängig von der Auflösung, da hier die Koordinaten selbst angegeben sind.

Aufgabe:

1. **Erstelle die Datei „kartenprojekt_v3-0.html“ mit folgendem Code:**

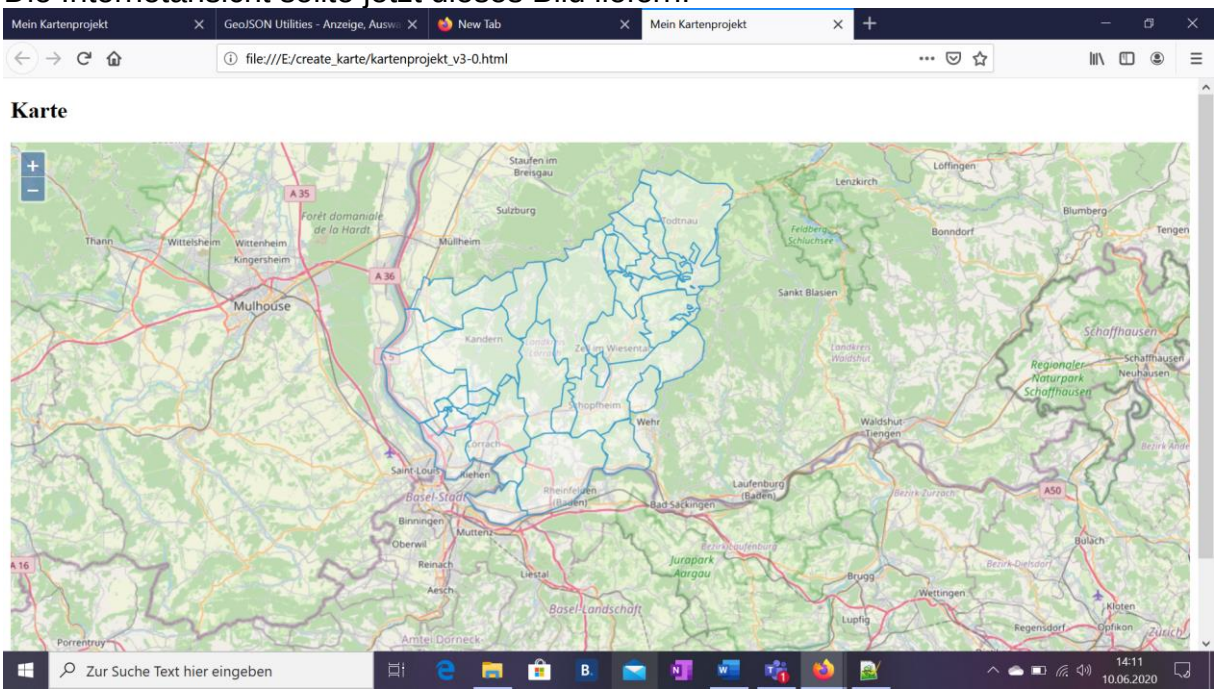
```
 <!DOCTYPE html>
<html lang="de">
  <head>
        <meta charset="utf-8"/>
    <title>Mein Kartenprojekt</title>
        <script
src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.
0/build/ol.js"></script>
        <script src="./gemeinden_lk_loe.geojson"></script>
        <link rel="stylesheet"
href="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3
.0/css/ol.css" type="text/css">
  </head>
  <body>
    <h2>Karte</h2>
    <div id="karte" class="karte">
</div>

        <script>

        var karte = new ol.Map({
        target: 'karte',
        layers: [
          new ol.layer.Tile({
            source: new ol.source.OSM(),
                name: 'osm'
          }),
                new ol.layer.Vector({
                        source: new ol.source.Vector({
                                features: (new
ol.format.GeoJSON()).readFeatures(geojson,{
                                featureProjection:  'EPSG:3857'
                                })

                        }),
                name: 'Loerrach'
```

```
                })

        ],

        view: new ol.View({
          center: ol.proj.fromLonLat([7.62241041028965,
47.595258340183065]),
          zoom: 10
        })
      });

        </script>
  </body>
</html>
```

Eine Datei, mit geojson Koordinaten deiner Lieblingsregion lässt sich z.B. unter http://opendatalab.de/projects/geojson-utilities/ herunterladen. Im Beispiel wurden die Gemeinden vom Landkreis Lörrach heruntergeladen und im SELBEN Verzeichnis wie die Karten-HTML Datei selbst als „gemeinden_lk_loe.geojson“ abgespeichert.

Im Editor muss die Datei „gemeinden_lk_loe.geojson“ geöffnet werden und dann vor die ersten Zeichen „var geojson = {“ und nach dem letzten Zeichen „};“ geschrieben werden.
Dies ist wichtig, damit die Datei mithilfe eines Skript-Tags eingebunden werden kann.
Wenn du einen Server besitzt oder einen localhost installiert hast genügt es unter source: new ol.source.Vector({ url: Pfad/zur/Datei/Dateiname }); anzugeben. Die Datei darf dann nicht wie oben beschrieben geändert werden. Dann entfällt natürlich auch der Skript-Tag.

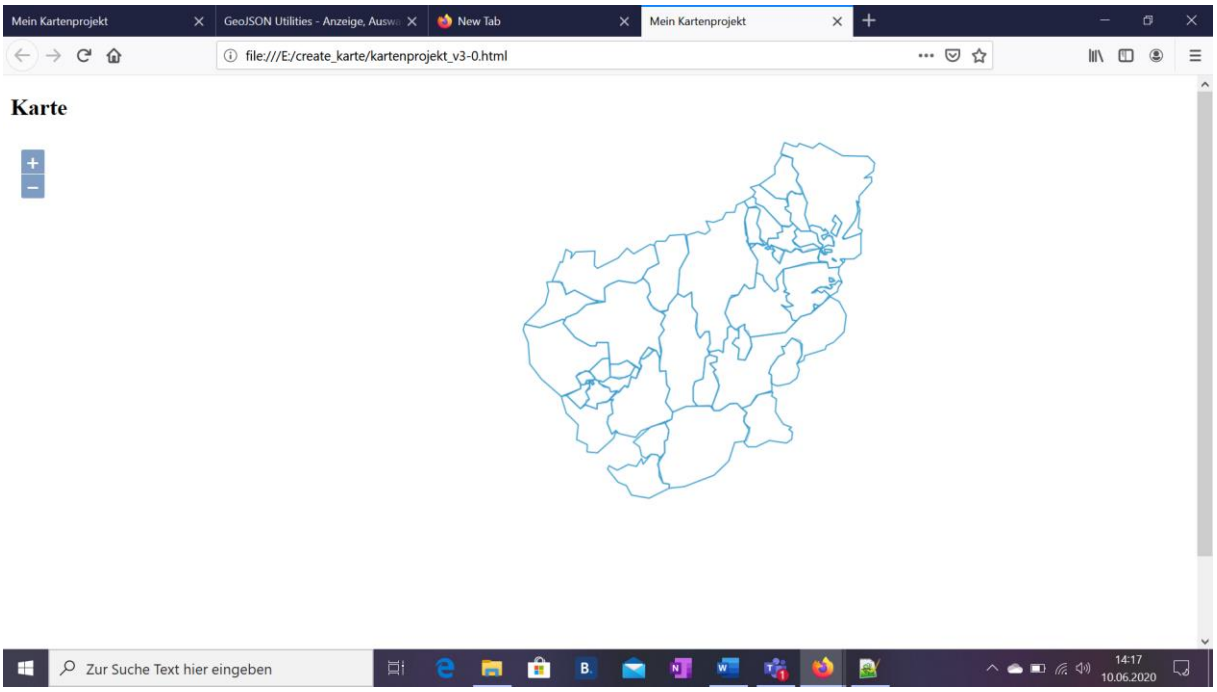
Die Internetansicht sollte jetzt dieses Bild liefern:



Natürlich abhängig davon, welche Punkte für das Kartencenter und welcher WFS benutzt wurde.

Aufgabe

2. **Blende den Tile-Layer aus, indem du alles Anweisungen bezüglich dieses Layers in Kommentare setzt. Ein mehrzeiliger Kommentar beginnt mit /* und endet mit */.**



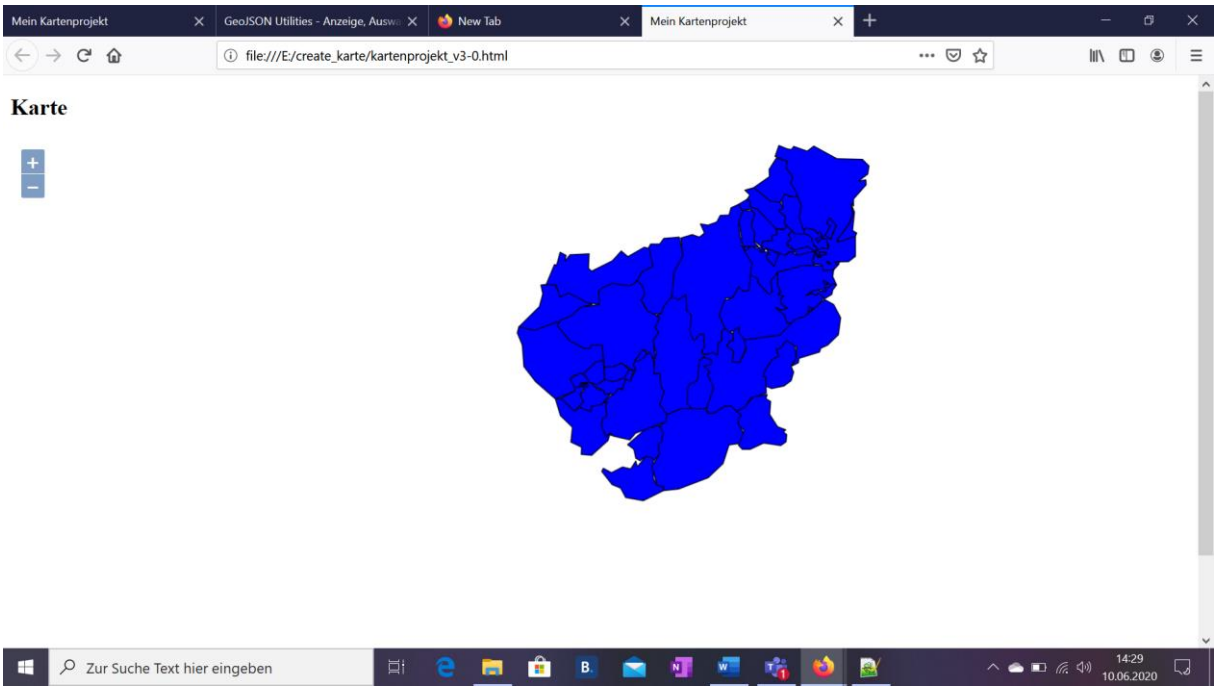
Wir wollen den neuen Layer jetzt „stylen“, d.h. die Gemeinden im Landkreis nach bestimmten Kriterien einfärben.
Dazu müssen wir eine Stylefunktion in der Layerdefinition anbringen:
style: styleGeoJsonFunction,

Wir bringen diese style-Eigenschaft des Layers nach source und vor der name-Eigenschaft unter. Daher muss auch nach der Eigenschaft ein Komma geschrieben werden, da ja noch eine Eigenschaft folgt.

Die Style-Funktion selbst muss als Funktion deklariert werden und enthält mindestens diesen Code:

```
function styleGeoJsonFunction(feature, resolution) {

var fillColour = new ol.style.Style({
        fill: new ol.style.Fill({
                color: 'blue' }),
        stroke: new ol.style.Stroke({
                color: 'black',
                width: 1
})
});

return [fillColour];
}
```

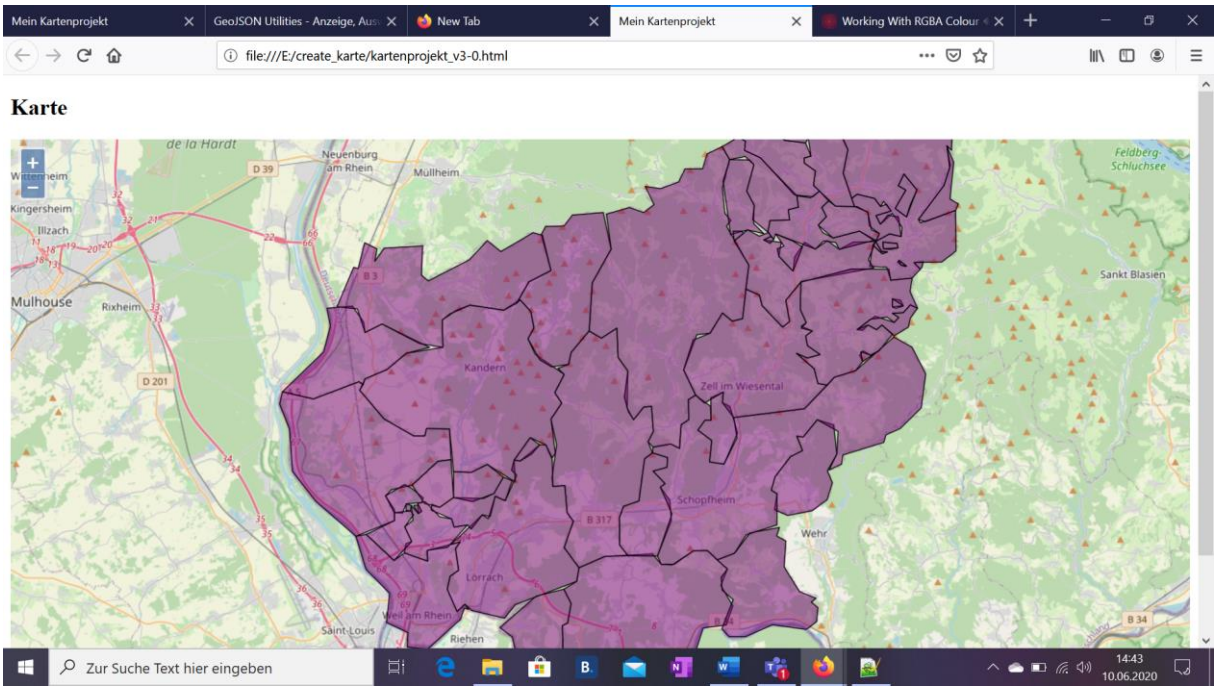
Nach Reload sollte die Browseransicht so aussehen:



Aufgabe

3. **Anstelle durch Klartextangabe: black, grey, green, kann auch eine beliebiege Farbe, sogar mit Transparenz ausgewählt werden. Benutze dazu die rgb-Nomenkaltur**. Beispiel: ' rgba(120, 0, 120, 0.5)'. Die Anführungszeichen sind wichtig. R,G und B stehen für Rot, Grün und Blau. Der letzte Wert (mit Punkt statt Komma [immer wichtig beim Programmieren]) gibt die Transparenz an. Suche dir einen Colorpicker im Internet und stelle deine Lieblingsfarbe ein. Wenn du eine transparente Farbe gewählt hast, wird auch die OSM-Karte hinter den Features wieder sichtbar. Blende die OSM-Karte wieder ein.

`Screenshot:`



Es sind Lücken in der WFS Karte erkennbar, dies kommt daher, dass ich den WFS nur in einer niedrigen Auflösung aus dem Netz geladen habe. Dies lässt sich unter der angegebenen URL unter „Optionen“ einstellen. Die Datei wird dann aber recht groß und es kann lange dauern, bis sie im Editor geladen wird.

Aufgabe:

4. **Lade im Netz oder unter Zuhilfenahme einer Gratisapp geojson-Daten herunter, z.B. Bewegungsdaten in deiner Umgebung auf dem Handy geloggt und binde die Datei wie oben beschrieben (nach Editieren mit Skript-Tag / oder unter url-Eingabe) ein.** Wichtig: wenn du beide WFS-Layer gleichzeitig benutzen möchtest, muss die zweite Variable einen anderen Namen haben. Z.B. var my_geojson = {…} . Ebenso muss sich natürlich der Dateiname unterscheiden. Je nach Ersteller kann es wichtig sein, die Quelle anzugeben. Datenquelle: … auf der Seite.

Unter Vorschau 1 findest du ein Beispiel, das auch schon durch eine Overlay-Funktion erweitert wurde (siehe unten).

IF-Programmstukturen:

Eine notwendige Programmierstruktur, die wir bisher noch nicht kennengelernt haben, ist die Verzweigung. Eingeläutet wird eine Verzweigung mit dem Schlüsselwort if gefolgt von einer Bedingung in einer RUNDEN Klammer.
Dann kommen zwei Anweisungsblöcke in geschweiften Klammern:
Anweisungsblock 1 wird ausgeführt, WENN die Bedingung erfüllt ist.
Dann kommt ein „else“, auf deutsch ansonsten, mit einem zweiten Anweisungsblock in geschweiften Klammern. Dieser Anweisungsblock wird ausgeführt, wenn die Bedingung nicht erfüllt ist.

Der Funktion zur Bestimmung der Füllfarbe werden automatisch zwei Parameter übergeben: feature und resolution.
Wenn wir ein Feature unserer geojson-features Datei anschauen, stehen da viele Eigenschaften drin:
{"type":"Feature","properties":{"ADE":6,"GF":4,"BSG":1,"RS":"083360014014","AGS":"08336014","SDV_RS":"083360014014","GEN":"Efringen-Kirchen","BEZ":"Gemeinde","IBZ":62,"BEM":"--","NBD":"ja","SN_L":"08","SN_R":"3","SN_K":"36","SN_V1":"00","SN_V2":"14","SN_G":"014","FK_S3":"R","NUTS":"DE139","RS_0":"083360014014","AGS_0":"08336014","WSK":"2009/01/01","DEBKG_ID":"DEBKGDL20000E24F","destatis":{"RS":"083360014014","area":43.75,"population":8642,"population_m":4319,"population_w":4323,"population_density":198,"zip":"79588","center_lon":"7,565548","center_lat":"47,649597","travel_key":"L13","travel_desc":"Südlicher Schwarzwald","density_key":"03","density_desc":"gering besiedelt"},"wikipedia":{"cid":"http://www.wikidata.org/entity/Q61640","name":"Efringen-Kirchen","AGS":"08336014","_official_website":"http://www.efringen-kirchen.de/","_local_dialing_code":"07628","_postal_code":"79588","_image":"http://commons.wikimedia.org/wiki/Special:FilePath/Mappach.jpg"}},"geometry":{"type":"MultiPolygon","coordinates":[[[[7.569710201598591,47.70356142676278],[7.587495459900158,47.68951433244719],[7.608912164838975,47.681298945280545],[7.617286661882573,47.681332244340524],[7.618338942528823,47.684768513272786],[7.635137835451453,47.68468141713708],[7.636201269796297,47.669484240920355]

,[7.626482621475001,47.66878734363157],[7.607862504603928,47.657034695120
11],[7.592856501228971,47.654845057329794],[7.588493329336471,47.65185057
582269],[7.595310701965581,47.644973364139894],[7.595293783361196,47.6398
01973580624],[7.566965294699721,47.632134902034565],[7.537903380420573,47
.64918050585068],[7.521079276205614,47.663847889162085],[7.51887983986138
7,47.684696627751855],[7.511589777772117,47.69675616191018],[7.5137690533
80703,47.70282381244302],[7.536160036780401,47.69934896058878],[7.5694358
7735111,47.707754030471854],[7.572312200443658,47.70620357829695],[7.5697
10201598591,47.70356142676278]]],[[[7.606591071383904,47.64459061486618],[7
.603534289182947,47.64475205087075],[7.603695948861411,47.64791287380102
],[7.607450249267414,47.64720342778888],[7.606591071383904,47.64459061486
618]]]]}},

Also neben ganz vielen Koordinaten, die das Gebiet definieren, z.B. unter dem Schlüssel „GEN“ der Gemeindename, unter dem Schlüssel „destatis“ ein Objekt in geschweiften Klammern mit weiteren Eigenschaften, z.B. „population“ die Einwohnerzahl.

Zurück zur Bedingung:
Wenn die Einwohnerzahldichte („population_density“) größer als 500 ist, wird die Gemeinde rot, sonst blau dargestellt.

In Pseudocode:

```
If (feature.get('destatis').population_density > 500) {
Style 1 color: blau z.B. rgba(0,0,120,0.5)
Stroke 1p, schwarz
}
Else {
Style 2 color:  rot z.B. rgba(120,0,0,0.5)
Stroke 1 p, schwarz

}
```

Hier ist auffällig: In den zwei Anweisungsblöcken steht zweimal fast derselbe Code. Es wäre viel einfacher, innerhalb der Bedingung nur die Farbe in einer Variablen festzulegen und dann nur einmal den Stil festzulegen.

Also:

```
If (feature.get('destatis').population_density > 500) {
Var farbe = 'rgba(0,0,120,0.5) '
}
else { var farbe = 'rgba(120,0,0,0.5) ' }

Style 2 (color  : farbe
Stroke 1 p, schwarz
```

Oder ausformuliert:

```
if (feature.get('destatis').population_density <500) {
                var color = 'rgba(0, 0, 120, 0.5)';
        }
        else {
                var color = 'rgba(120, 0, 0, 0.5)';
        }
var fillColour = new ol.style.Style({
        fill: new ol.style.Fill({
                color: color }),
        stroke: new ol.style.Stroke({
                color: 'black',
                width: 1
})
});

return [fillColour];
```

Aufgabe

5. **Füge weitere Abstufungen mit if, else if , else ein. Else if kann eine weitere Bedingung überprüfen, also if (dichte <200) {} else if (dichte <300) {} else if (dichte <400){} else {}.** In jedem Anweisungsblock wird eine Farbe definiert, nach Ende der Verzweigung wird eine Style Instanz erzeugt mit den fill und stroke Objekten und der Variablen fillColour zugewiesen, die anschließend in einer Liste zurückgegeben wird (return der Variablen in eckigen Klammern).

Bei drei Abstufungen <200, <300 (120,0,120,0.5) und die anderen sieht das Ergebnis etwa so aus: